Conga Configure Price Quote (CPQ)

Table of Contents

Learn to configure products, set up pricing, and generate quotes quickly and accurately using Conga CPQ.

CPQ Release Notes

Discover what's new in the latest release of Conga CPQ.

May22.03.31 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.03.31 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also

supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	**Latest Certified Version** *(Name / Number)*
Conga Approvals (Required if you are using Approvals)	13.0.300.47 | 13.300.47
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.5 | 3.233.5
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.2 | 14.173.2
Conga Configuration & Pricing **(New)**	14.0.1995.64 | 14.1995.64
Conga Contract Lifecycle Management	13.0.669.13 | 13.669.13
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4

Package	Latest Certified Version *(Name / Number)*
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.3 | 11.79.3
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.4 | 14.391.4
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

ⓘ For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00835393	CPQ-63911	On the cart page, when you reassign a collaboration request to a different user with Submit for Merge operation, then when the collaborator opens the request and clicks Configure Products, CPQ displays an error message.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart

Conga Internal ID	Description
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRN20230331

May 22.03.10 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.03.10 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name | Number)*
Conga Approvals (Required if you are using Approvals)	13.0.300.47 | 13.300.47
Conga Base Library **(New)** (Required if you are using Conga Configuration & Pricing)	3.0.233.5 | 3.233.5
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.2 | 14.173.2
Conga Configuration & Pricing **(New)**	14.0.1995.62 | 14.1995.62
Conga Contract Lifecycle Management	13.0.669.13 | 13.669.13
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.3 | 11.79.3
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.4 | 14.391.4
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later

Release	Package
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00824973	CPQ-63285	On the Order page, when you click the **Accept Order** button, CPQ displays an error message.
NA	CPQ-57274	In the Turbo mode, on the cart page, when you add a product with PLI having a price method as Related Price along with one of the products from the related price list items list, the price breakup records are created, but the **Related Product Name** is not displayed.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRN20230310

May22.02.03 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.02.03 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	**Latest Certified Version** *(Name / Number)*
Conga Approvals (**New**) (Required if you are using Approvals)	13.0.300.47 | 13.300.47
Conga Base Library (**New**) (Required if you are using Conga Configuration & Pricing)	3.0.233.4 | 3.233.4
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.2 | 14.173.2
Conga Configuration & Pricing (**New**)	14.0.1995.60 | 14.1995.60
Conga Contract Lifecycle Management (**New**)	13.0.669.13 | 13.669.13

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.3 | 11.79.3
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.4 | 14.391.4
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16

Package	Latest Certified Version *(Name | Number)*
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

This section describes the existing feature that is changed in this release.

Support for Asynchronous Pricing for Apex Job Monitoring

CPQ now supports retrieving details when cart asynchronous pricing happens. For more information, see Retrieving Apex Batch Job Information.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00824668	CPQ-61935	After an approval request has been rejected by a user, on the cart page, when the sales rep reprises a cart without adding any new products to the cart, the quote status changes from *Denied* to *Draft* instead of changing to *Approval Required*
00821512	CPQ-61560	In lightning mode, when the user is trying to download the CSV file for the Open Attribute Value Matrices, CPQ displays an error message.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).

Conga Internal ID	Description
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRN20230207

May22.01.12 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.01.12 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name | Number)*
Conga Approvals (**New**) (Required if you are using Approvals)	13.0.300.43 | 13.300.43

Package	Latest Certified Version *(Name / Number)*
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.3 | 3.233.3
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM) (**New**)	14.0.0173.2 | 14.173.2
Conga Configuration & Pricing (**New**)	14.0.1995.52 | 14.1995.52
Conga Contract Lifecycle Management	13.0.669.11 | 13.669.11
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0

Package	Latest Certified Version *(Name | Number)*
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration (**New**)	11.0.0079.3 | 11.79.3
Conga Quote Configuration Integration (**New**) (Required if you are using CPQ and Proposal Management)	14.0.0391.4 | 14.391.4
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

ⓘ For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

❗ Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

The following enhancement is new to this release.

Display a Spinner on the Service Catalog on Click of the Add to Cart Button

In the Turbo mode, CPQ introduces an admin setting *APTS_EnableBlockingCallsOnServiceCatalog* to display a spinner on the Service Catalog upon the click of the **Add to Cart** or **Update** buttons. When the admin setting is enabled, CPQ displays the spinner to ensure that the creation of related line items is completed for standalone products before you navigate away to a different page or click another button. Earlier, if you quickly navigated away from the Service Catalog after clicking **Add to Cart**, CPQ did not create related line items for standalone products.

For more information, see Configuring Admin Settings.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00816660	CPQ-60160	On the Incentives information page, when you update or create an incentive record without an expiration date field value, the record value does not get saved and CPQ displays the following error message. `Attempt to de-reference a null object`
00813146	CPQ-60747	On the catalog page, when you select an option inside bundle and then removes the quantity by mistake and add same option again CPQ displays the following error message on the cart page. `Attempt to de-reference a null`

Case Number	Conga Internal ID	Description
00814267	CPQ-61351	While working on Service CPQ in Turbo mode, while relating a standalone product, if you click **Add to Cart** and quickly navigate away from the Service Catalog, CPQ does not create related line items for the standalone product. **Resolution Info:** CPQ introduces an admin setting *APTS_EnableBlockingCallsOnServiceCatalog* to resolve this issue. When this admin setting is set to true and you click **Add to Cart** or **Update** on the Service Catalog, CPQ displays a spinner while the network calls in the background are completed. This prevents you from doing any other action after clicking **Add to Cart** or **Update** that can impact the network calls due to which CPQ will not create related line items for standalone service products. For more information, see Enhancements.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*

Conga Internal ID	Description
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20230113

May22.12.16 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.12.16 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name / Number)*
Conga Approvals (Required if you are using Approvals)	13.0.300.42 | 13.300.42
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.3 | 3.233.3
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing**(New)**	14.0.1995.50 | 14.1995.50
Conga Contract Lifecycle Management**(New)**	13.0.669.11 | 13.669.11
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0

Package	Latest Certified Version *(Name | Number)*
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

ⓘ For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

The following enhancement is new to this release.

Display a Spinner on the Service Catalog on Click of the Add to Cart Button

CPQ introduces an admin setting *APTS_EnableBlockingCallsOnServiceCatalog* to display a spinner on the Service Catalog upon the click of the **Add to Cart** or **Update** buttons. When the admin setting is enabled, CPQ displays the spinner to ensure that the creation of related line items is completed for service products (Product Type = Service) before you navigate away to a different page or click another button. Earlier, if you quickly navigated away from the Service Catalog after clicking **Add to Cart**, CPQ did not create related line items for service products.

For more information, see Configuring Admin Settings.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00807197	CPQ-58760	When a cart is configured and sent for approval and is rejected and returned to the cart, the Reprice button is visible even though none of the products are marked for repricing. After clicking the Reprice button, the approval status is changed to Draft which allows you to finalize the same cart.
00809941	CPQ-59512	When you add 40 line items to the cart and then add one miscellaneous item and click **Reprice**, the pricing for the first 40 line items is not displayed, and the repriced value is displayed only after the 41st line. This happens when you are using APTS_IncentivePricingBatchSize property to avoid certain governor limit issues related to incentives, because of which the custom logic in Pricing Callback fails.

Case Number	Conga Internal ID	Description
00813385	CPQ-59561	On the cart page, when you delete an asset and add the same assets from the installed products list, the cart is displayed in an incorrect sequence.
00816660	CPQ-60160	On the Incentives information page, when you update or create an incentive record without an expiration date field value, the record value does not get saved and CPQ displays the following error message. `Attempt to de-reference a null object`
00814267	CPQ-60957	In Service CPQ, while relating a standalone product, if you click **Add to Cart** and quickly navigate away from the Service Catalog, CPQ does not create related line items for the standalone product. **Resolution Info:** CPQ introduces an admin setting *APTS_EnableBlockingCallsOnServiceCatalog* to resolve this issue. When this admin setting is set to true and you click **Add to Cart** or **Update** on the Service Catalog, CPQ displays a spinner while the network calls in the background are completed. This prevents you from doing any other action after clicking **Add to Cart** or **Update** that can impact the network calls due to which CPQ will not create related line items for standalone service products. For more information, see Enhancements.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart

Conga Internal ID	Description
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20230222

May22.11.25 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.11.25 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name | Number)*
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.42 | 13.300.42
Conga Base Library **(New)** (Required if you are using Conga Configuration & Pricing)	3.0.233.3 | 3.233.3
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing **(New)**	14.0.1995.44 | 14.1995.44
Conga Contract Lifecycle Management **(New)**	13.0.669.10 | 13.669.10
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later

Release	Package
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00817023	CPQ-60277	Price rulesets are not working with related pricing as expected after you upgrade Conga CPQ to May '22.
00813327	CPQ-60186	When you terminate a one-time option, CPQ does not generate a negative billing schedule for that option.
00815004	CPQ-59769	When you use the Async Pricing API to process line items, some line items are remaining pending in the backend and you cannot proceed with the quoting process.

Case Number	Conga Internal ID	Description
00814647	CPQ-59744	When you select a large number of products (nearly 1500) for the Renew action, CPQ displays an error on the Installed Products page.
00806911	CPQ-58127	When you perform **Add/Remove** for a default option, CPQ displays the selected assets correctly on the Related Line Items pop-up but not on the sub-component pop -up.
00806909	CPQ-58125	When you perform **Add/Remove at all levels** on the bundle (bundle has an auto-included option) and add another asset, CPQ deletes all assets for the auto-included option.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*

Conga Internal ID	Description
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRN20221125

May22.11.04 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.11.04 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name / Number)*
Conga Approvals (**New**) (Required if you are using Approvals)	13.0.300.40 | 13.300.40
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.2 | 3.233.2
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing (**New**)	14.0.1995.37 | 14.1995.37
Conga Contract Lifecycle Management	13.0.669.8 | 13.669.8
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0

Package	Latest Certified Version *(Name | Number)*
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

ⓘ For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00808575	CPQ-59354	In the Proposal Approval flow, CPQ displays a CPU time limit error after invoking the Approval Check API.
00810588	CPQ-59406	In the TurboPricing flow, even when the product option price setting is selected as **Read-only**, the quantity is still editable on the cart page.
00811352	CPQ-59524	Clicking the name of the product on the cart page opens the attribute details of the product, wherein instead of the picklist name, the API name is displayed.
00796403	CPQ-59659	Copying a product in the cart page does not copy the usage tiers from the source product to the copied product.
00814647	CPQ-59744	On the Install Product page, when there are many products listed (Approximately 1500), and you select all the products and click Renew, CPQ displays an error message.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRN20221109

May22.10.14 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.10.14 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name / Number)*
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.37 | 13.300.37
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.2 | 3.233.2
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing **(New)**	14.0.1995.33 | 14.1995.33
Conga Contract Lifecycle Management **(New)**	13.0.669.8 | 13.669.8

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21

Package	Latest Certified Version *(Name | Number)*
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management **(New)** (Required if you are using Proposal Management)	12.0.0256.5 | 12.256.5

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00813398	CPQ-59368	Constraint rule execution does not filter assets based on the Asset Line Item Callback in client mode.
00811072	CPQ-58958	When you add more options than the **Option Pricing Chunk Size**, CPQ does not set the extended quantity on options correctly on the Cart.
PST-2584	CPQ-58794	In Turbo mode, after you delete a line item due to negative pricing, CPQ loads the cart with only 20 line items instead of showing all line items.
00807701	CPQ-58652	After you change an attribute expression, when you click the wrench icon next to the bundle product on the Cart, CPQ loads the bundle configuration page as blank.
00801273	CPQ-58334	When you click **Finalize** or **Submit for Approval** on the Cart, CPQ displays the following error: *Ramp dates must not overlap* This happens when you set **Enable Price Ramp**, **Enable Auto Ramp Creation**, and **Allow Price Ramp Overlap** to *True* and create a ramp overlap by keeping the periods of two ramps identical.
00804773	CPQ-58029	Related pricing does not work properly when a related option exists twice in a bundle. When both options go for pricing together, CPQ displays the following error: *Maximum stack depth reached: 1001*

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRN20221014

May22.09.27 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.09.27 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	**Latest Certified Version** *(Name / Number)*
Conga Approval (Required if you are using Approvals)	13.0.300.34 | 13.300.34
Conga Base Library **(New)** (Required if you are using Conga Configuration & Pricing)	3.0.233.2 | 3.233.2
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing **(New)**	14.0.1995.30 | 14.1995.30
Conga Contract Lifecycle Management **(New)**	13.0.669.7 | 13.669.7

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup **(New)** (Required if you are using CPQ Admin Console)	14.0.128.4 | 14.128.4
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21

Package	Latest Certified Version *(Name | Number)*
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.3 | 12.256.3

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00810481	CPQ-58758	When there are more than 1000 related price list items on a single price list item, CPQ displays the following error in Async pricing: *First error: Aggregate query has too many rows for direct assignment, use FOR loop*
00808238	CPQ-58557	Turbo Integration, the custom button **Submit for Approval** remains disabled even when the cart status is Approval Required.
00787695	CPQ-54831	The Apex Jobs page is displaying the UsageDataJobScheduler job as failed with the following error when the **Conga Base Library** package was installed: *Scheduler: failed to execute scheduled job: jobId: 7075e00000UaQ4a, class: common.apex.async.AsyncApexJobObject, reason: Dependent class is invalid and needs recompilation: Class Apttus_Base2.UsageDataBatchJob : Dependent class is invalid and needs re...* **Resolution**: CPQ no longer supports the **UsageDataJobScheduler** job, you must delete it from Scheduled Jobs before you upgrade the **Conga Base Library** package to release. This job was scheduled by the older version of the Conga Base Library package. For more information, see Preparing for Upgrade.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20221101

May22.09.09 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.09.09 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name / Number)*
Conga Approval **(New)** (Required if you are using Approvals)	13.0.300.34 | 13.300.34
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.1 | 3.233.1
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing **(New)**	14.0.1995.27 | 14.1995.27
Conga Contract Lifecycle Management	13.0.669.6 | 13.669.6

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup **(New)** (Required if you are using CPQ Admin Console)	14.0.128.3 | 14.128.3
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21

Package	Latest Certified Version *(Name | Number)*
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.3 | 12.256.3

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00803404	CPQ-57166	An attribute is not working as expected. When you configure a specific product, which is auto-including other products based on a constraint rule, CPQ displays an error on the Configuration or Cart page.
00794565	CPQ-55862	Turbo Integration: CPQ displays blank cart lines on Turbo quotes.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.

Conga Internal ID	Description
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20220927

May22.08.19 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.08.19 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name / Number)*
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.30 | 13.300.30
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.1 | 3.233.1
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing **(New)**	14.0.1995.25 | 14.1995.25
Conga Contract Lifecycle Management **(New)**	13.0.669.6 | 13.669.6
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128.1 | 14.128.1
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0

Package	Latest Certified Version *(Name | Number)*
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.3 | 12.256.3

ⓘ For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

The following feature is new to CPQ in this release.

Disable Email Notification for Price Calculation Batch Jobs

You can use the admin setting **APTS_DisableCartAsyncNotification** to disable email notifications you receive after completion of any async operation such as pricing calculation batch jobs in large carts.

For more information, see Turning off Async Operation Email Notification.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issue fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00780709	CPQ-52232 / CPQ-57941	When you apply a promotion, the Approval Required icon is displayed on the cart line item. But when you remove the promotion due to which the condition to display the icon is not fulfilled now, the Approval Required icon still remains on the Cart.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20220927

May22.07.25 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.07.25 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	**Latest Certified Version** *(Name / Number)*
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.23 | 13.300.23
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.1 | 3.233.1
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration **(New)** (Required if you are using CLM)	14.0.0173.1 | 14.173.1
Conga Configuration & Pricing **(New)**	14.0.1995.23 | 14.1995.23
Conga Contract Lifecycle Management	13.0.669.3 | 13.669.3

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup **(New)** (Required if you are using CPQ Admin Console)	14.0.128.1 | 14.128.1
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16

Package	Latest Certified Version *(Name | Number)*
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.3 | 12.256.3

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00792506	CPQ-55551	In the Turbo Pricing flow, when you click the **Save** button multiple times on the usage tier popup, CPQ creates multiple usage tiers.

Case Number	Conga Internal ID	Description
00801829	CPQ-56858	CPQ displays limitation error in the Apex Jobs after invoking the **syncCart** API. This happens when cart has 10000 line items.
NA	CPQ-57015	In the Smart Cart flow, CPQ displays a CPU time limit error after invoking the **finalizeCart** API. This happens when you finalize agreement with more than 3000 line items
NA	CPQ-57414	In the CPQ Admin UI, when you configure the **Currency** and **Value** fields for a formula field in a price ruleset, CPQ saves the default values upon saving the first time. This is rectified when you save that formula field the second time.
00802185	CPQ-57416	In the CPQ Admin UI, when you configure price rule with *%Discount* or *%Markup* in **Adjustment Type**, CPQ displays the "$" symbol in **Adjustment Amount**.

The following table lists the known issue fixed from the previous release.

Conga Internal ID	Description
CPQ-56513	CPQ does not update the finalized date and status of the agreement when an agreement is created from a quote with 5000 or 10000 lines.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart

Conga Internal ID	Description
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVB20220927

May22.07.08 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.07.08 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name | Number)*
Conga Approvals (Required if you are using Approvals)	13.0.300.20 | 13.300.20
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.1 | 3.233.1
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing **(New)**	14.0.1995.20 | 14.1995.20
Conga Contract Lifecycle Management	13.0.669.3 | 13.669.3
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management **(New)** (Required if you are using Proposal Management)	12.0.0256.3 | 12.256.3

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later

Release	Package
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00800970	CPQ-56790	In Turbo mode, when you use the RestoreLine API to send updates performed on a line item during and after approvals, any changes to the cart break the cart pricing and the cart stops to load.
00774166	CPQ-51928	The CurrencyISOCode on the Proposal product attribute record and the product attribute value record are mismatched.

Case Number	Conga Internal ID	Description
00108964	CPQ-46559	The line number on the cart page for line items with Pending Configuration is incorrect.
00101755	CPQ-43496	When you are reconfiguring a service bundle using the wrench icon, CPQ displays incorrect asset association on the UI.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-56513	CPQ does not update the finalized date and status of the agreement when an agreement is created from a quote with 5000 or 10000 lines.
CPQ-56097	You encounter Apex CPU time limit exceeded errors while performing the following actions in the large cart with 5000 lines. • Copying or deleting products • Relaunching the finalized cart
CPQ-55289 / CPQ-55186	Turbo Integrations: After you save or finalize the configuration, the status for Product Configuration is set as New instead of Saved/Finalized.
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55265	Turbo Integrations: When you remove an option group and perform hard revalidation on a bundle product, CPQ does not remove the product from the cart or option page and does not update the grand total.

Conga Internal ID	Description
CPQ-55229	When you click any custom action in the Lightning Console app from the Cart, the corresponding Visualforce page is displayed in the same tab.
CPQ-55221	In Lightening mode, a custom button redirection is encountering in the following error: *Error- Page doesn't exist. Enter a valid URL and try again*
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54995	Turbo Integrations: The custom field on the line item does not work in the constraint rule condition.
CPQ-54991	Turbo Integrations: CPQ calls the **updateAdHocGroup** method indefinitely when you switch to Sectional View on the Cart with both equipment and service products, resulting in multiple progress bars appearing on the Cart.
CPQ-54788	Turbo Integrations: When the secondary PLI of an option product is expired and a new PLI is added for the finalized quote, CPQ does not update Grand Total.
CPQ-54298	Turbo Integrations: Turbo cart does not revalidate approval checks after relaunching.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).

Conga Internal ID	Description
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20220927

May22.07.01 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.07.01 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name | Number)*
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.20 | 13.300.20

Package	Latest Certified Version *(Name / Number)*
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.1 | 3.233.1
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing	14.0.1995.18 | 14.1995.18 ⓘ There are no changes in the **Conga Configuration & Pricing** package. This patch includes TurboEngines backend change.
Conga Contract Lifecycle Management **(New)**	13.0.669.3 | 13.669.3
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117

Package	Latest Certified Version *(Name | Number)*
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256.2 | 12.256.2

ⓘ For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issue fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
PST-2498	CPQ-57005	When you perform several actions (for example, click **Configure Services** or **Configure Products**) on the cart, CPQ does not load the cart and the progress bar does not complete loading.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-55289 / CPQ-55186	Turbo Integrations: After you save or finalize the configuration, the status for Product Configuration is set as New instead of Saved/Finalized.
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.

Conga Internal ID	Description
CPQ-55265	Turbo Integrations: When you remove an option group and perform hard revalidation on a bundle product, CPQ does not remove the product from the cart or option page and does not update the grand total.
CPQ-55229	When you click any custom action in the Lightning Console app from the Cart, the corresponding Visualforce page is displayed in the same tab.
CPQ-55221	In Lightening mode, a custom button redirection is encountering in the following error: *Error- Page doesn't exist. Enter a valid URL and try again*
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54995	Turbo Integrations: The custom field on the line item does not work in the constraint rule condition.
CPQ-54991	Turbo Integrations: CPQ calls the **updateAdHocGroup** method indefinitely when you switch to Sectional View on the Cart with both equipment and service products, resulting in multiple progress bars appearing on the Cart.
CPQ-54788	Turbo Integrations: When the secondary PLI of an option product is expired and a new PLI is added for the finalized quote, CPQ does not update Grand Total.
CPQ-54298	Turbo Integrations: Turbo cart does not revalidate approval checks after relaunching.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.

Conga Internal ID	Description
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20220927

May22.06.24 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.06.24 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name | Number)*
Conga Approvals (**New**) (Required if you are using Approvals)	13.0.300.19 | 13.300.19
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.233.1 | 3.233.1
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing (**New**)	14.0.1995.18 | 14.1995.18
Conga Contract Lifecycle Management	13.0.669.1 | 13.669.1
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0

Package	Latest Certified Version *(Name | Number)*
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration (**New**) (Required if you are using CPQ and Proposal Management)	14.0.0391.3 | 14.391.3
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (**New**) (Required if you are using Proposal Management)	12.0.0256.2 | 12.256.2

ⓘ For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issue fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00799119	CPQ-56356	When you try to relate a second service to an asset, you cannot proceed to the Catalog page from the Installed Products page.

The following table lists the known issues fixed from the previous release.

Conga Internal ID	Description
CPQ-55999	When you apply adjustments with bucket adjustments for bundle products, CPQ does not create adjustment lines.
CPQ-54007	In Turbo mode, the Cart page does not display any product after re-configuration. This issue occurs when the PLI of the product, in which a constraint rule is added, is inactive.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-55289 / CPQ-55186	Turbo Integrations: After you save or finalize the configuration, the status for Product Configuration is set as New instead of Saved/Finalized.

Conga Internal ID	Description
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55265	Turbo Integrations: When you remove an option group and perform hard revalidation on a bundle product, CPQ does not remove the product from the cart or option page and does not update the grand total.
CPQ-55229	When you click any custom action in the Lightning Console app from the Cart, the corresponding Visualforce page is displayed in the same tab.
CPQ-55221	In Lightening mode, a custom button redirection is encountering in the following error: *Error- Page doesn't exist. Enter a valid URL and try again*
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54995	Turbo Integrations: The custom field on the line item does not work in the constraint rule condition.
CPQ-54991	Turbo Integrations: CPQ calls the **updateAdHocGroup** method indefinitely when you switch to Sectional View on the Cart with both equipment and service products, resulting in multiple progress bars appearing on the Cart.
CPQ-54788	Turbo Integrations: When the secondary PLI of an option product is expired and a new PLI is added for the finalized quote, CPQ does not update Grand Total.
CPQ-54298	Turbo Integrations: Turbo cart does not revalidate approval checks after relaunching.

Conga Internal ID	Description
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20220927

May22.06.03 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.06.03 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name / Number)*
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.16 | 13.300.16
Conga Base Library **(New)** (Required if you are using Conga Configuration & Pricing)	3.0.233.1 | 3.233.1
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing **(New)**	14.0.1995.14 | 14.1995.14
Conga Contract Lifecycle Management **(New)**	13.0.669.1 | 13.669.1
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7

Package	Latest Certified Version *(Name / Number)*
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration **(New)**	11.0.0079.2 | 11.79.2
Conga Quote Configuration Integration **(New)** (Required if you are using CPQ and Proposal Management)	14.0.0391.2 | 14.391.2
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256 | 12.256

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00798963	CPQ-56256	In Turbo mode, Deal Guidance and Approvals are not working in Saved Cart scenarios.
NA	CPQ-55672	In Turbo mode, there is a delay in the appearance of buttons on the Configure Services page.
00792522	CPQ-55575	In Turbo mode, CPQ performs a full search of options on the bundle configuration page on every keypress-keyup event. If you have over 1000 options, you encounter performance issues on the UI. **Resolution Info:** CPQ now performs a search only on the **Enter** key.
NA	CPQ-55496	In Turbo mode, the inclusion rule for service option products does not trigger as intended.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-55999	When you apply adjustments with bucket adjustments for bundle products, CPQ does not create adjustment lines.
CPQ-55289 / CPQ-55186	Turbo Integrations: After you save or finalize the configuration, the status for Product Configuration is set as New instead of Saved/Finalized.
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55265	Turbo Integrations: When you remove an option group and perform hard revalidation on a bundle product, CPQ does not remove the product from the cart or option page and does not update the grand total.
CPQ-55229	When you click any custom action in the Lightning Console app from the Cart, the corresponding Visualforce page is displayed in the same tab.
CPQ-55221	In Lightening mode, a custom button redirection is encountering in the following error: *Error- Page doesn't exist. Enter a valid URL and try again*
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54995	Turbo Integrations: The custom field on the line item does not work in the constraint rule condition.
CPQ-54991	Turbo Integrations: CPQ calls the **updateAdHocGroup** method indefinitely when you switch to Sectional View on the Cart with both equipment and service products, resulting in multiple progress bars appearing on the Cart.
CPQ-54788	Turbo Integrations: When the secondary PLI of an option product is expired and a new PLI is added for the finalized quote, CPQ does not update Grand Total.

Conga Internal ID	Description
CPQ-54298	Turbo Integrations: Turbo cart does not revalidate approval checks after relaunching.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-54007	Turbo Integrations: The Cart page does not display any product after re-configuration. This issue occurs when the PLI of the product, in which a constraint rule is added, is inactive.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).
CPQ-51757	When you click **Submit for Approval** button on the Cart page, CPQ opens a new tab in browser instead of opening the approval page in same tab.

DOC ID: CPQMAY22PRNREVA20220927

May22.05.13 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May22.05.13 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	Latest Certified Version *(Name / Number)*
Conga Approvals (Required if you are using Approvals)	13.0.300.1 | 13.300.1
Conga Base Library (Required if you are using Conga Configuration & Pricing)	3.0.232 | 3.232
Conga Billing (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing **(New)**	14.0.1995.5 | 14.1995.5
Conga Contract Lifecycle Management	13.0.669 | 13.669
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer	8.0.0029 | 8.29

Package	Latest Certified Version *(Name | Number)*
Conga CPQ Setup (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer	7.0.0007 | 7.7
Conga DocuSign Api	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration **(New)**	11.0.0079.1 | 11.79.1
Conga Quote Configuration Integration **(New)** (Required if you are using CPQ and Proposal Management)	14.0.0391.1 | 14.391.1
Conga Quote DocuSign Integration	4.0.0021 | 4.21
Conga Quote Echosign Integration (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management (Required if you are using Proposal Management)	12.0.0256 | 12.256

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later

Release	Package
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

There are no new features in this release. Keep checking the Conga Documentation Portal for new updates.

Enhancements

There are no enhancements in this release.

Data Model Changes

There are no data model changes in this release.

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00783174	CPQ-52448	When you clear the value you added in a numeric attribute field or enter "Null" on the Configuration page, CPQ displays the following error: *Apttus_Config2.ProductAttributeValueTrigger: execution of BeforeUpdate caused by: System.NullPointerException: Attempt to de-reference a null object (Apttus_Config2).*

Case Number	Conga Internal ID	Description
00790551	CPQ-55277	When you click **Cancel** on the confirmation pop-up to finalize the cart with revalidation warning, CPQ finalizes the cart regardless.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-55999	When you apply adjustments with bucket adjustments for bundle products, CPQ does not create adjustment lines.
CPQ-55289 / CPQ-55186	Turbo Integrations: After you save or finalize the configuration, the status for Product Configuration is set as New instead of Saved/Finalized.
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55265	Turbo Integrations: When you remove an option group and perform hard revalidation on a bundle product, CPQ does not remove the product from the cart or option page and does not update the grand total.
CPQ-55229	When you click any custom action in the Lightning Console app from the Cart, the corresponding Visualforce page is displayed in the same tab.
CPQ-55221	In Lightening mode, a custom button redirection is encountering in the following error: *Error- Page doesn't exist. Enter a valid URL and try again*
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.

Conga Internal ID	Description
CPQ-54995	Turbo Integrations: The custom field on the line item does not work in the constraint rule condition.
CPQ-54991	Turbo Integrations: CPQ calls the **updateAdHocGroup** method indefinitely when you switch to Sectional View on the Cart with both equipment and service products, resulting in multiple progress bars appearing on the Cart.
CPQ-54788	Turbo Integrations: When the secondary PLI of an option product is expired and a new PLI is added for the finalized quote, CPQ does not update Grand Total.
CPQ-54298	Turbo Integrations: Turbo cart does not revalidate approval checks after relaunching.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-54007	Turbo Integrations: The Cart page does not display any product after re-configuration. This issue occurs when the PLI of the product, in which a constraint rule is added, is inactive.
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).

DOC ID: CPQMAY22PRNREVB20220927

May '22 Release Notes

In these release notes, you can find packages, requirements, features, enhancements, fixed issues, and known issues for the CPQ May '22 release. For documentation updates, see What's New in CPQ Documentation.

> This documentation may describe optional features for which you have not purchased a license; therefore your solution or implementation may differ from what is described here. Contact your Customer Success Manager (CSM) or Account Executive (AE) to discuss your specific features and licensing.

All Conga customers have FREE access to getting started content and release training in the Conga Learning Center. To take your training further, ensure your organization has access to the Conga Learning Pass, which is a training subscription service. Click here to learn more.

Packages

The following packages and dependencies are required to upgrade to this release and use its new features. These are the *minimum* required versions; later versions are also supported. Prerequisites for each feature can be found in its documentation. Packages marked (**New**) are new for this release.

> You can register your org for the Conga Push Upgrade. Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, it also ensures all the Conga published managed packages are on the latest versions for your registered orgs. For more information, see Registering for Conga Push Upgrade.

Package	**Latest Certified Version** ***(Name / Number)***
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.1 | 13.300.1
Conga Base Library **(New)** (Required if you are using Conga Configuration & Pricing)	3.0.232 | 3.232
Conga Billing **(New)** (Required if you are using Conga Billing)	8.0.296 | 8.296

Package	Latest Certified Version *(Name | Number)*
Conga CLM Configuration Integration **(New)** (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing **(New)**	14.0.1995 | 14.1995
Conga Contract Lifecycle Management **(New)**	13.0.669 | 13.669
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer **(New)**	8.0.0029 | 8.29
Conga CPQ Setup **(New)** (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup **(New)** (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer **(New)**	7.0.0007 | 7.7
Conga DocuSign Api **(New)**	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14

Package	Latest Certified Version *(Name / Number)*
Conga Quote CLM Integration **(New)**	11.0.0079 | 11.79
Conga Quote Configuration Integration **(New)** (Required if you are using CPQ and Proposal Management)	14.0.0391 | 14.391
Conga Quote DocuSign Integration **(New)**	4.0.0021 | 4.21
Conga Quote Echosign Integration **(New)** (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management **(New)** (Required if you are using Proposal Management)	12.0.0256 | 12.256

For more information on installing Conga managed packages, refer to Installing CPQ Packages. For more information on upgrading Conga managed packages, refer to Upgrading CPQ.

System Requirements and Supported Platforms

For information pertaining to the requirements and recommendations you must consider before you proceed with the installation of the Conga product, see System Requirements and Supported Platforms Matrix.

Upgrade Notes

For more information on the prerequisites and post-upgrade tasks, see Upgrading CPQ.

Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

New Features

The following features are new to CPQ in this release.

Support for Translated Picklist Values in Document Generation

Non-English users can now get translated values of picklist during document generation. To achieve this, CPQ introduces the **APTS_TranslatePickValuesInDocs** admin setting. Earlier,

CPQ used the values defined in English in the generated documents instead of the translated values.

For more information, see Using Translated Picklist Values in Document Generation.

Express Proposal

The Express Proposal feature allows you to generate and send proposal in fewer clicks. After you add and price the products in the cart, click **Express Quote** to finalize the cart, select a template, generate the quote and email the document to the customers, and mark the quote as presented. You can use the Express Proposal from Catalog, Configuration, and Cart pages by clicking the **Express Quote** button. You can configure this feature in single-click mode or prompt mode. In prompt mode CPQ displays a popup on the Cart page where you must select a template for document generation, add recipients, and define the **Presented** checkbox; in single-click mode, CPQ executes all tasks automatically.

For more information, see Configuring Express Proposal and Using Express Proposal.

Configure the Execution of the RelatedLineItem Trigger

CPQ introduces a new global flag **isTriggerInitiatedUpdate** to control the execution of the RelatedLineItem trigger. When the value of the flag is set to *True*, CPQ skips the execution of the RelatedLineItem trigger. CPQ does not use the out-of-the-box logic to allocate the net price of the service line item to all related line items on updating the related line items from the custom code.

For more information, see Configuring the Execution of the RelatedLineItem Trigger and Working with the Service Cart.

Enhancements

The following section describes the existing features that are changed (or are no longer supported) in this release.

Adjust Column Width on Cart

You can now adjust the width of the columns on the Cart page by dragging the right border of the columns.

For more information, see Viewing the Cart in Grid View.

Support for Automatic Unlocking of Idle Cart with Concurrent Access

You can now define a time limit for unlocking idle carts with concurrent access, which are currently open in edit mode by a Sales rep. When the time limit is exceeded, other Sales reps can open the cart in edit mode, who previously could only open them in read-only mode. You can configure the time limit using the **Cart Edit Access Idle Timeout in Minutes** setting in Config System Properties.

For more information, see Enabling Cart Locking for Concurrent Access and About Cart Locking for Concurrent Access.

Enhanced Bundle-Option Scenarios with Auto Cascade Selling Term Calculation

CPQ supports more bundle-option scenarios for calculating the selling term with the Auto Cascade Selling Term setting set to True and/or False.

For more information, see About Quantity and Selling Term Calculation Using Default Pricing Settings.

Ability to Filter Products by Product Codes in the Criteria Section of Promotions

You can filter products by their codes in the Criteria section while defining a promotion. You could filter products by their names in the Criteria section so far.

For more information, see Defining the Criteria of a Promotion.

Support for Every X Get X Promotions for Multiple Products

You can create a promotion of type Every X Get X where X can be multiple products or any product from a group provided the quantity requirements are fulfilled for each product. In prior releases, you could select only a single product for X in *For Every X Get X* promotion.

For more information, see Defining the Benefits of a Promotion.

Display Benefit Quantity for Every X Get X or Every X Get Y Promotions

In a *For Every X Get X* or *For Every X Get Y* scenario, when a line item gets the benefit, CPQ displays the quantity of the product that received the promotion benefit, on the **Benefit Quantity** column on the Multiple Adjustments pop-up. When a line item gets the benefit in these scenarios, the entire quantity of that line item may not always receive the benefit and hence you may want to view the quantity that received the benefit.

For more information, see Applying Promotions on Line Items in the Cart.

Invoke Validation Callback on Click of View Cart

You can now invoke Validation Callback on the Cart page upon clicking the **View Cart** on the Mini-Cart. When the Validation Callback is invoked, CPQ executes the custom logic defined in the callback and displays any validation errors on the Cart page. You must enable the **Run Validation Callback On Add** setting in Config System Properties.

For more information, see Validation Callback Class and Config System Properties.

Launch the Installed Products Page Directly from a URL for the Add/Remove Flow

You can launch the Installed Products page directly from a URL for the Add/Remove flow, where you can modify the association of assets with services. CPQ enables you to define a custom formula action to launch the Installed Products page directly from the custom cart page.

For more information, see Configuring a Custom Button to Launch the Installed Products Page Directly for the Add/Remove Flow and Modifying the Association of Assets with Services by Launching the Installed Products Page Directly.

Enhanced the CPQRESTServiceExecuteJob REST API

The CPQRESTServiceExecuteJob REST API can now trigger the following maintenance or batch jobs:

- CategoryMaintenanceBatchJob
- BundleUpdateJob
- UpdatePriceQJob

- RepriceCartJob

Earlier, this API supported only the ConfigDataStaticResourceUpsertJob.

For more information, see Service Execute Job.

Deprecated Features

The following fields or settings are deprecated in this release.

- The **Allowable Action** field on the Details tab while creating a price rule and price pipeline rule
- The **Criteria Product** and **Product Family Criteria** fields on the Benefits section while creating a For Every X Get X promotion
- The **Revalidation Product Columns** setting in Config System Properties

Data Model Changes

The following objects and fields are introduced to or changed in the system or data model in this release.

Object	Fields	Description	System/ User	New/ Changed
Quote/Proposal		Quote/Proposal object which holds dates and line items, many quotes/proposals may be related to a single opportunity, a quote/proposal may have many detail lines.		Changed
	Param Data	The parameter data associated with the quote/ proposal	System	New
Related Line Item		Holds the relationships between line items		Changed

Object	Fields	Description	System/ User	New/ Changed
	Is System Calculated	Indicates whether the weightage is calculated by the system. System calculated weightage may be overridden by the user.	User	New

Fixed Issues

The following table lists the issues fixed in this release. If any actions are required, they will be listed in this table.

Case Number	Conga Internal ID	Description
00454714	CPQ-55151	When you send the proposal for Esignature using Adobe Sign, CPQ does not select default agreement template on the page to select attachment.
NA	CPQ-54791	In a Quick Quote flow, when you perform finalize, save, abandon, or close actions on the Cart page, you were redirected to the opportunity. **Resolution info:** Now, when you finalize, save abandon, or close on the cart in a Quick Quote flow, you are redirected to the Quote Details page created when you click **Create Quick Quote**. For more information, see Creating Quick Quotes.
00787319	CPQ-54739	If you perform an operation that requires order line items or other order related objects to be deleted, the operation fails. **Resolution Info:** The administrator must grant Delete permission to you or enable Bypass sharing.
NA	CPQ-54699	During agreement renewal, the name validation causes an issue because of the current date format with slashes (/). **Resolution Info:** The new date format on renewal agreement is 2021-12-01.

Case Number	Conga Internal ID	Description
NA	CPQ-54520	You cannot control the execution of the RelatedLineItem trigger. **Resolution Info:** CPQ introduces a new global flag isTriggerInitiatedUpdate to control the execution of the RelatedLineItem trigger.
NA	CPQ-54233	After selecting an option during configuration, when you click **Add More Products**, CPQ displays the following error on the Catalog page: `"Id not specified in an update call"`
NA	CPQ-52518	Syncing line items with opportunities using the **syncWithOpportunity()** API in the Smart Cart flow does not work.
00116771	CPQ-52445	CPQ does not cascade attributes to option lines automatically when the attributes are defined in one of the Product Attribute Values extension objects and the options are auto-added by inclusion criteria.
00782324	CPQ-52435	On the Configuration page, when you select an attribute that is defined in one of the Product Attribute Value Extension objects, CPQ clears the selection automatically. This happens when you define a product attribute rule and a constraint rule on that attribute.
NA	CPQ-52413	You cannot launch the Installed Products page directly from a URL for the Add/Remove flow, where you can modify the association of assets with services.
NA	CPQ-52379 / CPQ-52265	You cannot create a promotion of type Every X Get X where X can be multiple products.
00453119	CPQ-52309	Even when you have configured Translation Workbench to translate the merge field values, CPQ does not populate translated values on generated documents.
NA	CPQ-52264	In a For Every X Get X or For Every X Get Y scenario, when a line item gets the benefit, CPQ does not have a column to display the quantity of the product that received the promotion benefit, on the Multiple Adjustments pop-up.

Case Number	Conga Internal ID	Description
00779547	CPQ-52168	If a requested edit-access user abandons the cart, CPQ does not send an email notification to the requestor. This issue occurs if you do not enable **Keep Abandoned Carts** in Config System Properties.
00454242	CPQ-51179	When you deselect an option that auto-included a product, CPQ removes the product auto-included by the parent bundle of that option along with the product auto-included by that option itself.
NA	CPQ-51060	The **Auto Cascade Selling Term** setting does not work in bundle-options scenario when you update the selling term from the Cart grid.
00117629	CPQ-50905	When you click **Add/Remove** on the related line item pop-up to add service products, CPQ hides options on the Service Catalog. This happened when you define a search filter to hide service options.
00115760	CPQ-50738	CPQ does not display Validation Callback warning messages on the Cart page when you perform **View Cart** action from the Mini Cart.
NA	CPQ-49744	The **Auto Cascade Selling Term** setting does not work if the first option has it False and the second option has it True.
NA	CPQ-49702	Selling term is incorrect for an option when the bundle has two options where one option has **Auto Cascade Selling Term** set to True and other option has **Auto Cascade Selling Term** set to False, and you decrease the bundle term and reprice the cart.
NA	CPQ-49679	When **Auto Cascade Selling Term** set to True and the end date is cascaded from the primary charge type, CPQ does not calculate the selling term for quarterly and half yearly secondary charge types.
00101897	CPQ-43905	When you apply a promotion to more than 100 lines in the Cart, you encounter an error during reprice.
00102188	CPQ-43891	You are unable to adjust the column width on the Cart page. **Resolution Info:** CPQ now supports adjustment to the width of the column. For more information, see "Adjust Column Width on Cart" section under Enhancements.

Case Number	Conga Internal ID	Description
00101542	CPQ-43263	You cannot filter a product by product code in the Criteria section of promotions.

Known Issues

The following table provides the cumulative list of known issues up to this release.

Conga Internal ID	Description
CPQ-55289 / CPQ-55186	Turbo Integrations: After you save or finalize the configuration, the status for Product Configuration is set as New instead of Saved/Finalized.
CPQ-55277	When you click **Cancel** on the confirmation pop-up to finalize the cart with revalidation warning, CPQ finalizes the cart regardless.
CPQ-55272	Turbo Integrations: When you delete the secondary PLI of a bundle product, remove an existing option product, add a new option product to the bundle, and perform hard revalidation on the bundle product, you cannot finalize the quote.
CPQ-55265	Turbo Integrations: When you remove an option group and perform hard revalidation on a bundle product, CPQ does not remove the product from the cart or option page and does not update the grand total.
CPQ-55229	When you click any custom action in the Lightning Console app from the Cart, the corresponding Visualforce page is displayed in the same tab.
CPQ-55221	In Lightening mode, a custom button redirection is encountering in the following error: *Error- Page doesn't exist. Enter a valid URL and try again*

Conga Internal ID	Description
CPQ-55159	Turbo Integrations: When you perform hard revalidation, CPQ performs soft revalidation automatically.
CPQ-54995	Turbo Integrations: The custom field on the line item does not work in the constraint rule condition.
CPQ-54991	Turbo Integrations: CPQ calls the **updateAdHocGroup** method indefinitely when you switch to Sectional View on the Cart with both equipment and service products, resulting in multiple progress bars appearing on the Cart.
CPQ-54788	Turbo Integrations: When the secondary PLI of an option product is expired and a new PLI is added for the finalized quote, CPQ does not update Grand Total.
CPQ-54298	Turbo Integrations: Turbo cart does not revalidate approval checks after relaunching.
CPQ-54278	Turbo Integrations: You encounter the following error while finalizing the revalidated cart if the revalidation callback is configured in your org: *entity is deleted*
CPQ-54022	Turbo Integrations: Attribute based pricing is not working for Numeric Expression.
CPQ-54007	Turbo Integrations: The Cart page does not display any product after re-configuration. This issue occurs when the PLI of the product, in which a constraint rule is added, is inactive.
CPQ-52448	When you clear the value you added in a numeric attribute field or enter "Null" on the Configuration page, CPQ displays the following error: *Apttus_Config2.ProductAttributeValueTrigger: execution of BeforeUpdate caused by: System.NullPointerException: Attempt to de-reference a null object (Apttus_Config2)*
CPQ-49315	Config asset pricing criteria are not functioning on a single ramp renewed line (renew one ramp = true).

DOC ID: CPQMAY22RN20220504

About CPQ

A quote or a proposal is a formal statement of promise that lists the products and services to be sold to a customer at a defined price. Quote creation involves configuring products and services, pricing them, and generating quotes based on predefined rules. CPQ is a sales tool for companies to generate quotes for orders quickly and accurately. CPQ produces accurate quotes making complex products, pricing, and business rules centralized, automatic, and available in real-time.

As an **administrator**, you can use CPQ to achieve Configuration, Pricing, and Quoting tasks involved in generating a quote. Configuration involves creating products, options, attributes, categories and associating them appropriately with each other for visibility on Catalog. A product can be created as a standalone product or as a bundle product with options and attributes. You can control the selection of a product on the catalog by configuring constraint rules. You can also control the selection of attributes on the configuration page of a product by setting up attribute-based configuration for that product.

You can set up pricing structures for the products so that the price for all products is calculated accurately. Pricing comprises of mainly two components: Price Lists and Price List Items. A price list controls the visibility of products to the user. A price list contains several price list items, each linked to a product. CPQ calculates the price for each product based on the applied price list, price list items, and various pricing and discounting rules. You can price any products or bundles based on their features or options selected. You can set up pricing rules, constraints, dependencies, and extraneous variables in CPQ to calculate the price for any product.

You can enable quote collaboration to allow a Sales Representative to get other users to contribute in configuration and pricing. The Collaborators can add or remove products and adjust pricing on the same configuration.

You can create templates for quote creation with details such as configuration and pricing details, a summary of the quote, and associated opportunity. In addition, by integrating CPQ with your contract processes, you can automate renewal quotes based on previously agreed upon pricing and terms.

You can manage the assets of a customer with a variety of billing models to ensure efficient collections and accounting. You can define the asset management functions with different data objects to track quote and contract details until an order is fulfilled.

As a **Sales Representative**, you need to configure products according to your customer's requirements. A product can be standalone or bundle with options and attributes. You can configure products accurately based on predefined business rules and constraints.

You can manage product pricing and adjustments. The price calculation of a product also includes its options and attributes. CPQ allows you to apply discounts, markups, promotions to further adjust the price of the product.

After you finalize the configuration and pricing, you can generate quotes. You can generate a quote using predefined templates and send the quote to the customer for approval using an email. Your customer can use E-signature to sign and accept the quote. In addition, by integrating quoting software with your contract processes, you can automate renewal quotes based on previously agreed upon pricing and terms.

The following table lists the tasks that administrators and users can perform using CPQ.

Administrator	Administrator/User

• Manage products
  • Create, edit, clone, and delete standalone, bundle (including multi-level bundles), option products, option groups, and product groups
  • Create multi-level bundles
• Manage categories and category hierarchies
  • Create, edit, and delete categories and category hierarchies
• Manage attributes
  • Create, edit, and delete product attribute values
  • Create, edit, and delete product attribute groups
• Configure product visibility
  • Set up Refine You search
  • Configure Search Filter(CPQ)
  • Configure visibility rules
• Configure product comparison
  • Set up feature sets
• Configure constraint rules
  • Create, edit, clone, and delete constraint rules, constraint rule conditions, and constraint rule actions
• Attribute-based configuration
  • Create, edit, and delete attribute value matrices
  • Create, edit, and delete product attribute rules
  • Associate product attribute groups with products
• Manage pricing
  • Create, edit, clone, and delete price lists and price list items
  • Configure related pricing
  • Enable price ramps
  • Configure tiered pricing
  • Enable multiple adjustments at line items
  • Enable contract pricing
  • Configure price breakup
	• Manage quotes
	  • Create, Edit, Clone, and delete quotes
	• Search products on the catalog
	  • Refine Your Search
	  • Product comparison
	  • Guided selling
	• Configure and add products to the cart
	  • Configure product on the catalog by selecting options and attributes
	  • Import a saved favorite configuration
	  • Collaborate to configure products
	• Price products
	  • Create price ramp
	  • Define tiered-price
	  • Apply adjustments and bucket adjustments
	  • Apply promotions
	• Finalize Cart
	  • Save cart configuration as favorite
	• Finalize quote
	  • Generate a proposal document
	  • Present a proposal document
	  • Analyze a quote
	  • Accept a quote
	  • Activate an order
	• Manage assets
	  • Search assets
	  • Renew, change, swap, or terminate assets
	  • View asset transaction history
	  • Integrate assets and contracts
	  • Generate bills for assets
	• Manage services
	  • Configure a service product
	  • Manage a service cart

Administrator	Administrator/User
• Configure cost and profitability • Set up unit of measure and conversion rate • Define currency rounding • Manage price rule • Create price ruleset, price rule • Create price matrices • Manage Cart • Configuring save as favorite • Enable quote collaboration • Cart locking for concurrent access • Manage quote/proposal • PDF security for generated quote/proposal documents • Synchronize cart lines with quote • Set up quick quote mode • Manage promotions • Configure coupons • Configure promotions • Manage assets • Configure asset management flow • Manage asset-based ordering • Configure asset-based ordering • Set up asset-based pricing • Configure installed product page • Set up opportunity-based renewals ⓘ For administrator functionality, see CPQ for Administrators.	• Manage service pricing ⓘ For end-user functionality, see CPQ for Users.

Key Terminology

It is important to understand how terms are used when working with CPQ.

Term	Definition
ABO	Asset-based ordering (ABO) functionality enables you to service and manage existing orders based on customer requirements.

Term	Definition
Adjustments	Any discounts or markups applied to the product on the cart.
Administrators	Individual responsible for maintaining and mapping product catalogs and pricing.
Asset Line Items	Child of Asset object, these records depict the actual calculation of assets for each line item. After you receive payments for an order, the order line item changes to an Asset line item.
Assets	Assets define a purchased product or service. An asset is associated with an account. After being processed and fulfilled, new quotes and orders result in new assets, which are listed on a customer's account and a line item from an asset becomes an Installed Product.
Asset Pricing	A pricing method that allows you to leverage the sale price of an asset for its ongoing management.
Attribute-Based Pricing	The price is determined by factoring in attributes of a product.
Attribute Groups	A group that contains attributes that are shared by multiple products. For example, attributes of a computer are color, size, RAM and so on. These attributes are shared across different types of product such as a computer.
Attributes	Features of a product, such as color, size, weight, and more.
Base Extended Price	This is derived by Base Price x Quantity x Term.
Batch Job	A maintenance job that updates the database and checks for configuration inconsistencies. A batch maintenance job runs maintenance jobs in regular intervals or runs multiple maintenance jobs at a time.
Bucket Adjustments	A bucket (field on Adjustment Line Item object) is a container for the adjustments that a sales representative applies on a line item in the cart.

Term	Definition
Bundle within a Bundle	A feature that allows you to configure products that may contain multi-level bundles. This feature also allows the configuration of multi-level bundles with the ability to show ramps or tiers contained within the sub-bundles. You can also apply constraint rules to multi-level bundles.
Bundled Products	A combination of standalone products that offer added value to the customer while increasing overall sales.
Bundled Services	Services sold together, as a package, rather than separately.
Cart	A product and pricing view for the user to review all configuration and pricing information at a glance.
Cart Views	Pre-defined criteria to filter products on the Cart page.
Catalog	A view that allows hierarchical categorization of products for users to search through and add to their configuration.
Categories	High-level logical groupings of products. Their grouping affects the way the end-user sees them for selection in a product catalog. A category is created for browsing products in the selection pages, or they can be created to use options groups for a bundled product. Products are associated with a category through a category hierarchy. Category hierarchies are maintained using the Hierarchy Manager. They are also used to organize product prices into logical groupings.
Category Hierarchy	Products are associated with a category through a category hierarchy. Category hierarchies are maintained using the Hierarchy Manager.
Change	An ABO action that is available on the Installed Products page to increase or decrease the quantity of assets. Applicable for all product types.
Clone	To replicate a field, record, template, or other objects.
Configure Products	Button on Quote/Proposal record that allows end users to start configuring a quote.

Term	Definition
Configuration Rules	CPQ administrator defines configuration rules to help the customer or user to select the required product or a range of products per requirements. Rules make it easier to organize products on the catalog, simplifying the decision-making experience of the customer or the user.
Constraint Rule Action	Captures information about rule actions, which are only applied when the rule conditions are satisfied.
Constraint Rules	Rules that drive automatic inclusion, exclusion, recommendation or replacement of products selected in the catalog page. Each Constraint Rule is composed of three parts: the Rule Detail, the Rule Condition, and the Rule Actions.
Contract Price List	A price list that helps the customer keep track of specific price agreements as applicable for that account.
Contract Pricing	A way to negotiate a price for a product that is not being purchased in the current quote. CPQ uses a set of agreed upon prices for a customer for all the subsequent transactions.
Cost Model	A cost model is a container that holds all cost types (parent and child cost types).
CPQ Console	The page where you can manage most product and pricing functions.
CSR Flow	Customer Service Representative (CSR) flow. A user can launch the Installed Products page from an account and skip the usual Quote/Proposal flow by using the CSR Flow.
Customer Priority	Field on an Account record that defines the customer's priority or rating.
Deal Guidance	A guided approach to Sales Representatives (other users) that helps decide whether the offered price for any line item is more, less, or not profitable based on the set criteria for the company.
Defer Pricing	Performing pricing of products only after you click the Go to Pricing button as opposed to performing pricing as and when you add products to the cart or delete products from the cart.

Term	Definition
Delta Price	The change to the net price of the asset due to the latest order (may include any ABO action). It is positive for increments, negative for decrements, and zero when there is no change. It is the difference between the net price of an asset line item and the net price of line item after an asset action has been performed.
Delta Quantity	The change in quantity for the asset as a result of the latest order (which may include any ABO action). It is positive for increments, negative for decrements, and zero when there is no change. It is the difference between the quantity of an asset line item and the quantity of the line item.
Discrete	A type of Price Matrix dimension value. When you select the discrete option, the application considers specific matrix values only.
Document	A document is the record of the finalized product configuration in PDF, DOC, DOCX, and RTF format generated from the quote. The document includes the information about products, pricing, and custom details that you configured.
Evergreen Asset	An asset without an expiration date. The asset is billable forever until the subscription is cancelled (evergreen billing).
Express Proposal	A proposal that you can generate and send in fewer clicks.
Large Cart	Any quote that contains more than 50 line items. A line item can either be a standalone product or a bundle.
Line Item	Represents a product or service line item.
Merge Assets	Merge is an ABO action that merges multiple assets created over a period as a result of actions such as new sale, quantity increment, split. This action consolidates individual streams of assets into a single asset, which can also help you avoid duplicates.
Objects	A definition of a specific type of information you can store in Salesforce. Some objects are native to Salesforce (such as Contacts or Accounts), while others are specific to Apttus functionality (such as Proposals or Agreements).

Term	Definition
Opportunity	A contact or an account that has a possibility or intent of business with you. An opportunity record is created when a new opportunity is created in the CRM. Sales representatives create quotes for an existing opportunity.
Option Group	A group of product options that are associated with a bundled product.
Order	A confirmation document created by CPQ for a customer before delivering the goods or services.
Organization	A deployment of CRM that has a defined set of licensed users. Your organization includes all of your data and applications and is separate from all other organizations. The short form of organization is *org*.
Price Adjustments	Price rules that can be made in CPQ (example: Price Matrix and Price Ruleset).
Price Dimensions	Represents a price criteria dimension.
Price List Items	Each price list item includes the list price for the product including details of cost such as per unit, flat price, one time, recurring, and more. Each price list item also represents the different ways that a customer is charged for a product; for example: license fees, implementation fees, etc. Price list items are categorized into Price Lists.
Price Lists	Containers of items that are grouped in a price list. A price list controls which products are visible to the end-user. A price list contains several price list items; each linked to a product. A product can be set up with one or more price list items.
Price Matrix (Matrices)	These are an advanced pricing concept used to define tiered pricing paradigms, or complex pricing structures with multiple criteria. Common examples are pricing tiers for a product based on user count or particular customer or transactional dimensions.
Price Ramps and Price Tiers	Pricing mechanisms that establish pricing that varies by time (ramp) or quantity (tiers).

Term	Definition
Price Rule	Represents a single rule in a price ruleset.
Price Rulesets	Price Rulesets are a mechanism to allow particular families, categories or groupings of products to have either line item pricing adjustments applied or summary pricing adjustments applied. Typical examples of these are volume discounting rules or promotional pricing rules.
Price Waterfall	A bar graph where line item fields are plotted vertically and cost is plotted horizontally.
Product Attribute Group	A group that contains attributes that are shared by multiple products. For example, attributes of a computer are color, size, RAM and so on. These attributes are shared across different types of product such as a computer and a laptop. Attributes are grouped together as a product attribute group.
Product Attribute Value	Represents the attribute values for a product class. For example, color has attribute values such as red, green, blue, orange and so on.
Product Console	The product console helps administer product properties and associate products with other CPQ artifacts.
Product Family	Product Family is a field on the product record with configurable values. This construct allows you to create combinations of products with similar characteristics/qualities for use in a Price Rule.
Product Group	A logical grouping of one or more product records. This construct allows you to create combinations of products with similar characteristics/qualities for use in a Price Rule.
Product Hierarchy	Defines the structure of the product catalog according to the product configuration. A CPQ administrator configures products and its structure.
Product Option Group	A list of option groups that are associated with a product.

Term	Definition
Products	A product or service that can be sold on its own as a standalone item, an option of other products, or as part of a bundled product. Conga utilizes the standard Salesforce Products2 table to store product records and Conga-specific fields on those records that manage the behavior of products.
Product Cloning	Replication of products from Source organization to Target Organization during product migration.
Promotions	Promo codes or coupons to apply additional adjustments.
Proposal	The Conga custom object used to record information about a specific quote for a specific Opportunity and Account. The tab for the object is presented as Proposals and the record is stored as a Quote/Proposal.
Quick Quote	A quote that is created directly from an opportunity where the Sales rep navigates to the Catalog page, configure products, and finalizes the cart without having to create a proposal first.
Quote	A structured definition of a prospective sale that contains product configurations, pricing, and customer opportunity information.
Quote Collaboration	A feature you can use to enable multiple users to work together on the same quote.
Records	A collection of fields that store information about a contact, an account, or an opportunity.
Relate	Defines an ABO action, available on the Installed Products page. It is used to display all the relevant service products for already purchased products (equipment). Eligibility rules are used to control the products listed in the catalog.
Related Asset Line Items	Depicts the relationship between the type of products, service, and equipment. When a service is related to equipment or normal asset, corresponding records are populated in Related Asset Line Items (From) and Related Asset Line Items (To) related lists.
Related Pricing	Deriving the price of a line item based on the price of another line item, product group or formula fields.

Term	Definition
Related Purchase	A tab on the Installed Products page that lists the related services for an already purchased product (asset). The eligibility rules control the products listed in this tab.
Renew	A tab on the Installed Products page. The renew action is applicable for all product types. Supports options to choose the renewal date in the form of Proposal End Date, Asset End Date, Farthest End Date, and a manually entered Renewal Date.
Sales Representative or Sales Rep	Individuals responsible for configuring and pricing the products on the cart and generate quotes.
Selling Term	A field on the line item object (column in cart) that defines the length of time the seller will allow the buyer to pay for a product.
Smart Cart	A cart with a large number of line items that are priced in groups that are divided based on the threshold and split criteria to avoid hitting governor limits.
Standalone Products	Refers to a device or software that is self-contained, one that does not require any other devices or software to function.
Standalone Services	Refers to a service that is self-contained, one that does not require any other service to function.
Service CPQ	A feature that caters to customer requirements related to services for existing or newly purchased products from Apttus. A sales rep associates a service product eligible for the equipment or an asset. The Eligibility rules, which are client-side constraint rules, play a vital role in guiding the association between a service product and an asset. A sales rep can set an existing asset as a service or equipment while the service pages (Installed Products, Catalog, Config, and Cart pages) leverage the end to end cycle for service management.
Smart Search	A search definition that is configured to search for products in large and complex product catalogs, with multiple levels of categories and stricter rules.
Split Asset	Split is an ABO action that divides an existing asset into multiple split lines to negotiate an upsell quote with customers.

Term	Definition
Swap	Swap is an ABO action that is available as a tab on the Installed Products page. The Swap action uses the constraint rule of type = Replacement for execution.
Terminate	A tab on the Installed Products page. The Terminate action is used to terminate the contract for a product. Applicable for all product types.
Tiered Pricing Model	The longer a subscription term on a quote, the lower the monthly rate will be.
TurboConfig	A configuration engine microservice on the Flexible Computing Platform that computes complex product configurations and product rules.
TurboEngines	A concurrent processing engine provided by Conga comprising various microservices that process and sync product configurations, pricing configurations, and data.
TurboPricing	A pricing engine microservice on the Flexible Computing Platform that computes complex pricing computations and callbacks.
Validating	To check or prove the validity or accuracy of a value or instance.

Glossary for Approval Stages

Term	Definition
Draft	New Quote Created.
Approval Required	Changes to the Quote or Quote Line Items have triggered a need for approval.
In Review	The quote is in the process of being Approved or Denied.
Approved	Quote has been approved by management.
Generated	Quote document has been generated.
Presented	Quote document was presented to the end customer, either via email or manually.

Term	Definition
Accepted	The quote was accepted by the end customer.
Denied	Quote has been denied by management.

Glossary for Proposal Actions

Term	Definition
Configure Products	An action button on the Quote page that starts a new configuration process for the Quote, or reconfigure an existing set of products on the Quote.
Generate Proposal	An action button on the Quote page that generates a Proposal document and attaches it to the Quote/Proposal record.
Present Proposal	An action button on the Quote page that presents the attached document to the Quote to the end customer through an email.
Accept Proposal	An action button on the Quote page that refers to an end customer Accepting the final proposal. This button synchronizes all Summary lines with the Opportunity Products for the Related Opportunity.
Make Primary	An action button on the Quote page that sets this Quote to primary and all other Quotes for the Opportunity as non-primary. Also enables the Synchronize with Opportunity button.
Synchronize with Opportunity	An action button on the Quote page that synchronizes the finalized quote with the Opportunity, each time the cart is Finalized. If you are using automatic synchronization, you must remove this from the page layout.
Create Agreement with Line Items	An action button on the Quote page that that enables the user to create a contract after the quote is accepted.

Glossary for Related Lists

Name	Description
Actions	Lists the various actions you can execute for this stage of the quoting process. A list of actions with the description of their function is available in Proposal Actions.
Line Items	Contains the detailed line items (bundles, options, and standalone) of a finalized configuration. This corresponds to the Detail tab in configuration.
Summary	Contains the summary view (standalone and bundles) of a finalized configuration. This corresponds to the Summary tab in configuration.
Summary Groups	Contains the totals for the finalized configuration according to the product hierarchy including a Grand Total.
Notes & Attachments	Enables you to attach any reference document or file relevant to a quote/proposal. The proposal document when generated from the template is stored by the application in this section.
Open Activities	Enables you to create and assign new task corresponding to a quote.
Approval History	Initiate and track approvals corresponding to the quote. Note: Approvals can be initiated only if the corresponding approval processes have been set up by the System Administrator.
Activity History	Lists an Audit history of significant changes during the life of the quote.

For more information about terms used with Conga products, see Conga Product Glossary.

What's New in CPQ Documentation

This section lists changes in the documentation to support each release.

May '22

Document	Publication Date	Topic	Description
May '22	22 May 2023	Creating Price List Items	Updated the description of the **Charge Type** field that it is mandatory while creating a price list item.
	26 Apr 2023	Managing Constraint Rules	Added a limitation about CPQ displaying a constraint rule message on the cart in the "Client" mode.
		Creating Constraint Rule Actions	Added a limitation about CPQ displaying a constraint rule message on the cart in the "Client" mode.
		Configuring Products from the Catalog	Added a note about waiting until the constraint rule execution is complete while changing attributes.
		Retrieving Products and List Prices for a Price List	Added a note about associating a price list and price list items to a category.
	17 Mar 2023	Configuring Admin Settings	Updated the description of the **Value** field for the *APTS_DefaultEmailContactName* admin setting.
	22 Feb 2023	Configuring Admin Settings	Updated the description of the *APTS_EnableBlockingCallsOnServiceCatalog* admin setting.
	17 Feb 2023	Creating Price List Items	Updated the description of the **Min/Max Price Applies To** field.

Document	Publication Date	Topic	Description
		Applying the Default Pricing to Products	Added a note in the description of the **Default Selling Term** field about calculation of the selling term on the cart.
		Viewing the Cart in Grid View	Added the "Calculating the selling term of recurring fee" point.
		Enabling Auto Ramp Creation	Removed a note about a non-supported feature from Scenario 1.
	13 Feb 2023	Pricing Extension Callback Class	Removed the limitation about Pricing Extension Callback being supported only for Smart Cart.
		Validation Callback Class	Added information about the **Validate** button for *validateCart()* method.
		Option Filter Callback Class	Updated the description.
	19 Dec 2022	Use Cases: Related Pricing	Added the "Use Case 6: Locations" section.
	01 Nov 2022	Preparing for Upgrade	Added steps to delete the **UsageDataJobScheduler** job.
	17 Oct 2022	Assigning the Configuration to the Collaborator or Queue	Added information about CPQ not supporting auto-included primary line item for collaboration.
		Config System Properties	Added the **Grid Render Threshold** config system property.
	11 Oct 2022	Config System Properties	Added a note about the deprecation of **Show Attributes In Cart**.
		Configuring Cart Page Settings	Added a note about the deprecation of **Show Attributes In Cart**.

Document	Publication Date	Topic	Description
	28 Jun 2022	Using Proposal Document Generation	Added a section about uploading local files to quotes.
	15 Jun 2022	Adding Line Items	New topic.
		About Auto Sync and Manual Sync	Added information about Accept Quote.
	31 May 2022	Sharing Settings	Added an entry for the Bundle Component View object in the table.
	16 May 2022	Finalizing the Shopping Cart	Updated the topic with a note about re-finalizing a finalized quote.
	04 May 2022	About Reports	Updated the topic to add all Conga reports.
		About Dashboards	Updated the topic to add all Conga dashboards.
	27 Apr 2022	Pricing Products	Updated the description of the following fields: • Option Price • Extended Price • Adjusted Price • Flat Option Price
		System Fields	New topic.
		Merge Web Service	Added a note about Salesforce API version change.
		Creating Proposal Document	Changed Salesforce API version.
		Creating Proposal Document with Draft Indication	Changed Salesforce API version.

Document	Publication Date	Topic	Description
		Generating Documents for Proposal with Large Number of Line Items	Changed Salesforce API version.
		Generating Documents Asynchronously	Changed Salesforce API version.
		Creating Price Matrices	Added a note for the **Enable Date Range** checkbox.
		Configuring Custom Button to Access Configuration Page Directly	Updated the structure of URLs to navigate directly to the Configuration page.
	06 Apr 2022	Configuring Admin Settings	New topic.
		Using Shield Platform Encryption	New topic.
		Use Cases: Related Pricing	New topic.
		About Quantity and Selling Term Calculation Using Default Pricing Settings	Updated the topic with more bundle-option scenarios for calculating selling term.
		Configuring Express Proposal	New topic.
		Using Express Proposal	New topic.
		Using Translated Picklist Values in Document Generation	New topic.
		Viewing the Cart in Grid View	Added information about adjusting column width.

Document	Publication Date	Topic	Description
		About Cart Locking for Concurrent Access	Added information about **Cart Edit Access Idle Timeout in Minutes** setting.
		Enabling Cart Locking for Concurrent Access	Added information about **Cart Edit Access Idle Timeout in Minutes** setting.
		Defining the Criteria of a Promotion	Updated the topic with information about filtering products by product codes.
		Defining the Benefits of a Promotion	Updated the topic with information about support for Every X Get X promotions for multiple products. Also, removed the step about selecting **Criteria Product** and **Product Family Criteria**.
		Applying Promotions on Line Items in the Cart	Updated the topic with information about: • Support for Every X Get X promotions for multiple products. • Displaying benefit quantity for Every X Get X or Every X Get Y promotions.
		Configuring a Custom Button to Launch the Installed Products Page Directly for the Add/Remove Flow	New topic.
		Modifying the Association of Assets with Services by Launching the Installed Products Page Directly	New topic.
		Configuring the Execution of the RelatedLineItem Trigger	New topic.
		Working with the Service Cart	Updated the topic with a note about the global flag **isTriggerInitiatedUpdate**.

Document	Publication Date	Topic	Description
		Validation Callback Class	Added information about enabling **Run Validation Callback On Add** setting.
		Config System Properties	• Added the setting **Run Validation Callback On Add**. • Updated the description of the setting **Selling Term Calculation Method**. • Updated the description of **Revalidation Product Columns (D)**. This setting is deprecated.
		Swapping an Asset	Updated the note about the **Selling Term Calculation Method** setting.
		Creating Price Rules	Removed the step about selecting an **Allowable Action**.
		Creating Price Pipeline Rules	Removed the step about selecting an **Allowable Action**.
		Service Execute Job	New topic.
		Revalidating the Product Configuration	Updated the topic description.
		About Search Filters (CPQ)	Added a note.
		Enabling CPQ for Partner Community Users	New topic.
		Configuring a Profile	New topic.
		Configuring Field-Level Security	New topic.
		Configuring a Permission Set	New topic.

Document	Publication Date	Topic	Description
		Sharing Settings	New topic.
		Configuring a Partner Community User	New topic.
		Configuring a Partner Community Site	New topic.
		Configuring a Custom Button on the Quote Object for Partners	New topic.
		Validating the Partner Community User	New topic.
		Disabling the Clone Icon on the Cart Page	Renamed the topic from "Hiding the Clone Icon for a Cart Line Item" and updated the topic.
		Configuring a Price Pipeline	Updated the image of the Manage Price Pipeline page.
		Configuring Price Waterfall to a Price Pipeline	Updated the image of the the **WATERFALL SETUP** tab and updated the description of the **Criteria** field.
		Configuring Price Pipeline Ruleset	Added an image for the New Price Pipeline Ruleset page and updated the description of the **Ruleset Criteria** field.
		Using Price Waterfall	Renamed the title and reorganized the content.

December '21

Document	Publication Date	Topic	Description
December '21 Rev C	21 Dec 2021	Suspending an Asset	New topic.
		Resuming an Asset	New topic.
		Suspending Assets	New topic.
		Resuming Assets	New topic.
		CPQ Asset-Based Ordering APIs	• Added the Suspend and Resume REST API information. • Updated the Terminate REST API description
December '21 Rev B	17 Dec 2021	Config System Properties	Added **Check Many Options**.
		Configuring Custom Settings for Different Flows	Added **Check Many Options**.
		Configuring Products from the Catalog	Added description of **Confirm Option Selections**.
		Configuring Mass Option Selection on the Configuration Page	New topic.
		Merge Web Service	New topic.
		Creating Proposal Document	New topic.
		Creating Proposal Document with Draft Indication	New topic.

Document	Publication Date	Topic	Description
		Generating Documents for Proposal with Large Number of Line Items	New topic.
		Generating Documents Asynchronously	New topic.
December '21 Rev A	07 Dec 2021	Config Data Cache	Updated the topic description.
		Configuring CSS Override	Updated the description of the example.
		Viewing the Cart in Grid View	Added information about constraint rule error message display on the Cart page.
		Configuring Product Sort	Added an example.
		Using Proposal Document Generation	Added a note about **Merge Call Timeout Millis**.
		Option Filter Callback Class	Updated the description.
		Managing Constraint Rules	Added a note about cancelled line item.
		Creating Quick Quotes	Updated the description.
		Retrieving Apex Batch Job Information	Added list of apex jobs.

Document	Publication Date	Topic	Description
		Use Case: Early Renewal of Ramped Bundle Assets with Options	Added some notes about required settings.
		Use Case: Renewed Ramped Assets Start from the End Date of the Last Ramp Line	Added some notes about required settings.
		Adding Miscellaneous Items	Updated the topic with information about order line item creation for miscellaneous item.
		CPQ Asset-Based Ordering APIs	Added a note about having Conga Digital Commerce installed to use ABO REST APIs.

Document	Publication Date	Topic	Description
December '21	03 Nov 2021	Use Case: Renewed Ramped Assets Start from the End Date of the Last Ramp Line	New topic
		Displaying Assets from Multiple Accounts on the Installed Products Page	New topic.
		Use Case: Terminating an Asset with Ramp Lines	Updated the topic with Same Day Cancellation support for Billing Preference and added more use cases.
		Use Case: Early Renewal of Ramped Bundle Assets with Options	New topic.
		Use Case: Cancelled Asset Options Fetch Original Asset Start and End Dates During Renewal	New topic.
		Use Case: A Product with Parent and Child Price Lists	New topic.
		Defining the Benefits of a Promotion	Updated the topic with information about **For Every X Criteria**, **Benefit Product**, and **Benefit Type** sub-sections.
		Enabling Enterprise as a QTC Profile Value	Updated the list of operations the Sales rep can perform.

Document	Publication Date	Topic	Description
		Configuring Custom Button to Access Configuration Page Directly	Updated information about service bundles.
		Recommendations to Import Legacy Products into Conga CPQ	New topic.
		Configuring Progress Tracker for Async Operation	Added information about deep clone.
		Advanced Approval Callback Class	New topic.
		Configuring Field Set Settings	Updated the description of **Asset Line Fields For Selected Assets**.
		Recommendations to Improve CPQ Performance	Added more best practices to the Rules section.
		Configuring the Smart Cart	Updated the topic with information about **APTS_CartLineItemsUpdateChunkSize** admin setting.
		Creating Quick Quotes	New topic.
		Configuring Quick Quote Mode	New topic.
		Configuring CSS Override	New topic.
		Creating and Updating Related Line Items	New topic.
		Saving Related Line Items	New topic.

Document	Publication Date	Topic	Description
		Splitting Related Line Items	New topic.
		Retrieving Apex Batch Job Information	New topic.
		CPQ Asset-Based Ordering APIs	Updated the Terminate Assets REST API documentation with information about: • Passing Cart ID in the URL • Removal of the Cart ID parameter from the Request body
		Service CPQ APIs	New topic.
		About the Installed Products Page	New topic but moved content from the "Viewing Assets from an Account" topic.
		Configuring Display Actions Settings	Updated the description of the use case. Removed a note about **Defer Pricing**.
		Configuring Products from the Catalog	Updated the description of the **Update Price** button.
		Configuring Pricing Engine Settings	Updated the description of the **Defer Pricing** setting.

Document	Publication Date	Topic	Description
		Config System Properties	Updated the description of the **Defer Pricing** setting.
		Configuring Push Alerts	New topic.

Summer '21

Document	Publication Date	Topic	Description
Summer '21 Rev E	14 Sep 2021	Disabling Constraint Rule Execution upon Cart Finalization	New topic.
Summer '21 Rev D	01 Sep 2021	Creating Quick Quotes	Removed the content as this feature is no longer supported.
		Configuring Quick Quote Mode	Removed the content as this feature is no longer supported.
Summer '21 Rev C	09 Aug 2021	Registering for Conga Push Upgrade	New topic.
		Installing CPQ Packages	Updated the topic with a tip about Conga Push Upgrade.
		Preparing for Upgrade	Updated the topic with a tip about Conga Push Upgrade.
		Upgrading to CPQ May '22 Release	Updated the topic with a tip about Conga Push Upgrade.

Document	Publication Date	Topic	Description
Summer '21 Rev B	26 Jul 2021	Creating Promotional Banners	Removed the description because the feature is deprecated.
		Defining Multiple Contracts to be Used in a Quote	Updated the description of the "Solution" section.
Summer '21 Rev A	12 Jul 2021	About TurboEngines	New topic.
		About TurboConfig	New topic.
		About TurboPricing	New topic.
		Managing Price Waterfall	New topic.
		Defining Price Points	New topic.
		Configuring a Price Pipeline	New topic.
		Configuring Price Waterfall to a Price Pipeline	New topic.
		Configuring Price Pipeline Ruleset	New topic.
		Creating Price Pipeline Rules	New topic.
		Using Price Waterfall	New topic.
		Viewing the Cart in Grid View	Added a note under the "Support for Deal Guidance" point.
Summer '21	07 Jul 2021	Configuring Installed Products Settings	Updated the description of the **Editable Fields for Cancelled Lines** setting.

Document	Publication Date	Topic	Description
		Configuring Display Columns Settings	Updated the description of the **Display Type** (Asset Termination) and **Field Name** fields.
		Running Maintenance Jobs	Added the "To specify the batch size for the Category Maintenance batch job" section.
		Configuring Cart Activity History	New topic.
		Creating Visibility Rules	Updated the topic with information about Product Attribute Value.
		Tax Callback Class	New topic.
		Configuring Custom Settings for Different Flows	New topic.
		Creating Products	Added information about **APTS_DisableExpandInCart** .
		Applying the Default Pricing to Products	Updated the description of the **Auto Cascade Selling Term** setting.
		Creating Price Rules	Updated the description of the **Adjustment Applies To** drop-down.
		Creating Price Matrices	Updated the description of the **Matrix Type** field with a note.

Document	Publication Date	Topic	Description
		Configuring Bundle Expansion on Cart Page	New topic.
		Excluding Optional Product in Min/Max Criteria	Updated the steps to configure **Exclude Optional Products**.
		Configuring Config Page Settings	Added the setting **Exclude Optional Products**.
		Config System Properties	Added the setting **Exclude Optional Products**.
		Config Custom Classes	Updated **Formula Callback Class** and **Metadata Callback Class**.
		About Custom Setting Order of Precedence	New topic.
		Recommendation Callback Class	New topic.
		Lookup Field Settings	Updated the description of the **Enable Quick View** setting.
		Formula Callback Class	New topic.
		Metadata Callback Class	New topic.
		Defining the Benefits of a Promotion	Updated the topic with information about the support for different charge types in Buy X Get Y promotions.

Document	Publication Date	Topic	Description
		Managing CPQ Formula Fields	Updated the topic with an example for formula field.
		Creating Price Matrices	Updated the description of the **Adjustment Amount Source** field.
		Use Case: Using a CPQ Formula Field in Price Matrix	New topic.
		About Permission Sets	New topic.
		Managing Translation of Fields and Values in Different Language	New topic.
		Translating Standard and Custom Field Labels	New topic.
		Translation of Field Labels	New topic.
		Translating and Overriding Managed Packaged Field Label	New topic.
		Translating CPQ Record Values	New topic.
		Creating Custom Buttons for Different Flows	Updated the topic description.
		Managing Guided Selling	Removed all descriptions because the feature has been deprecated.
		Creating Custom Fields in the Search Attribute Value Object	Removed the chapter.

Document	Publication Date	Topic	Description
		Creating a Sample Flow Using Flow Designer	Removed the chapter.
		Creating a Guided Selling Visualforce Page	Removed the chapter.
		Defining Guided Selling Rules	Removed the chapter.
		Profile Settings and Security	Removed the chapter. For information about permission sets, refer to About Permission Sets.
		Finalizing the Shopping Cart	Added information about *Enterprise* profile.
		Configuring Progress Tracker for Async Operation	New topic. New feature.
		Associating Options to a Bundle	Added a note about option groups.
		Creating Option Groups	Added information about the distinction between *Shared* and *Standalone* option groups.
		Configuring Custom Fields on Price List Related Pages	New topic.
		Configuring the Tooltips	New topic.
		Navigating the Product Page	Updated the topic with information about tooltip.

Document	Publication Date	Topic	Description
		Using Proposal Document Generation	Updated the topic with information about the following: • Contact name in the email. • Updates to **Presented Date** on the Quote Detail page.
		About Quantity and Selling Term Calculation Using Default Pricing Settings	New topic.
		Use Case: Auto Cascade Quantity and Auto Cascade Selling Term for Standalone Products	New topic.
		Use Case: Auto Cascade Quantity and Auto Cascade Selling Term for Bundle Products	New topic.
		Managing Mass Update of Product Fields	Updated the topic with information about rollup line items.
		Terminating an Asset	Updated the topic with information about editing custom asset line item fields on the Confirm Termination page.
		Viewing the Cart in Grid View	Updated information about bundle expansion and custom fields on Summary Group.

Document	Publication Date	Topic	Description
		About Quote Lifecycle Collaboration	Updated information about the disabled **Submit for Approval** button.
		Working on the Configuration Request	Updated the description with information about the inability to submit cart for approval.
		Merging the Configurations in the Parent Cart	Updated the description with information about the inability to submit cart for approval.
		Use Case: Cancelling Options in a Bundle Asset with Ramp Lines	New topic.
		Use Case: Terminating an Asset with Ramp Lines	New topic.
		Creating Quotes from Opportunities	Updated the following information: • The description of the **Single Transaction Adjustment** checkbox. • Added information about the **Enterprise** quote profile.
		Managing Assets in the Contract Flow	Updated the topic with information about the **Single Transaction Adjustment** checkbox.
		Configuring Products from the Catalog	Updated the optional products section.

Document	Publication Date	Topic	Description
		Finalizing the Shopping Cart	Added information about the **Enterprise** quote profile.
		Using Guided Selling	Removed the content as the feature has been deprecated.
		Async APIs Using Batch Apex	New topic.
		Repricing the Cart	New topic.
		Copying Product Configuration to the Proposal	New topic.
		Asset Web Services	New topic.
		Change Assets	New topic.
		Renew Assets	New topic.
		Get a List of Products to be Swapped with Assets	New topic.
		Swap Assets	New topic.
		Terminate Assets	New topic.
		Get a Count of Assets	New topic.
		Get a List of Assets	New topic.
		Retrieving Products Auto-included to the Cart	New topic.
		Dissociating Attribute Group and Products	New topic.

Document	Publication Date	Topic	Description
		Associating Attribute Group and Products	New topic.
		Adding and Removing Attributes from an Attribute Group	New topic.
		Creating New Attribute Group	New topic.
		Retrieving Products Included by Auto-inclusion Constraint Rule	New topic.
		Creating a Cart from a Quote	Updated the code sample.
		CPQ Asset-Based Ordering APIs	Updated the description of the Change Asset API with information about cascading attribute changes to sub-bundle child options.

Spring '21

Document	Topic	Description
Spring '21 Rev L	Disabling Constraint Rule Execution upon Cart Finalization	New topic.
Spring '21 Rev K	Configuring Quick Quote Mode	Removed the content as this feature is no longer supported.
	Creating Quick Quotes	Removed the content as this feature is no longer supported.
Spring '21 Rev J	Configuring Field Set Settings	Updated the description of the **Asset Line Fields For Selected Assets** setting.

Document	Topic	Description
Spring '21 Rev I	Using Guided Selling	Removed the content as the feature has been deprecated.
	Performing the Post-Upgrade Tasks	Updated the description with a note about API names in Config LineItem Custom Fields.
Spring '21 Rev H	Registering for Conga Push Upgrade	New topic.
	Installing CPQ Packages	Updated the topic with a tip about Conga Push Upgrade.
	Preparing for Upgrade	Updated the topic with a tip about Conga Push Upgrade.
	Upgrading to CPQ May '22 Release	Updated the topic with a tip about Conga Push Upgrade.
Spring '21 Rev G	Configuring Flow Settings	Updated the topic with a note that special characters are not supported in the flow name.
	Configuring Post-Installation Settings	Updated the topic with a note that special characters are not supported in the flow name.
	Creating Products	Updated the topic with a note that special characters are not supported in the product name.
	Creating Promotional Banners	Removed the description because the feature is deprecated.
	Managing Guided Selling	Removed the description because the feature is deprecated.
	Defining Multiple Contracts to be Used in a Quote	Updated the description of the "Solution" section.
	Creating Custom Fields in the Search Attribute Value Object	Removed the chapter.

Document	Topic	Description
	Creating a Sample Flow Using Flow Designer	Removed the chapter.
	Creating a Guided Selling Visualforce Page	Removed the chapter.
	Defining Guided Selling Rules	Removed the chapter.
Spring '21 Rev F	Configuring Promotions on Benefit Products in Buy X Get Y	New topic.
Spring '21 Rev E	Defining Batch Size to Process Line Items with Promotions	Updated the topic with information about **Buy X Get Y** type promotion.
	Applying Promotions on Line Items in the Cart	Updated the topic with information about *APTS_EnableMultiBenefitItems* Admin Setting.
Spring '21 Rev D	Excluding Optional Product in Min/Max Criteria	New topic.
	Configuring Products from the Catalog	Updated the topic with information about option group Min/Max validation for optional bundles.
Spring '21 Rev C	Creating Visibility Rules	Updated the topic with information about Classic UI.
	Configuring Products from the Catalog	Updated the topic with information about option group Min/Max validation for optional bundles.
Spring '21 Rev B	Defining Batch Size to Process Line Items with Promotions	New topic.
	Disabling Time Adjustment in Price List Item Fields	New topic.

Document	Topic	Description
	Pricing Callback Class	Updated the topic. Add information about **isCartTotalingDisabled**.
	Configuring Products from the Catalog	Updated the topic with information about Min/Max criteria of option group for optional options.
	Repricing the Cart	Updated the sample code.
Spring '21 Rev A	Enabling Enterprise as a QTC Profile Value	Updated the topic with information about a cloning proposal.
	Tasks Available in the Smart Cart Flow	Updated the topic with information about cloning proposals that are not finalized.
	Async APIs Using Batch Apex	New topic.
	Repricing the Cart	New topic.
	Adding Custom Bundles	Updated the sample code.
	Copying Product Configuration to the Proposal	New topic.

Spring '21	Configuring Custom Settings	• Updated the description of the **Relate Action Criteria Fields** setting under Installed Products Settings.
		• Updated the description of the **Service Price Distribution Method** setting under Config System Properties.
		• Updated the description of the following setting under Proposal System Properties:
		  • **Enable Fast Doc Gen**
		  • **Enable File**
		• Added **CR Maintenance Governor Limits Threshold** in Config System Properties.
		• Added **Enable Quick View** in Lookup Field Settings.

Document	Topic	Description
	Configuring Document Generation for Quote	Renamed the topic from "**Configuring Large Document Generation for Quote**". Updated the topic with information about how to configure Doc Gen.
	Configuring Field Set Settings	Updated the description of the **Related Line Item Fields** and **Asset Line Fields For Selected Assets** settings.
	About Product Attribute Value Extensions	Updated the description with information about Document Generation.
	Option Filter Callback Class	Updated information about hiding option groups.
	Defining the Benefits of a Promotion	Updated the description of the **Product** field.
	Configuring Application Management Settings	Added a note for information about Custom Settings Maintenance Job.
	Configuring Option Net Adjustment Rollup to Bundle	New topic.
	Enabling Enterprise as a QTC Profile Value	New topic.
	Configuring Service CPQ	New topic.
	Configuring Service Pricing	New topic. Moved from the *Conga CPQ User Guide*.
	Defining Service Line Split Criteria	Moved this topic from the "Configuring Asset-Based Ordering" section to the "Configuring Service CPQ" chapter.
	Synchronizing Related Line Items with Opportunity Line Items	Moved this topic from the "Managing Quotes or Proposals" chapter to the "Configuring Service CPQ" chapter.

Document	Topic	Description
	Running the Constraint Rule Maintenance Job	Updated the topic. Added information about the threshold for Constraint Rule Maintenance job.
	Lifecycle Callback Class	New topic.
	Adjustment Spread Callback Class	New topic.
	Deal Optimizer Callback Class	New topic.
	Cart Approval Callback Class	New topic.
	Associating Options to a Bundle	Updated the description of the **Auto-Update Quantity** field.
	Configuring Exclusion Rules	Added information about **Match in Service Assets**.
	Creating Constraint Rule Conditions	Added a use case.
	Creating Quotes from Opportunities	Updated the topic information about the **Single Transaction Adjustment** checkbox.
	Applying Promotions on Line Items in the Cart	Updated the topic information about applying *Buy X Get Y* promotions on multiple line items of Y products, and with information about line items with $0 price.
	Searching Assets	Updated topic with a note about pagination.
	Pricing Products	Updated the topic with information about the **Submit For Pricing (Async)** button.
	Tasks Available in the Smart Cart Flow	Updated the topic added information about **Sync with Opportunity** and Doc Gen.
	About Smart Cart	Updated the limitation of the Smart Cart.

Document	Topic	Description
	Configuring Products from the Catalog	Updated the section about **is Optional** checkbox.
	Managing Services	Updated the topic to remove the ambiguous content.
	Associating Services with Assets	New topic
	Working with the Service Catalog	New topic.
	Configuring Service Products	Updated the topic with missing information.
	Working with the Service Cart	Renamed from "Viewing the Service Cart" and updated the topic with missing information.
	Viewing Service Products on the Installed Products page	Removed the topic.
	Service Pricing	Moved this topic to the *Conga CPQ Administrator Guide*. See Configuring Service Pricing.
	Splitting the Related Line Items from Services	New topic. Moved the content from the "Service Pricing" topic to this topic.
	Use Case: Bundle to Bundle	New topic.
	Use Case: Bundle to Components	New topic.
	Use Case: Bundle to Bundle and Components	New topic.
	Using Proposal Document Generation	Updated the topic.

Document	Topic	Description
	Viewing the Cart in Grid View	Updated the information about lookup field columns.
	CPQ Admin Web Service	New topic.
	Creating New Products	New topic.
	Creating New Categories	New topic.
	Associating Products to Category	New topic.
	Building Hierarchy of Products	New topic.
	Checking if Product Exists in Hierarchy	New topic.
	Removing Products from Hierarchy	New topic.
	Retrieving the List Option Group	New topic.
	Retrieving of Product Hierarchy	New topic.
	Retrieves List of Parent Product	New topic.
	Retrieves List Product IDs	New topic.
	Retrieves List of Category IDs	New topic.
	Constraint Web Service 2	New topic.
	Associating Constraint Rule to Products Added to the Cart	New topic.

Document	Topic	Description
	Troubleshooting CPQ SOAP APIs	New topic.
	Remote CPQ Admin Controller	New topic.
	Searching Custom Fields in Objects	New topic.
	Retrieving Incentives on the Cart	Added information about the new parameter **BundleLineItemID**.
	Creating Coupons for Incentives	New topic.
	CPQ Asset-Based Ordering APIs	Updated the request and response parameters of the Change Asset API.

Winter '20

Document	Topic	Description
Winter '20 Rev I	Disabling Constraint Rule Execution upon Cart Finalization	New topic.
Winter '20 Rev H	Configuring Quick Quote Mode	Removed the content as this feature is no longer supported.
	Creating Quick Quotes	Removed the content as this feature is no longer supported.
Winter '20 Rev G	Registering for Conga Push Upgrade	New topic.
	Installing CPQ Packages	Updated the topic with a tip about Conga Push Upgrade.
	Preparing for Upgrade	Updated the topic with a tip about Conga Push Upgrade.

Document	Topic	Description
	Upgrading to CPQ May '22 Release	Updated the topic with a tip about Conga Push Upgrade.
Winter '20 Rev F	About Product Attribute Value Extensions	Updated the topic with information about using Product Attribute Value Extension Objects in Search Filter(CPQ).
	Creating Promotional Banners	Removed the description because the feature is deprecated.
	Defining Multiple Contracts to be Used in a Quote	Updated the description of the "Solution" section.
Winter '20 Rev E	Enabling Enterprise as a QTC Profile Value	Updated the topic with information about a cloning proposal.
	Configuring Custom Button to Access Configuration Page Directly	Updated the topic with information about multi-currency org.
	Tasks Available in the Smart Cart Flow	Updated the topic with information about cloning proposals that are not finalized.
Winter '20 Rev D	Enabling Optimized Attribute Retrieval	Updated the topic.
	Creating Option Groups	Updated the topic.
	Associating Options to a Bundle	Updated the description of the field **Allow Cloning**.
	Adding Miscellaneous Items	Added a note about the inability to modify miscellaneous line items once add to the cart.
	Assigning the Configuration to the Collaborator or Queue	Added a note about miscellaneous line items not being supported in Quote Collaboration.

Document	Topic	Description
Winter '20 Rev C	Configuring Option Net Adjustment Rollup to Bundle	New topic.
	Configuring Renewal Settings	Added the "To synchronize renewal lines to renewal opportunities" section.
	Pricing Products	Updated the topic with information about Bundle and Summary Group Net Adjustment %.
	CPQ Admin Web Service	New topic.
	Creating New Products	New topic.
	Creating New Categories	New topic.
	Associating Products to Category	New topic.
	Building Hierarchy of Products	New topic.
	Checking if Product Exists in Hierarchy	New topic.
	Removing Products from Hierarchy	New topic.
	Retrieving the List Option Group	New topic.
	Retrieving of Product Hierarchy	New topic.
	Retrieves List of Parent Product	New topic.
	Retrieves List Product IDs	New topic.
	Retrieves List of Category IDs	New topic.
	Constraint Web Service 2	New topic.

Document	Topic	Description
	Associating Constraint Rule to Products Added to the Cart	New topic.
	Troubleshooting CPQ SOAP APIs	New topic.
	Remote CPQ Admin Controller	New topic.
	Searching Custom Fields in Objects	New topic.
Winter '20 Rev B	Configuring Lookup Field Settings	Added the following fields and description: • Lookup Record Limit • Junction Object Name • Junction Field Name • Junction Default Flag • Filter Criteria 2
	Creating Price List Items	Updated the description of the **Price Type** and **Price Method** fields.
	Adding Additional Details to a Price List Item	Updated the description of the **Price Type** and **Price Method** fields.
	Creating Quotes from Opportunities	Updated the topic. Added information about **QTC Profile**.
	Finalizing Products	Updated the topic. Added information about **QTC Profile**.
	Creating Favorite Configuration	Updated the sample code.
Winter '20 Rev A	Configuring Custom Settings	Updated the description of the **Enable File** setting under Config System Properties and Proposal System Properties.
	Configuring Display Actions Settings	Added a note about the **Update Price** button.

Document	Topic	Description
	Creating Quotes	Updated a note with information about fields **Grand Total**, **Net Amount**, and **Unit Price**.
	Marking Quotes as Primary	Updated the topic with steps on making a quote as primary.
	Tasks Available in the Smart Cart Flow	Updated description of **Clone(With Line Items)**.
	Updating Category Hierarchy for multiple Categories	Updated the sample code.
	Updating Category Hierarchy for a single Category	Updated the sample code.
	CPQ Asset-Based Ordering APIs	Updated the Authentication section.
Winter '20	Configuring Installed Products Settings	Added the **Cotermination Preferences During Renewal** and **Default Renewal Cotermination Option** settings in RENEWAL section.
	Configuring Post-Installation Settings	Updated the topic with information about running Custom Setting Maintenance Job.

Document	Topic	Description
	Configuring Custom Settings	• Added the **Alert Asset Related To Renewal Cart**, **Cotermination Preferences During Renewal** and **Default Renewal Cotermination Option** settings in Installed Product Settings. • Added a note for information about Custom Settings Maintenance Job. • Updated the description of **Constraint Rules Execution Mode**. • Updated the description of **Enable Field Expressions** under Config System Properties.
	Creating Custom Buttons for Different Flows	Added a new parameter **styleSheetURL**.
	Creating Products	Updated the topic with the **Renewal Lead Time** field description.
	Configuring Config Page Settings	Updated the topic description and added information about support for *Sidebar Action* in the **Display As** field.
	Configuring Display Actions Settings	Updated the topic description.
	Creating Custom Buttons for Different Flows	Replaced **isCartApprovalDisabled** with **useAdvancedApproval** and isPricingGuidanceDisabled with **useDealOptimizer**, updated descriptions.
	Defining the Scope of a Promotion	Updated the description of the **Product** field with the **Include Bundle Options (AND)** option under *Buy X Get X*.

Document	Topic	Description
	Defining the Benefits of a Promotion	Removed the **In** option on the Benefits section for Buy X Get Y promotions.
	Defining the Limits of a Promotion	Updated the topic with information about applying incentive limits across quotes for the same account.
	Running Maintenance Jobs	Updated the topic. Added information about Custom Settings Maintenance batch job.
	Action Params Callback Class	New topic.
	Action Invoker Callback Class	New topic.
	Creating Custom Product Attribute Value Extensions	Updated the topic description.
	About Product Attribute Value Extensions	New topic.
	Configuring Conga Sign for eSignature	New topic. New feature.
	Configuring Configuration Engine Settings	Added the **Constraint Rule Execution Mode** field
	Configuring Config Page Settings	Added the following fields: • **Max Inline Option Attributes** • **Enable Option Page Search**
	Cascading Shared Attribute Values	New topic. New feature.
	Classifying Attributes	New topic. New feature.
	Enabling Optimized Attribute Retrieval	New topic. New feature.

Document	Topic	Description
	Associating Options to a Bundle	Updated the information about the lock icon for the quantity of options.
	Action Callback Class	New topic.
	IActionCallback2 Interface	New topic.
	IActionCallback3 Interface	New topic.
	Using Proposal Document Generation	Updated the topic with a note about warning message on email pop-up.
	Configuring Application Management Settings	Updated the topic with a note about disabled values.
	Creating Price Ramps for a Product	Added a note about the increased scalability of price ramps.
	Applying Promotions on Line Items in the Cart	Updated the topic with information about display of promotions on the cart.
	Viewing Assets from an Account	Added a note about the display of the **Use the Proposal End Date** option on the Confirm Renewal intermediate page.
	About Asset-Based Ordering Flows	Added a note about the display of the **Use the Proposal End Date** option on the Confirm Renewal intermediate page.
	Renewing Assets Manually	Updated the topic with information about enhanced Confirm Renewal intermediate page and handling manual renewal of assets that are associated with a system-generated renewal quote.

Document	Topic	Description
	Renewing Assets in Auto Renewal Mode	Updated the topic with information about a warning during renewal.
	Renewing Assets in OnDemand Renewal Mode	Updated the topic with information about a warning during renewal and renewing assets in OnDemand Renewal mode based on the Renewal Lead Time specified in products.
	Managing Asset Increment with Coterminate Lines	Added a note about the display of the **Use the Proposal End Date** option on the Confirm Renewal intermediate page.
	Renewing Assets Manually in the Contract Flow	Updated the topic with information about enhanced Confirm Renewal intermediate page and handling manual renewal of assets that are associated with a system-generated renewal agreement.
	Renewing Assets in Auto Renewal Mode in the Contract Flow	Updated the topic with information about a warning during renewal.
	Renewing Assets in OnDemand Renewal Mode in the Contract Flow	Updated the topic with information about a warning during renewal and renewing assets in OnDemand Renewal mode based on the Renewal Lead Time specified in products.
	About Smart Cart	Updated the topic with the following information: • Removed multiple adjustments as limitations • Added synchronizing proposal line item with opportunity as a limitation.

Document	Topic	Description
	Tasks Available in the Smart Cart Flow	Updated the topic. Added information about multiple adjustments and revalidation.
	Sending Documents for eSignature Using Conga Sign	New topic. New feature.
	Managing Mass Update of Product Fields	Updated the topic with information about Advanced Filter.
	Applying Advanced Filters on the Cart	Updated the topic with information about the **Select All Qualified Lines** checkbox.
	CPQ Asset-Based Ordering APIs	New topic.
	REST API Guide	New topic.

Summer '20

Document	Topic	Description
Summer 2020 Rev A	Enabling PDF Security for Generated Proposal Documents	Updated the code for Admin Entry in the topic.
	Configuring Installed Products Settings	Updated the description of the **Asset Termination Fields** setting with a warning.
	Configuring Custom Settings	Updated the description of **Asset Termination Fields** setting with a warning.
	Configuring Display Columns Settings	Updated the description of **Display Type** and **Field Name** settings.
	Setting Up Service CPQ	New topic.
	Working with the Mini-Cart	New topic.

Document	Topic	Description
	Configuring a Product from the Catalog	Updated the topic with information about Mini-cart, and **Edit** and **View** buttons. .
	Using Proposal Document Generation	Updated the topic with information about **Rename** and **Delete** buttons.
	Viewing the Cart in Grid View	Updated the topic with information about **Edit** and **View** buttons.
	Presenting a Quote/Proposal	Updated the topic with a note about contact details.
Summer 2020	Configuring Custom Button to Access Configuration Page Directly	New topic. New feature.
	Configuring Display Columns Settings	Updated the topic with a note for the **Display Type** Asset Termination.
	Configuring Large Document Generation for Quote	New topic. New feature.
	Managing Constraint Rules	Updated the topic with information about support for *Show Messages* and *Prompt* in **Match in Assets** Match Conditions in CSCR. Updated the topic description.

Document	Topic	Description

	Configuring Custom Settings	• Added **Update View Category Batch Size** in Config System Properties.
		• Added the following settings in Proposal System Properties:
		  • **Large Doc Threshold**
		  • **Large Doc Process Batch Size**

	Managing Currency Conversion	New topic. New feature.
	Enabling Price Ramps	Updated the topic with information about ramps inheriting *Cumulative Range - Line Item* price matrices from the price list item.
	Enabling Auto Ramp Creation	Updated the topic description.
	Creating Custom Buttons for Different Flows	Updated the topic with information about the **useAdvancedCurrency** parameter.
	Defining the Criteria of a Promotion	Updated the description of the **Value** field.
	Configuring the Apply Promotion Window	New topic.
	Configuring Display Actions Settings	Updated the topic with information about configuring action buttons on the Configuration page.
	Creating Product Attribute Rules	Updated the topic with information about support for null value in scope.

Document	Topic	Description
	Pricing Extension Callback Class for Smart Cart	Updated the topic description.
	Creating Custom Product Attribute Value Extensions	Updated the topic description.
	Enabling Quote Collaboration in the Org	Updated the topic with information about Field Det Settings for Quote Collaboration
	Running Maintenance Jobs	Updated the topic with information about Category and Bundle Maintenance.
	Managing Products	Updated the topic description.
	Creating Products	Removed information about **Has Defaults**.
	Managing Categories and Hierarchies	Updated the topic description. Updated the description of **Offering and Option Group**.
	Reordering Root Categories on the Catalog Page	Updated the topic description and added a note about editing layout.
	Running a Category Maintenance Job	Updated the topic description and moved the Category Maintenance section to Running Maintenance Jobs topic.
	Creating Option Groups	Updated the description of **Offering and Option Group**.
	Creating Price Lists	Updated the topic description.
	Associate a Price List with Categories	Updated the topic description.

Document	Topic	Description
	Adding Additional Details to a Price List Item	Updated the topic description.
	Applying the Default Pricing to Products	Updated the topic description.
	Creating Price Rulesets	Updated the topic description.
	Creating Price Rules	Updated the topic description.
	Creating Price Dimensions	Updated the topic description.
	Managing Contract Pricing	Updated the topic description.
	Defining Contracts to be Used in a Quote	Updated the topic description.
	Defining Multiple Contracts to be Used in a Quote	New topic.
	Configuring Contract Pricing through Price Agreements	New topic.
	Enabling Multiple Adjustments at the Line-Item Level	Updated the topic description.
	Managing Product Groups	Updated the topic description.
	About Search Filters (CPQ)	Updated the topic description.
	Configuring Search Filter (CPQ)	Updated the topic with a reference image.
	Managing Product Rules	Updated the topic description.
	Creating Constraint Rules	Updated the topic with information about **Bundle Context**.
	Creating Constraint Rule Conditions	Updated the topic description.

Document	Topic	Description
	Creating Constraint Rule Actions	Updated the topic description.
	Scenarios for Using Constraint Rules	Updated the topic description.
	Creating Product Attribute Values	Updated the topic description.
	Creating Attribute Value Matrices	Updated the topic description.
	Types of Promotions	Updated the topic description.
	Defining User Profiles and Permissions	Updated the topic description.
	Configuring Incentive Records	Updated the topic description.
	Creating Price Dimensions for a Promotion	Updated the topic description.
	Creating Child Filters	Updated the topic description.
	Configuring Data Rollups for Promotions	Updated the topic description.
	Defining the Scope of a Promotion	Updated the topic with information about including bundle with options in the Scope of promotions.
	Defining the Criteria of a Promotion	Updated the topic description.
	Defining the Benefits of a Promotion	Updated the topic description.
	Generating Coupons for a Promotion	Updated the topic description.
	About Integration of Assets with Contracts	Updated the topic description.

Document	Topic	Description
	Configuring Auto Renewal of Assets	Updated the topic description.
	Renewing an Asset in Auto Renewal Mode	Updated the topic description.
	Renewing an Asset in OnDemand Renewal Mode	Updated the topic description.
	Using the Search Filter	Removed the topic.
	Configuring Conditional Inclusion of Options	Removed the topic.
	Applying Promotions on the Products in the Shopping Cart	Updated the topic with information about combining promotions.
	Managing Cart Views	Updated the topic description.
	Splitting an Asset	Updated the topic with minor changes on the Cart page for split-swap bundles and options.
	Swapping an Asset	Updated the topic with a minor change about the user seeing the Confirm Installed Products Swap page or the Confirm Swap pop-up.
	Terminating an Asset	Updated the topic with a minor change about read-only fields on the Cart page.
	Searching Assets	Updated the topic with search optimization for the Installed Products page.

Document	Topic	Description
	Generating a Quote/Proposal	Updated the topic with information about the **Submit** button for Large Document Generation for a quote.
	Tasks Available in Smart Cart Flow	Updated the description of the tasks to clone(with Line Items) and create an agreement with line items.
	Quote Lifecycle Collaboration	Updated the description.
	Assigning the configuration to the Collaborator or Queue	Updated the description.
	Working on the Configuration Request	Updated the description.
	Merging the Configurations in the Parent Cart	Updated the description.
	Creating a Price Ramp for a Product	Updated the topic with information about creating price ramps for products with Related Pricing.
	Creating Quote/Proposals from Opportunities	Updated the description.
	Contract Pricing	Updated the description.
	Renewing an Asset	Updated the description.
	Renewing Assets Manually	Updated the description.
	Use Case: Different Renewal Dates	New topic.
	Renewing Assets Automatically	New topic.

Document	Topic	Description
	Renewing an Asset in Auto Renewal Mode	New topic.
	Use Case: Grouping of Assets without Renewal Group Fields During Renewal	
	Use Case: Grouping of Assets with Renewal Group Fields During Renewal	New topic.
	Renewing an Asset in OnDemand Renewal Mode	New topic.
	Use Case for Renewals without Grouping	New topic.
	Terminating an Asset	Updated the description.
	Use Case: Terminating an Asset When Billing Management is Installed	New topic.
	Use Case: Same Day Cancellation	New topic.
	Use Case: Terminating a Renewed Asset	New topic.
	Managing Assets through Contracts	Updated the description.
	Managing Assets in the Contract + Proposal Flow	New topic.
	Changing Assets in the Contract + Proposal Flow	New topic.
	Renewing Assets in the Contract + Proposal Flow	New topic.

Document	Topic	Description
	Swapping Assets in the Contract + Proposal Flow	New topic.
	Terminating Assets in the Contract + Proposal Flow	New topic.
	Managing Assets in the Contract Flow	New topic.
	Changing Assets in the Contract Flow	New topic.
	Renewing Assets in the Contract Flow	New topic.
	Renewing Assets Manually in the Contract Flow	New topic.
	Renewing Assets Automatically in the Contract Flow	New topic.
	Renewing an Asset in Auto Renewal Mode in the Contract Flow	New topic.
	Renewing an Asset in OnDemand Renewal Mode in the Contract Flow	New topic.
	Swapping Assets in the Contract Flow	New topic.
	Terminating Assets in the Contract Flow	New topic.
	Adding a Miscellaneous Item to the Cart	New topic. New API.
	Abandoning a Configuration	New topic. New API.

Document	Topic	Description
	Finalizing Configuration in Smart Cart	New topic. New API.
	Cloning Primary Line Items	New topic. New API.
	Creating Bundle Line Items	New topic. New API.
	Copying Attachments	New topic. New API.
	Creating a Task After Proposal is Presented	New topic. New API.
	Retrieving Name of the First Template	New topic. New API.
	Retrieving Templates in a Proposal	New topic. New API.
	Deleting an Email Template	New topic. New API.
	Updating a Proposal	Updated sample code.
	Retrieving Incentives on the Cart	Updated the topic with information about new parameters.
	Retrieving Incentives Applied on the Cart	New topic. New API.
	Setting Incentives for Cart	New topic. New API.

Spring '20

Document	Topic	Description
Spring 2020 Rev B	Integrating Conga with External Systems	New topic for external integrations using the Conga CPQ SOAP API.

Document	Topic	Description
	Generic API Operations in Salesforce (SOAP)	New topic describing examples of calling generic Salesforce SOAP methods for creating and retrieving common Salesforce objects referenced by Conga APIs.
	CPQ Web Service(Apttus_CPQApi)	Renamed the topic from "CPQ Web Service". Updated the topic description.
	Creating a Cart from a Quote	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Retrieving Categories for a Price List	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Retrieving Products and List Prices for a Price List	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Retrieving Products and List Prices for a Price List and Category	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Retrieving Products and List Prices For a Price List and Search Text	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Retrieving Option Groups, Options, and List Prices for a Price List Product	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Adding Products to a Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.

Document	Topic	Description
	Adding a Bundle to a Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Adding a Custom Bundle	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Adding Options to a Bundle	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Cloning Bundle Line Items on the Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Cloning Line Items on the Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Cloning Option Line Items on the Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Comparing Products	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Associating Constraint Rules to a Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.

Document	Topic	Description
	Applying Constraint Rules to Deleted Products	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Adding Price Ramps to a Cart (CPQ Web Service)	Updated the topic with API signature.
	Computing the Net Price for a Bundle	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Updating Price For A Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Price Breakup for a Cart or Specific Line Item	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Removing a Bundle from a Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Removing Multiple Bundles from a Cart	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Removing Line Items from the Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.

Document	Topic	Description
	Removing Options from a Bundle	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Retrieving Constraint Rules Results	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Retrieving Incentives on the Cart	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Retrieving Asset Line Items	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Abandoning a Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Finalizing a Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Synchronizing a Cart	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Updating Quote Terms	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Computing Shipping for Cart Line Items	Updated the topic with API signature.

Document	Topic	Description
	Computing Taxes for Cart Line Items	Updated the topic with API signature.
	CPQ Web Service (Apttus_Config2)	New topic.
	Creating Product Line Items	Moved the topic from "CPQ Web Service". Updated the topic with API signature.
	Registering Split Cart Parameters	Moved the topic from "CPQ Web Service". Updated topic to identify API as a global method and added signatures.
	Finalizing the Configuration Version	Moved the topic from "CPQ Web Service". Updated topic to include API signature, SOAP Response/ Request XML, and prerequisites for API integration.
	Finalizing the Configuration Version with Synchronization Tasks	Moved the topic from "CPQ Web Service". Updated the topic with API signature.
	Batch Update Service	Updated the topic description.
	Updating Category Hierarchy for a single Category	Renamed the topic from "Updating a Specific Category".
	Updating Category Hierarchy for multiple Categories	Renamed the topic from "Updating Multiple Categories".
	Quote/Proposal Config Web Service	Updated the topic description.
	Accepting a Quote	Updated topic to include API signature, and SOAP Response/ Request XML for API integration.

Document	Topic	Description
	Quote Collaboration Service	Renamed the topic from "Collaboration Structure". Updated the topic description.
	Adding Products to a Collaboration Request	Updated the topic with API signature.
	Favorite Configuration Global Service	New topic.
	Creating Favorite Configuration	Renamed the topic from "FavoriteConfiguration Web service". Updated the topic with API signature.
	Asset Service	Updated the topic description.
	Fetching Asset Line Items	Updated topic to identify API as a global method and added signatures.
	Getting a count of Asset Line Items	Updated topic to identify API as a global method and added signatures.
	Swapping Assets	Updated topic to identify API as a global method and added signatures.
	Getting a list of Products to be swapped with Assets	Updated topic to identify API as a global method and added signatures.
	Terminating Assets	Updated topic to identify API as a global method and added signatures.
	Renewing Assets	Updated topic to identify API as a global method and added signatures.

Document	Topic	Description
	Incrementing Assets	Updated the topic with API signature.
	Changing Assets	Updated topic to identify API as a global method and added signatures.
	Merging Duplicate Assets	Updated topic to identify API as a global method and added signatures.
	Batch Job Service	Updated the topic description.
	Splitting a Smart Cart	Updated topic to identify API as a global method and added signatures.
	Updating Price for the Smart Cart	Updated topic to identify API as a global method and added signatures.
	Updating Price For A Cart	Updated topic to identify API as a global method and added signatures.
	Proposal Web Service	Updated the topic description.
	Retrieving Proposal Field Values from Account	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Retrieving Proposal Field Values from Opportunity	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.

Document	Topic	Description
	Retrieving Field Values from Proposal	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Creating Proposal Line Items from Opportunity	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Updating a Proposal	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Cloning Email Template	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Cloning Document Templates	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
	Getting the Proposal Document Output Format	Updated topic to include API signature, SOAP Response/Request XML, and prerequisites for API integration.
Spring 2020 Rev A	Configuring Custom Settings	Updated the description of the **Document Naming Convention** setting under Proposal System Properties.
	Synchronizing Related Line Items with Opportunity Line Items	Updated the topic with a note that this feature is applicable only for Service CPQ.

Document	Topic	Description
Spring 2020	Configuring Installed Products Settings	Updated the description of the **Editable Fields for Cancelled Lines**, **Asset Termination Fields**, and **Renewal Business Object Type** settings. Added the **Account Hierarchy Batch Size** setting. Removed the *Change:MassEdit* option from the **Submenu Actions** setting.
	Creating Price List Items	Updated the note in the description of the **Charge Type Criteria** setting.
	Configuring Pricing Engine Settings	Updated the topic with information about turning on or of defer pricing at a flow level and selectively skipping the totaling of pricing of products on the Cart.

Document	Topic	Description

	Configuring Custom Settings	Updated the topic with information about the following custom settings:
		• Added the following custom setting:
		  • **Base Private Properties**
		• Updated the Config System Properties > **CSS Override** setting description.
		• Added the following settings in Config System Properties:
		  • **Product Sorting Fields**
		  • **Refine Search Checkbox Display Limit**
		• Added a note for the **Enable Price Matrix Audit Trail** setting in Config System Properties.
		• Added the following settings in Proposal System Properties:
		  • **Document Naming Convention**
		• Added the following field in Config Custom Classes:
		  • **Revalidation Callback Class**
		• Added the **Account Hierarchy Batch Size** field in Installed Products Settings. Removed the **Mass Edit Action Criteria Fields** setting. Removed the *Change:MassEdit* option from the **Submenu Actions** setting.
		• Updated the description of the **Editable Fields for Cancelled Lines** and **Renewal Business Object Type** fields in Installed Products Settings.

	CPQ Package Objects	Added the object of the **Apttus Base Library** package.

Document	Topic	Description
	Installing CPQ Packages	Updated the topic with information about the **Apttus Base Library** package.
	Upgrading CPQ	Updated the topic with information about upgrading from the previous two releases to the current release.
	Configuring Product Sort	New topic. New feature
	Creating Price Ruleset	Updated the topic with information about defining currency for the price ruleset.
	Creating Price Rule	Updated the topic with information about setting the UOM and effective dates for price rule entry.
	Creating Price Dimensions	Updated the topic with more information about **Context Type** and **Type**.
	Applying Miscellaneous Pricing to a Product	Updated the description of the **Disable Asset Integration** setting with a note.
	Managing CPQ Formula Fields	New topic. New feature
	Creating Constraint Rule Actions	Updated the description of the **Match in Assets** Match Conditions.
	Managing Constraint Rules	Updated the topic. Added information about support for the **Match in Assets** Match Conditions in inclusion rules.

Document	Topic	Description
	Opportunity Renewal - OnDemand	Moved the section about **Asset Renewal Callback Class** to the new topic "Asset Renewal Custom Callback Class".
	Creating Custom Buttons for Different Flows	Renamed the topic from "Creating a custom button for the new user interface". Updated the topic with information about the **asyncFinalize** parameter. Added the **isCartTotalingDisabled**, **isCartApprovalDisabled**, **isPricingGuidanceDisabled**, and **&deferPricingUntilCart** parameters.
	Visibility Rules through Custom Classes	Renamed the topic to "Product Filter Callback Class" and moved the new topic under "Customizing CPQ Using Callbacks".
	Using Asset Line Item Callback Class	Renamed the topic to "Asset Line Item Callback Class" and moved the new topic under "Customizing CPQ Using Callbacks".
	Customizing CPQ Using Callbacks	New topic. Added description for Callback Class
	Display Action Callback Class	Updated the description.
	Option Filter Callback Class	Updated the description.
	Asset Line Item Callback Class	Updated the topic description. Renamed this topic from "Using Asset Line Item Callback Class" to "Asset Line Item Callback Class".
	Asset Renewal Custom Callback Class	New topic.

Document	Topic	Description
	Product Attribute Callback Class	New topic.
	Related Pricing Callback Class	New topic.
	Validation Callback Class	New topic.
	Product Filter Callback Class	Updated the topic description. Renamed this topic from "Visibility Rules through Custom Classes" to "Product Filter Callback Class".
	Adjustment Line Item Callback	Updated the description.
	Pricing Callback Class	Updated the description of the topic.
	Revalidation Callback Class	New topic. New feature.
	Managing Promotions	Updated the definition and scope of Promotions Management.
	Defining the Benefits of a Promotion	Updated the topic with information about using operators to define multiple benefit products.
	Creating Approval Workflows for Promotions	Updated the topic with information about selectively skipping approval of the Cart.
	Managing Categories and Hierarchies	Updated this topic with information about category types.
	Creating Categories	Updated the description of the topic. Shifted the details about category types to the topic "Managing Categories and Hierarchies". And removed information about the **Manage Options Groups** menu.

Document	Topic	Description
	Creating Option Groups	Updated the navigation path to the **Manage Option Groups** menu.
	Associating Options to an Option Group	Updated the navigation path to the **Manage Option Groups** menu.
	About Manage Field Expressions Page	New topic. Added information about a dedicated page to manage field expression.
	About Manage Rollups Page	New topic. Added information about a dedicated page to manage rollups.
	About Manage Rules Page	New topic. Added information about a dedicated page to manage rules.
	Associating Products to a Product Group	Updated the topic description with information about searching product associated with the product group.
	Logging in to CPQ	Updated the topic with information about **My Domain**.
	Configuring Display Columns Settings	Added the **Asset Termination** Display Type and a note for the **Mass Edit (Assets)** Display Type.
	Enabling Price Breakup for Products	Updated the topic with information about displaying custom fields in Price Breakup pop-up.
	Adding Custom Columns to the Mass Edit (Assets) Window	Removed the topic because it is an invalid feature.

Document	Topic	Description
	Configuring Renewal Settings	Updated the topic for clarity and added the "To schedule a batch job for creating renewal opportunity" section. Updated the description of the **Renewal Business Object Type** setting.
	Searching a Product from the Catalog	Updated the topic with information about sorting the products on the Catalog.
	Configuring a Product from the Catalog	Updated the topic with information about **is Optional** checkbox.
	Opening Cart in Read-only mode	New topic. Added information about Cart in read-only mode
	Use Case: Renewing Merged Assets with the Same Purchase Identifier	New topic. New feature for this release.
	Managing Assets	Updated the topic with information about sorting assets on the Installed Products page.
	Renewing an Asset	Updated the topic with notes about **Farthest End Date**. Updated the topic with an enhanced note on renewable products.
	Merging Assets	Updated the topic with notes about **Effective Date** field and **Related Line Items** popup. Also updated that the end date is not editable for cancelled lines.
	Swapping an Asset	Updated the topic with notes about selectively displaying fields while swapping assets.

Document	Topic	Description
	Terminating an Asset	Updated the topic with information about editing a custom currency field.
	Impact of Asset actions on Billing Schedules	Added the **Proration Period Treatment** field and updated the definition of **Billing Day of the Month** and **Billing Cycle Start** fields.
	Performing Mass Update for Assets	Removed the topic because it is an invalid feature.
	Viewing Service Products on the Installed Products page	Updated the note about validating service assets based on criteria.
	Adding Products	Updated the description.
	About Product Configuration	New topic. Added information about product configuration status.
	Using Deal Guidance	Updated the topic with information about enhanced support for Deal Guidance and selectively skipping Deal Guidance.
	Smart Cart	Updated the description. Added Multiple Adjustments as a limitation.
	Tasks Available in Smart Cart Flow	Updated the description. Removed Multiple Adjustments as an available functionality.
	Logging on to CPQ	Updated the description.
	Cloning Bundle Line Items on the Cart	Updated the topic with a note about using the **Line Number** field instead of the **Primary Line Number** field.

Document	Topic	Description
	Batch Update Service	Renamed from the topic "BatchUpdate Web Service".
	Proposal Web Service	New topic.
	Adding a Custom Bundle	New topic. New API
	Retrieving Proposal Field Values from Account	New topic. New API
	Retrieving Proposal Field Values from Opportunity	New topic. New API
	Creating Proposal Line Items from Opportunity	New topic. New API
	Updating Proposal	New topic. New API
	Removing Options from a Bundle	New topic. New API
	Retrieving Field Values from Proposal	New topic. New API
	Getting the Proposal Document Output Format	New topic. New API
	Cloning Email Template	New topic. New API
	Cloning Document Templates	New topic. New API
	Creating Product Line Items	New topic. New API
	Finalizing the Configuration Version	New topic. New API

Document	Topic	Description
	Finalizing the Configuration Version with Synchronization Tasks	New topic. New API
	Merging Assets	New topic. New API

Winter '19

Document	Topic	Description

Winter 2019	Configuring Custom Settings	Updated the topic with information about the following custom settings:
		• Updated Lookup Field Settings section. Added information about Product Configuration object in **Object Name** field.
		• Updated the Config System Properties > **Selling Term Calculation Method** setting description.
		• Updated the Config System Properties > **Constraint Rules Execution Mode** setting description.
		• Added the Installed Products Settings > **Merge Action Criteria Fields** setting details.
		• Added the following settings in Config System Properties:
		  • **Auto Execute Pre-Pricing Step**
		  • **Auto Execute Post-Pricing Step**
		  • **Custom Pre-Pricing Fields**
		  • **Split Cart Threshold**
		  • **Split Cart Criteria Fields**
		  • **Defer Pricing**
		  • **Pricing Batch Size**
		  • **Pricing Profile**
		  • **Auto Sync With Opportunity**

Document	Topic	Description
	Configuring Installed Products Settings	Updated the topic with information about the **Submenu Actions**, **Merge Action Criteria Fields**, **End Date preferences for Cotermination**, and **Hide Asset Actions** settings.
	Configuring Custom Settings	Updated the topic with information about the following custom settings: • Config Asset Pricing Criteria Fields • Config Asset Pricing Default • Config Custom Classes • Config Custom Display Actions • Config Custom Display Columns • Config Data Cache • Config Field Set • Config Flow • Config Select Options Settings • Config Select Products Settings • Config User Preferences
	Using Asset Line Item Callback Class	Updated the topic with a new sample code.
	Configuring Smart Cart	New topic. New feature for this release.
	Use Case for Configuring Smart Cart	New topic. New feature for this release.
	Pricing Extension Callback Class For Smart Cart	New topic. New feature for this release
	Creating Price Dimensions	Updated the topic with more options in **Context Type**.
	Synchronizing Related Line Items with Opportunity Line Items	New topic. New feature for this release.
	Option Filter Callback	Updated the topic description.

Document	Topic	Description
	CPQ Package Objects	Update the description of the following objects: • **Billing Plan** • **Billing Plan Item** • **Billing Plan Template** • **Billing Plan Template Item** • **Billing Preference**
	Defining the Scope of a Promotion	Updated the topic with information about adding product groups to promotions scope.
	Enabling Price Breakup	Updated the topic with information about displaying custom fields in Price Breakup.
	Creating Price Rules Creating Price Matrices Configuring Related Pricing	Updated the topics with information about formula fields when defining price or adjustment.
	Contract Pricing	Updated the topics with information about generating contract price list items for bundles with the same option.
	Running a Category Maintenance Job	Updated the topic with details on incremental updates for category views.
	Configuring Asset Visibility on the Installed Products Page	Updated the topic with a note about critical attributes.
	Configuring the Custom Attributes Page	Updated the topic with enhanced description.
	Configuring Cart Page Settings	Updated the topic to add the details for **CART LINE COLUMNS**.
	Associating an Attribute Group to a Product	Updated the topic with the description of the **Is Primary** setting.

Document	Topic	Description
	Migration Script for Cart Views	Updated the topic. Updated the Migration Script
	Constraint Rules	Updated the topic. Added a note about Service CPQ.
	Applying Advanced Filters	New topic. New feature for this release.
	Adding Miscellaneous Items	New topic. Added more information about adding miscellaneous items.
	Managing Assets	Updated the topic with the sorting method of the assets.
	Merging Assets	New topic. New feature for this release.
	Service Pricing	Updated the description of the topic.
	Smart Cart	New topic. New feature for this release.
	Tasks Available in Smart Car Flow	New topic. New feature for this release.
	Smart Cart Pricing	New topic. New feature for this release.
	Swapping an Asset	Updated the topic with information about selling term calculation.
	Splitting an Asset	Updated the topic with information about splitting bundle products and the Amended status of standalone and bundle products.
	Terminating an Asset	Updated the topic with information about editable fields on cancelled lines.
	Changing a Configurable Bundle Asset	Updated the topic with the step of clicking the wrench icon.

Document	Topic	Description
	Changing the Attributes of an Asset	Updated the topic with the step of clicking the wrench icon.
	Renewing an Asset	Updated the topic with information about **Renewal Adjustment Type** and **Renewal Adjustment Amount** columns.
	Configuring a Product from the Catalog	Updated the topic. Added a note in the section **To configure product attributes**.
	Pricing Products	Updated the topic. Added a note in the section **To price a product on the shopping cart**.
	Registering Split Cart Parameters	New topic. New API
	Batch Job Service	New topic.
	Splitting a Smart Cart	New topic. New API
	Updating Price for the Smart Cart	New topic. New API

Summer '19

Document	Topic	Description
Summer 2019	Configuring Installed Products Settings	Updated the topic with information about the **ASSET VIEWS** setting.
	System Properties	Added **Rounding Mode** setting.
	Creating Price Matrices	Updated the topic with information about getting the price matrix entry and the **Cumulative Range - Line Item** setting.
	Configuring Currency Rounding	New topic. New feature for this release.

Document	Topic	Description
	Defining the Scope of a Promotion	Updated the topic with information about the **Product** field.
	Running a Bundle Maintenance Batch	Updated the topic with information about using Batch Mode maintenance for complex bundles.
	Config System Properties	Updated the topic with the following new settings. • **Adhoc/Product Totaling Hierarchy** • **Enable Price Matrix Audit Trail** • **FavoriteFilters** • **Groupby Fields** • **Service Line Split Criteria** • **Totaling Group Type** • **Update View Product Batch Size** • **Update View Use DML Limit**
	Defining Split Criteria	Updated the topic with information about Related Line Item and Asset Line Item objects.
	Installed Product Settings	Updated the topic with the following new settings. • **Apply Adj To Current Contract Term** • **Base Price Defaulting Method For Renewal**
	Client Side Constraint Rules	Removed the topic. The relevant information is mentioned with the appropriate topic.
	Constraint Rules	Updated the topic with information about Client-Side Constraint Rules

Document	Topic	Description
	Create Constraint Rules	Updated the topic with information about effects of deactivated or deleted constraint rules on the finalized cart.
	Constraint Rule Actions	Updated the topic. Modified description of **Min/Max Match Rule**.
	Setting up an Inclusion Rule	Updated the topic with information about **Added By** and **Added By Rule Info** fields.
	Setting up an Exclusion Rule	Updated the topic with information about **Hide** functionality.
	Applying Promotions on the Products in the Shopping Cart	Updated the topic with information about reapplying a promotion.
	Contract Pricing	Updated the topic with information about adjustments applied as part of contract price and fallback method when contract list price is unavailable.
	Managing Views for Assets Grid	Updated the topic with information about views on the Assets grid created by the administrator.
	Renewing an Asset	Updated the topic with information about renewing an asset using current contract value.
	Terminating an Asset	Updated the topic with information about validating the termination date and displaying contract term billing details.
	Favorite Configurations on the Cart	Updated the topic with information about categorizing favorite configuration using **Favorite Filters**.

Document	Topic	Description
	Searching a Product from the Catalog	Updated the topic with information about counter on mini-cart and recommendation icons.

Spring '19

Document	Topic	Description
Spring 2019 Rev A	System Properties	Added **Enable Custom Rounding** setting.
	Terminating an Asset	Updated the topic with information about showing net price of the current year asset on the termination page.
Spring 2019	Configuring Config Page Settings	Added **Enable Option Page Search** setting.
	Configuring Installed Products Settings	Updated the topic with information about **Change:Split** sub-menu action and **Split Asset Action** settings.
	Associating Options to Bundles	Updated the topic with information about the message displayed to run Bundle Maintenance batch job.
	Client Side Constraint Rules (CSCR)	Updated the topic with a note about support for **Match in Location**.
	Defining a Cost Model	Updated the topic with information about the **Hide Child Cost Types in Price Waterfall** option.

Document	Topic	Description
	Enabling Quote Collaboration in your org	Updated the topic with information about revoked requirement **Modify All** setting.
	Defining the Scope of a Promotion	Updated the topic with information about searching a product by its name or code.
	Data Flow	Updated the topic with information about using the **Auto Renewal** flag correctly.
	Manage Product Page	Added the topic with information about Manage Product page in CPQ Admin.
	Cloning Products	Added the topic with information about cloning product in CPQ Admin.
	Searching a Product from the Catalog	Updated the topic with information about actions products inheriting location of the condition products.
	Configuring a Product from the Catalog	Updated the topic with information and a note about search functionality for options and attribute on the configuration page.
	Favorite Configurations on the Cart	Updated the topic with information about saving constraint rule in favorite configuration.
	Working on the Configuration Request	Updated the topic with the information about support for replacement and deletion of collaborated product

Document	Topic	Description
	Merging the Configurations in the Parent Cart	Updated the topic with information about support for automatic child cart updates.
	Billing Schedules for Incremented Assets	Updated the "Increment Asset with Coterminate Lines" topic with a use case for billing schedules.
	Splitting an Asset	New topic. New feature for this release.
	Terminating an Asset	Updated the topic with information about showing the original asset start date on the intermediate page UI.
	Assigning the configuration to the Collaborator or Queue	Updated the topic with information about adding products to a collaboration request.
	Accepting a Quote	New topic. New API
	Adding Products to a Collaboration Request	New topic. New global method.

Winter '18

Document	Topic	Description
Winter 2018 Rev B	Applying and Removing Promotions or Coupons on Line Items in Cart	Moved this topic from this guide to *CPQ on Salesforce Winter 2018 User Guide* and merged content in the "Applying Promotions on the Products in the Shopping Cart" topic.

Document	Topic	Description
	Creating the Price Ramp on the Cart Page	Updated the topic. • Removed a limitation about Cart Grid UI. • Removed the note about support for IE 11.
	Billing for Assets	Updated the topic with information about billing.
	Configuring Document Generation Preview Setup in Internet Explorer	Updated the topic with a note that the feature is no longer supported.
	Applying Promotions on the Products in the Shopping Cart	Moved the "Applying and Removing Promotions or Coupons on Line Items in Cart" topic from *CPQ on Salesforce Winter 2018 Administrator Guide* to this guide and merged the contents.
	Creating the Price Ramp on the Configuration Page	This topic is removed because the feature is no longer available.
Winter 2018 Rev A	Config System Properties	The Config System Properties related to large cart were removed because the large cart feature was not part of the Winter 2018 release.
	Splitting Large Cart	This topic is removed because the Large Cart feature was not part of the Winter 2018 release.
Winter 2018	Config System Properties	Added a new field. **Enable Notification Feed**
	Defining Split Criteria	New topic. New feature for this release.
	Multiple Adjustments at the Line Item	Updated the topic with details on Adjustment Types.

Document	Topic	Description
	Creating Constraint Rule Actions	Updated the topic with information about the value *Hide* in the **Action Intent** table.
	Client Side Constraint Rules (CSCR) Enhancements	Updated the topic with information that **Match in Cart Options** is now supported.
	Pricing Callback Classes	Updated the topic with the method **onPriceItemSet()**.
	Enabling Quote Collaboration in your org	New topic. New feature for this release.
	Defining Permissions for Favorite Feature	New topic. New feature for this release.
	Associating Options to Bundles	Updated the topic with a note in the topic about shared option groups.
	Structure Section	Updated the topic with a new feature for CPQ Admin.
	Lookup Field Setting	Updated the topic with information about the **Filter Criteria 2** field.
	Installed Products Settings	Updated the topic with information about End Date Preference Cotermination and sub-menu actions.
	Applying and Removing Promotions or Coupons on Line Items in Cart	Updated the topic with information about applying promotions. (Moved this topic to *CPQ on Salesforce Winter 2018 User Guide* in Winter 2018 Rev B)
	Generating Coupons	Updated the topic with a note about generating coupons.

Document	Topic	Description
	Adding Custom Columns to the Mass Edit (Assets) Window	New topic. New feature for this release. Removed the topic in Spring 2020 because it is an invalid feature.
	Configuring Service Products	Updated the topic with information about cloning service line.
	Pricing Products	Updated the topic with new options in Adjustment Type and a note to the **% Discount Off Lift** option.
	Assigning the configuration to the Collaborator or Queue	Updated the topic with notes about access permissions being considered in the Collaboration Feature.
	Bucket Adjustments	Updated the topic with a note about bucket adjustment applicable on the preceding bucket.
	Service Pricing	Updated the topic with information about splitting the Related Line Items from Service Line.
	Configuring a Product from the Catalog	Updated the topic with information about selecting products from a prompt.
	Favorite Configurations on the Cart	Updated the topic with information about access permissions.
	Managing Views for Assets Grid	New topic. New feature for this release.
	Performing Mass Update for Assets	New topic. New feature for this release. Removed the topic in Spring 2020 because it is an invalid feature.

Document	Topic	Description
	Increment Asset with Coterminate Lines	New topic. New feature for this release.
	Creating Quote/Proposals from Opportunities	Updated the topic with information about purchase identifier.

Summer '18

Document	Topic	Description
Summer 2018 Rev A	Currency Field Precision	Information about the application of Currency Field Precision is added as a note.
	Expand Bundles in Cart	Information about Pagination feature is added as a note
Summer 2018	Creating a Price List	New configuration for creating a price list is added: **Disable Based on Currency Adjustment**
	Applying Miscellaneous Pricing to a Product	Disable Cost Model field is added.
	Creating Price List Items	New grouping type such as Summary Group is added.
	Adjustment Line Item Callback	Adjustment Line Item callback is added.
	Types of Promotions	Sales promotion details are added.
	Defining the Criteria of a Promotion	Contains operator is added.
	Use Case: Configuring a Promotion for the Total Price on the Cart	Use case for an order level promotion is added.

Document	Topic	Description
	Managing Coupons	The Assigned To filter is added. Information about storage of coupon code is added as a note.
	Extending the Lead Time Functionality	Sample callback classes to extend the lead time functionality is added.
	Bucket Adjustments	Included a note about the creation of an adjustment line.
	Viewing Installed Products Page	Changes to the Installed Products User Interface is added.
	Searching Assets	Enhancements to the Search functionality are added.
	Viewing Service Products on the Installed Products page	The Relate and relate Component flows are added.
	Service Pricing	Service Configuration and pricing information are added.

CPQ for Administrators

This section explains how to configure Conga Configuration & Pricing. This section describes the administrative tasks to be performed to enable the features in CPQ and covers the most common use cases for CPQ administration.

Topic	Description
What's Covered	This section walks the administrator through the process of CPQ administration. It provides conceptual information, step-by-step instructions, and use cases for CPQ administrative tasks. This section describes the workflows for configuring products, defining the price for configured products, defining product and pricing rules and so on.
Primary Audience	Administrator
IT Environment	Refer to the latest *Conga CPQ Release Notes* for information on System Requirements and Supported Platforms.
Updates	For a comprehensive list of updates to this section for each release, see the What's New in CPQ Documentation topic.
Other Resources	Refer to *Conga CPQ Release Notes* for information on system requirements and supported platforms, new features and enhancements, resolved issues, and known issues for a specific release.

This section describes the following tasks:

- Create Products
- Manage Bundles
- Create Categories and Hierarchy
- Manage Attributes
- Manage Rules
- Create Price Lists
- Manage Pricing Rules
- Configure Quote/Proposal
- Manage Promotions
- Manage Assets and Asset Line Item

Before using CPQ, you must be familiar with the following:

- Basic Salesforce administration
- Salesforce and Conga terms and definitions

Select one of the following topics for more information:

Getting Started with CPQ

Select one of the following topics for more information about Conga CPQ (formerly known as Apttus CPQ)

- Enabling CPQ for Partner Community Users

Before You Install CPQ

This section explains the tasks to be completed before you install CPQ.

To enable Salesforce CRM content

Before you can install CPQ module packages, you must ensure that Salesforce CRM Content is enabled in your org. You must be logged into Salesforce.com.

1. Go to **Setup** > **Customize** > **Salesforce Files** > **Settings** > **Salesforce CRM Content** and click **Edit**.
2. Select the **Enable Salesforce CRM Content** check box.
3. Click **Save**. Salesforce CRM Content is enabled.

Now, you can start installing CPQ packages.

Installing CPQ Packages

Multiple packages must be installed to implement the complete CPQ solution. Packages for CPQ must be installed in the order indicated in the table in this section. You begin with the Conga base packages and then install the integration packages that enable the various products to function together.

You may not need to install all of the packages. For example, if your implementation does not use Advanced Approvals you do not need to install Conga Approvals or Conga Quote Approvals. Refer to the table in this section to learn which packages are required for standalone CPQ and which packages are required only when you are using other Conga products.

> Conga recommends downloading and upgrading Conga packages in a Salesforce sandbox **before** installing them in your production environment. For information on installing and upgrading in a sandbox, please contact Conga Support before you install any packages.

The Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, It ensures all the Conga published managed packages are on the latest versions for the registered orgs. To register your org for push upgrade, see Registering for Conga Push Upgrade.

Install the packages in the following order.

The Conga Base Library provides a shared library of reusable components that other Conga packages may use. All CPQ and Contract Management customers must install this package in order to use Conga CPQ and Conga CLM products. In addition, beginning in the Spring '20 release, Conga is adding a User Experience Analysis component that collects product usage data. No action is required on your part and there is no impact to any existing product features, system functionality, or configured workflows. No personally identifiable information (PII), nor any other proprietary, confidential, or sensitive data will be collected. For more information, see CPQ on Salesforce Spring 2020 Release Notes.

Order	Package	Install Center tab to access the package	Required?	Install this package only if you also have...
1	Conga Base Library	Contract Management	Y	
2	Conga Contract Lifecycle Management	Contract Management	Y	
3	Conga Quote Management	CPQ	Y	
4	Conga Configuration & Pricing	CPQ	Y	
5	Conga Quote Configuration Integration	Integrations	Y	
6	Conga CPQ Api	CPQ	Y	

Order	Package	Install Center tab to access the package	Required?	Install this package only if you also have...
7	Conga CLM Configuration Integration	Integrations	N	Contract Management
8	Conga Quote CLM Integration	Integrations	N	Contract Management
9	Conga Quote Asset Integration	Integrations	Y	Renewals
10	Conga CPQ Setup	CPQ	Y	

If you are using Advanced Approvals for Quote, you must install the following packages:

Order	Package	Install Center tab to access the package	Required?
1	Conga Approvals	Advanced Approvals	N
2	Conga CPQ Approvals	Advanced Approvals	N
3	Conga Quote Approvals	Advanced Approvals	N

You can additionally install the following packages if you are using the Deal Maximizer:

Order	Package	Install Center tab to access the package	Required?
1	Conga Deal Maximizer Setup	Deal Maximizer	N
2	Conga Deal Maximizer	Deal Maximizer	N
3	Conga CPQ Maximizer	Deal Maximizer	N

If you are using Adobe Sign to electronically sign any type of document associated with quotes, you must install the following packages:

Order	Package	Install Center tab to access the package	Required
1	Adobe Sign	NA	Y
2	Conga Quote EchoSign Integration	Echosign Integration	Y

ⓘ You must have Conga-provided login credentials to the Conga Community Portal to be able to download packages.
If, during installation, you see an "Approve Third-Party Access" prompt, please select the **Yes, grant access to these third-party web sites** check box and click **Continue**.

To install the CPQ managed packages

1. Go to the **Resources** > **Install Center** tab on the Conga Community Portal.
2. In **My Packages** navigation link, click **CPQ**. From the **VERSION** drop-down, select the version that you want to install.
3. Click **Install Now**.
4. Select the environment in which you want to install the packages. Click **Install in Production** to install the packages in your production environment. Click **Install in Sandbox** to install the packages in your sandbox.
5. In the Salesforce login screen, enter your login credentials and click **Log In**.
6. On the Upgrade page, enter the password provided by Conga.
7. Select the profile for which you want to install the package. Conga recommends that you select **Install for All Users**.
8. If you want to **Install for Specific Profiles**, you must define the access level for all profiles. Select from one of the following options.
    - **No Access** - This is the default setting. Apply this access level to disable all object permissions.
    - **Full Access** - Apply this access level to assign users permissions to Read, Create, Edit, Delete, View All, and Modify All for all objects in the CPQ package.
    - **Pricing Administrator** - Apply this access level to assign users permissions to Read, Create, Edit, Delete, View All, and Modify All for all pricing-related objects in the CPQ package.

> If a permission is not enabled for a profile, you can apply Conga-provided permission sets such as the **Apttus CPQ Admin** Permission Set and **Apttus CPQ User** Permission Set to enhance the access levels of the user profile. To assign a permission set, select it from the **Available Permission Sets** box and click **Add**. To remove **a permission set assignment**, select it from the **Enabled Permission Sets** box and click **Remove**. For more information, see About Permission Sets.

9. Click **Set**.
10. Click **Upgrade**.

A message is displayed indicating the installation is underway. Once installed, repeat this procedure for each of the packages.

Registering for Conga Push Upgrade

You can register your org for Conga Upgrade Program to auto-upgrade Conga managed packages. Conga Upgrade Program supports upgrading from the Spring '18 release or later to the latest major release (*n*) or previous two major releases (*n* – 1 or *n* – 2).

To register your org with Conga Upgrade Program for auto-upgrade

This section provides a high-level flow of the org registration process and early feedback programs. Refer to *Conga Upgrade Program documentation* for a comprehensive overview of features.

1. Launch the Conga Upgrade Program's registration page.
2. Select your Salesforce org type (**Production** or **Sandbox**). The Salesforce login page is displayed.
3. Enter your credentials and click **Login**. You are redirected to the connected app called **Conga Push Upgrade CA**, which prompts you to allow access to the org information.
4. Click **Allow**. The org registration form is displayed.
5. Click **Set Package Access Control** to set the accessibility for Conga packages to be upgraded with Conga Upgrade Program.
6. Click **Submit**.
   The **Early Feedback** pop-up is displayed. In addition, Conga sends an email

notification to the system administrator on successful registration with a list of packages (including dependent packages, if any) handled in the upgrade process.

> ⚠ The early feedback program executes a test suite to check if the customizations in your org are compatible with the upgrade. For more information, see the Conga Automation Early Feedback program.

7. Click **Try Now**. A **Conga Automation Dashboard** is displayed with a **Quick Guide** pop-up. Click **Yes** to go through the quick tour of the dashboard.
8. Click **Click to see the test t**o view a list of tests to be executed for your org**,** A **List of Tests** pop-up is displayed.
9. Enter your Email ID and Organization Name. You can enter multiple email ids.
10. Click **Trigger Welcome Email**. Conga sends a welcome email to the email ids entered in the previous step with a list of tests performed in the early feedback program.
11. Click **Proceed**. A **Conga Services Test Automation Execution Page** is displayed.
12. Click the checkbox next to the Job ID and click **Begin Test Execution** to run the predefined automated test suite against your org.

For the latest information on Conga Upgrade Program, see Conga Upgrade Program documentation.

Logging in to CPQ

Log in to your Salesforce.com org to access CPQ.

> ⓘ Do not use the Back button on your browser when using CPQ.

Before you log in to CPQ, make sure you meet the following criteria.

- You have installed all of the required CPQ module packages.
- You have administrative privileges.
- You have login credentials provided by Conga.

To log in to CPQ

1. Go to http://www.salesforce.com/.
   Or
   If your organization is using a sandbox or test environment to access CPQ (for example, if you are doing user acceptance testing), go to http://test.salesforce.com/ instead.
2. From the toolbar at the top of the page, click **Login**. The login page opens.

3. Enter your user name and password, and click **Log in**.
4. Navigate to **Apttus Product Setup** or **Apttus Pricing Setup** to access the Product or Pricing setup tabs, respectively.
    - In Salesforce Classic: Click the App Menu (

).
    - In Salesforce Lightning Experience: Click the App Launcher ().

Now you are ready to start configuring your products and pricing in CPQ.

> For using Lightning Experience in your Salesforce org, you must enable My Domain in your organization. To know how to enable My Domain in your org, refer to the Salesforce Help page My Domain.

Navigating the CPQ Admin User Interface

As an admin, you can use the CPQ Admin console to manage products and its related components such as options, attributes, and categories. The admin console helps you to create a product, associate the product with the categories, price lists, and price list items on a single page. You can define various settings in the Application Management section that are required to use CPQ.

> Alternatively, you can also use the classic Salesforce UI to create a product, associate the product with the categories, price lists, and price list items.

You can navigate to **Apttus CPQ Admin** app, and click **CPQ Admin** tab to access the admin console. The homepage of the CPQ Admin user interface consists of the **Products**, **Catalog**, and **Pricing** tabs on the primary toolbar and the **search** and the **filter** options on the secondary toolbar. You can configure the fields that are displayed on this page. For more information, see Configuring the CPQ Admin Landing Page Fields and Configuring the Product Creation Screen Fields.

A description of each of these user interface elements is provided as follows:

- The **Products**, **Catalog**, and **Pricing** tabs enable you to add and manage products, product associations, and configure pricing. After you have selected any feature from the primary toolbar, you see a secondary toolbar that enables you to configure subsequent flows.
- The **Products** homepage consists of a list of products configured for the organization. The following tasks are available on the secondary toolbar.

    - Search filters enable you to define search criterion to search products. For more information about searching configured products, see Configuring Product Visibility.
    - Click **New Product.** A product creation page is displayed. For more information about creating products, see Creating Products.
    - From the menu icon () next to the **New Products** button, click **New from Clone** to create a new product by cloning an existing product configuration. For more information about cloning a product, see Cloning Products.
    - You also search for the product on the homepage. See Navigating the Product Page for more information.
- The **Catalog** homepage consists of a list of categories.
    - Click field headers to sort the list on the Categories homepage to sort the column.
    - Enter the name of the product that you want to search in the search window. The category that matches the search is displayed.
    - Click **New Category** to navigate to the category creation page. For more information about creating categories, see Creating Categories.
- The **Pricing** homepage consists a list of price lists.
    - Click field headers to sort the list on the Pricing homepage to sort the column.
    - Enter the name of the price list that you want to search in the search window. The price list that matches the search is displayed.
    - Click New PriceList to navigate to the price list creation page. For more information about creating price lists, see Creating Price Lists.
- Click **Showing *<number>* Records** to select the number of products or category records that you want to display in the homepage. Click arrows **< >** to move to the next page.
- The **More** icon () on the primary toolbar enables you to execute maintenance jobs and configure various Application Management settings required to use CPQ.

Configuring the CPQ Admin Landing Page Fields

1. Navigate to **Setup > Customize > Products > Field Sets > Product List Fields.** This field set displays the fields on the landing page of CPQ Admin.
2. Click **Edit** for **Product List Fields**
3. Drag and drop the required fields in the **In the Field Set**.
4. Click **Save**.

Click **New Product** to create a new product configuration.

You can configure the fields that are displayed on the Product Creation page as well.

Configuring the Product Creation Screen Fields

1. Navigate to **Setup > Customize > Products > Field Sets > Product Detail Fields.** This field set displays the fields on the product creation screen of CPQ Admin.
2. Click **Edit** for **Product Detail Fields.**
3. Drag and drop the required fields in the **In the Field Set** column.
4. Click **Save**.

When you click **New Product**, you land on the Product Creation screen.

Once you are done, you can click **Save Product**. Click **Cancel** to roll back the product creation and go back to the previous page.

There are multiple tabs for each product that you can use to associate the categories, create attributes, price list items, and more. As you navigate from one tab to the other, CPQ saves the data that you enter in the previous screen.

Configuring the Tooltips

You can display additional details of product records on the different pages of CPQ Admin. When you hover the cursor on the product name, a tooltip appears that displays more details about the product. By default, you can see the **Product Code**, **Product Family**, and **Product Description** fields on the tooltip. You can configure the tooltip to display different fields.

The tooltips are displayed on the following pages:

- Product Details page
- Product Groups Details page
- Price List Items Tab

To configure tooltips

1. Navigate to **Setup > Customize > Products > Field Sets**. This field set displays the fields on the tooltip.
2. Click **Edit** for **Product Compact Fields**.
3. Drag and drop the desired fields in the **In the Field Set** column. You can add a maximum of 3 fields only. You must remove the unwanted fields.
4. Click **Save**.

For more information on the Product Details page, refer to Navigating the Product Page.

Configuring Custom Fields on Price List Related Pages

You can add custom fields to price list related pages through the CPQ Admin UI. You can add and remove the fields displayed on the pages where you can configure and manage pricing records. You can also add custom fields with **Text**, **Number**, **Date**, **Picklist**, and **Lookup** datatype.

To configure the display of the pricing fields

Perform the following steps to configure the fields:

1. Navigate to **Setup > App Setup > Create > Objects**.
2. Selected one of the following objects where you want to add or remove pricing fields. Click **Edit** next to Field Set based on the page you want to configure.

Object	Field Sets	Description
Price List	Price List List Fields	Configure this field set to add or remove fields from the Manage Price Lists landing page.
	Price List Detail Fields	Configure this field set to add or remove fields from the Price List Detail page.
Price List Item	Price List Item Detail Fields	Configure this field set to add or remove fields from the Price List Item Detail page.
	Price List Item Product Detail Fields	Configure this field set to add or remove fields from the **Pricing** tab on the Product Detail page.
Price Matrix Entry	Price Matrix Entry Detail Fields	Configure this field set to add or remove fields from the **Matrices** tab page.

> ⓘ You can display fields that store IDs such as **External ID** or **Created By ID**, and fields like **Delete**. However, you cannot save any changes you make to them as they are not editable.

3. Drag and drop the required fields in the **In the Field Set** column.

4. Click **Save**.

For more information on how to configure product pricing, refer to Managing Pricing.

About Permission Sets

CPQ has a standard set of permissions that contain the access rights to fields and objects available in CPQ. These permission sets define whether the User can read, create, edit, and delete the records of an object and its fields. These permissions and the impact they have on the User are described below:

Permission	Description
Read	Allows the User to view the existing records created for the object.
Create	Allows the User to create a record for the object.
Edit	Allows the User to edit the existing record of the object.
Delete	Allows the User to delete a record of the object.
View All	Allows the User to read and create records for an object.
Modify All	Allows the User to edit and delete records for an object.

The following table lists the standard permission sets available in CPQ and their usage.

Permission Sets	Usage
Apttus CPQ Admin	This permission set must be assigned to system administrators.
Apttus CPQ User	This permission set can be assigned to any user who is not a system administrator.

There are different sets of permissions for an administrative role and a regular user role. You must assign the permission sets to the Users accordingly.

To view the permission sets

You can find the standard CPQ permission sets by following the steps below.

1. Go to **Setup** > **Administration Setup** > **Managed Users** > **Permission Sets**.
2. Click **Apttus CPQ Admin** or **Apttus CPQ User**.
3. Click **Object Settings** which contains a list of all the objects with their permissions.
4. Click the object names to view the fields of the objects and their permissions.

> ⚠ If you are using permission sets that are not standard to CPQ, you must update permissions for new fields manually based on **Apttus CPQ Admin** and **Apttus CPQ User** permission sets.

CPQ Package Objects

CPQ is comprised of multiple packages as mentioned in the installation section and this page lists out the objects inside each package.

Conga Base Library

Conga Base Library provides a shared library of reusable components that the other Conga packages can use. The following table list the objects in Conga Base Library.

Object	Purpose of the Object
Apttus UI Component Configuration	Object Records holds stored configurations for configurable higher-order components
Apttus UI Component Provider	The object records store the information about Providers of all types that are implemented by consuming applications
Configuration Action Provider (Deprecated)	Junction Object to Correlate Configurations to Action Provider(s)
Configuration Action Provider 2	Junction Object to Correlate Configurations to Action Provider(s)

Conga Configuration & Pricing

Conga Configuration & Pricing gives you complete visibility and control of the quote-to-order phase of your sales process. Beginning in an opportunity, a user can select products or services, configure and price them for simple or highly complex scenarios, and produce high-quality customer-facing quotes and proposals.

Object	Purpose of the Object
Account Billing Summary	Represents the billing summary for an account.
Account Location	Represents a location for an account.
Adjustment Line Item	Represents an adjustment to the price of a product/ service.
Agreement Price Rule	Represents a special pricing rule associated with an agreement or an asset.
Agreement Price Tier	Represents a single entry in an agreement price tier.
Application Feature	An object to contain features in the Applications, where Applications are Conga products like CPQ, CLM etc.
Application Setting	Object to contain metadata about application settings.
Applied Expression Info	Represents expressions applied to the configuration.
Applied Rule Action Info	Rule actions that are applied to the configuration.
Applied Rule Info	Keeps track of rules applied to a configuration.
Asset Attribute Value	Represents the attribute values for a product class.
Asset Attribute Value History	Represents the attribute values for a product class.
Asset Line Item	Represents a product or service asset line item.
Asset Line Item History	Represents a product or service asset line item history.
Asset Transaction History	Represents a transaction that records the change to an asset.

Object	Purpose of the Object
Asset Usage Price Tier	Represents a single entry in an asset usage price tier.
Asset Usage Price Tier History	Represents a single entry in an asset usage price tier.
Attribute Group Translation	This object holds the translated values of attribute group object fields for all the languages.
Attribute Value Matrix	Represents the information about attribute value compatibilities or incompatibilities.
Attribute Value Matrix Entry	Represents a row (entry) in an attribute value matrix.
Batch Job (CPQ)	Represents an individual batch job.
Batch Job (CPQ) Parameter	Represents a batch job bind parameter.
Billing Plan	Represents a billing plan associated with a product or service that is part of a quote or order.
Billing Plan Item	Represents a billing entry in a billing plan of a quote or order.
Billing Plan Template	Represents the template to create a billing plan for a quote or order.
Billing Plan Template Item	Represents a billing plan template item associated with the quote or order.
Billing Preference	Represents a functional preference for billing purposes of a quote or order
Bundle Component View	Keeps the list references of component products related to the bundle and sub-bundle.
Cart Clone Info	
Category	Represents a category.
Category Hierarchy	Represents a category hierarchy.

Object	Purpose of the Object
CategoryTranslation	Catalog Hierarchy field translations are stored in this object. Each Category Translation object holds a translation for a SFDC supported language for a defined Category Hierarchy Object record.
Charge Group	Represents a group of charge types associated with a product or service.
Charge Group Member	Represents the mapping between a charge group and the charge types associated with the group.
Charge Type	Represents a type of charge for a product or service.
Collaboration Request	This object will persist information for quote collaboration tasks and will be used to relate the parent configurations and its related child configurations.
Constraint Rule	Captures information about configuration constraint rule.
Constraint Rule Action	Captures information about rule actions. Rule actions are applied only when the rule conditions are satisfied.
Constraint Rule Action Expression	
Constraint Rule Condition	Captures information about the condition of a constraint rule. In order for a rule to trigger all or some of the rule conditions should be true and the combining expression should be true.
Cost Adjustment	Represents an adjustment to the cost of a product/ service.
Cost Item	Represents a product entry in a Cost Model.
Cost Line Item	Represents cost associated with base price of line item.
Cost Model	Cost Model represents cost model of a FES or Factory.

Object	Purpose of the Object
Cost Type	Represents cost hierarchy definition.
Custom Message	Used to provide translation for messages used in custom objects.
External Order Adjustment Item	Represents an adjustment to the price of a product/ service.
External Order Summary	Represents an external order summary associated with an account.
External Order Summary Item	Represents a product or service line item associated with an external order summary.
Favorite Configuration	
Feature	Represents a single feature in a feature set.
Feature Set	Represents a set of features shared by products.
Field Expression	Represents a field expression.
Footnote	Represents a single footnote in a document.
Formula Field (CPQ)	Represents a formula field used in price dimensions and rules.
Frequency Conversion Rate	Represents the conversion rates used to convert pricing from one frequency to another.
G/L Account Map Entry	Represents a single entry in a G/L account map rule.
G/L Account Map Rule	Represents a G/L account map rule to determine G/L account numbers for a product or service.
Guided Question	Represents a question in a guided interview.
Guided Search Rule Entry	Represents a single entry in a guided search rule.

Object	Purpose of the Object
Guided Search Rule Filter	Represents a single search filter in a guided search rule entry.
Guided Selling Rule	Represents a rule to search for products based on a guided interview.
Help Doc	Object holds help documentation.
Incentive	Represents an incentive associated with one or more products.
Incentive Adjustment Item	Represents an adjustment to the price of a product/ service.
Incentive Benefit Data	Represents a single benefit data row associated with the incentive benefit.
Incentive Coupon	Represents a coupon associated with an incentive.
Incentive Group	Represents a group of incentives.
Incentive Limit	Represents a single limit associated with an incentive.
Incentive Limit Data	Represents a single limit data row associated with the incentive limit.
Incentive Loyalty Enrollment	Represents a single customer enrollment into an incentive loyalty program.
Line Item	Represents a product or service line item.
Line Item Rollup	Holds rollup summary custom fields.
Loyalty Point Accrual Adjustment Item	Represents loyalty points accrual adjustments for a product/service.
Loyalty Point Accrual Summary	Represents a summary of loyalty points accrued for the account or contact.

Object	Purpose of the Object
Milestone	Represents milestones associated with other business objects.
Milestone Adjustment Item	Represents an adjustment to the price of a product/service.
Milestone Template	Represents a template for creating milestones.
Notification Feed	A generic object to contain Notification Feeds generated by any object for any page.
Order	Represents an order associated with an account.
Order Adjustment Line Item	Represents an adjustment to the price of a product/service.
Order Fulfillment	Represents an order fulfillment associated with an account.
Order Fulfillment Line Item	Represents a fulfillment line item associated with one or more orders.
Order Incentive Benefit Data	Represents a single benefit data row associated with the incentive benefit for the order.
Order Incentive Limit Data	Represents a single limit data row associated with the incentive limit for the order.
Order Line Item	Represents a product or service line item associated with an order.
Order Loyalty Point Accrual Item	Represents loyalty points accrued for a product/service.
Order Pocket Adjustment Line Item	Represents a pocket price adjustment to the price of a product/service.
Order Product Attribute Value	Represents the attribute values for a product class in an order line item.
Order Rollup Data	Represents the dynamic rollup data for use in criteria.

Object	Purpose of the Object
Order Tax Breakup	Represents the tax breakup of a product/service
Order Usage Price Tier	Represents a single entry in an order usage price tier.
Payment Term	Represents a payment term associated with a quote, order or an agreement.
Permission Set Relationship	
Price Breakup	Represents the price breakup of a product/service.
Price Dimension	Represents a price criteria dimension.
Price List	Represents a price list that contains the list of products the organization sells.
Price List Category	Represents the categories associated with a price list.
Price List Item	Represents a product entry in a price list.
Price Matrix	Represents a price matrix associated with a price list entry.
Price Matrix Entry	Represents a single entry in a price matrix.
Price Rule	Represents a single rule in a price ruleset.
Price Rule Entry	Represents a single entry in a price rule.
Price Ruleset	Represents a rule set that contains the list of rules to modify prices of products the organization sells.
Product Attribute	Represents a single attribute in a product attribute group.
Product Attribute Group	Represents a product attribute group that contains attributes shared by products.

Object	Purpose of the Object
Product Attribute Group Member	Represents the association between a product and an attribute group.
Product Attribute Matrix View	Represents a denormalized (materialized) view of product attribute and matrices.
Product Attribute Rule	Represents the rule definition of a product attribute value.
Product Attribute Rule Action	Represents a single action in a product attribute rule.
Product Attribute Rule View	Represents a denormalized (materialized) view of product and attribute rule.
Product Attribute Value	Represents the attribute values for a product class.
Product Classification	Represents a product classification which is a member of the list of products associated with a classification.
Product Configuration	Represents a product configuration.
Product Constraint	Represents a set of constraints a product must satisfy before it can be added to the product configuration.
Product Constraint Entry	Represents a single entry in a product constraint.
Product Constraint View	Creates product and constraint rule view table based on the definition of product scope in constraint rule condition.
Product Default Rule	Represents a rule to default products, quantity, and term.
Product Default Rule Filter	Represents a single filter in a product default rule.
Product Default Value	Represents the default values for a product.
Product Feature	Represents the association between a product and a feature set.

Object	Purpose of the Object
Product Feature Value	Represents the feature values for a product and its associated feature set.
Product Filter View	Represents the filter values for each product in a category.
Product Footnote	Represents the footnote available in a product or option.
Product Group	Represents a group of products.
Product Group Member	Represents a single product in a product group.
Product Hierarchy View	Represents a denormalized (materialized) view of product and classification hierarchies.
Product Information	Represents information associated with a product or a classification.
Product Option Component	Represents a product option component which is a member of the list of options associated with a product option group.
Product Option Group	Represents a product option group which is a member of the list of option groups associated with a product.
Product Option Price	Represents the price for an option or component.
Product Translation	This object holds the translated values of product object fields for all the languages.
Published Favorite	A junction object between Favorite Configurations and Price Lists to support many to many relationships.
Related Account Location	Holds the relationships between account locations.
Related Asset Line Item	Holds the relationships between asset line items.
Related Incentive	Holds the relationships between incentives.

Object	Purpose of the Object
Related Price List Item	Holds the relationships between price list items.
Related Product	Related products are for up-sell, cross-sell or related products (search helper).
Revenue Recognition Policy	Represents a revenue recognition policy associated with a product or service.
Revenue Split Policy	Represents an order revenue split policy associated with a product or service.
Revenue Split Policy Entry	Represents a single entry in a revenue split policy.
Rollup Data	Represents the rollup data for use in criteria.
Saved Search	Represents a saved search criteria.
Search Attribute Value	Holds product search parameters and default values for line items.
Search Filter (CPQ)	Represents a search filter.
Service Location	Represents a service location.
Setting Group	Object to group settings into logical sets.
Smart Search Activity History	Contains a history of the actions occurring during the search process when connecting to the Azure system (activation, deactivation, sync).
Summary Group	Represents a summary of line items.
Tax Breakup	Represents the tax breakup of a product/service.
Tax Certificate	Represents a tax certificate.
Tax Code	Represents a tax code associated with a product or service.

Object	Purpose of the Object
Temp Display Column	Holds information about columns displayed in the cart view.
Temp Filter	Helps in the binding of search filter values.
Temp Incentive Benefit Data	Represents a single benefit data row associated with the incentive benefit for the product configuration.
Temp Incentive Limit Data	Represents a single limit data row associated with the incentive limit for the product configuration.
Temp Object (CPQ)	A container for temporary objects used by the application.
Temp Renew	Temporary object for capturing user input to create asset line item during the amendment/renewal process.
Temp Renew Asset Group	Temporary object to store renew asset group processed by RenewOpportunityJob.
Temp Renew Asset Line Item	Temporary object to store renew asset line item processed by RenewOpportunityJob.
Temp Rollup Data	Represents the dynamic rollup data for use in criteria.
Totaling Group	Represents a totaling group.
Transfer Price	Represents association of source price like factory price with source cost like FES source cost.
Usage Price Tier	Represents a single entry in a usage price tier.

Conga Contract Lifecycle Management

The Conga Contract Lifecycle Management Suite gives you complete visibility and control of your contract process. This suite is easy to deploy and use; you can immediately gain benefits such as process control, high-risk management, and immediate ROI.

Object	Purpose of the Object
Account Hierarchy	This defines a single instance of an account hierarchy relationship. It contains the leaf Id, the parent Id, the root Id, and the level.
Admin	Store admin preferences and metadata.
Agreement	Represents an agreement in the system.
Agreement Action Condition	Represents conditions to enable agreement actions.
Agreement Clause	Represents a clause associated with an agreement.
Agreement Document	Links to agreement documents that are not stored in Conga/Salesforce repository. These could be URLs to files stored on a file server or document management system behind the client's firewall.
Agreement Document Output Format	Holds default output format preferences by user profile and agreement type for use in the agreement document generation process.
Agreement Explorer	A reusable, unique combination of Field Sets and Filter Sets.
Agreement Hierarchy	This defines a single instance of an agreement hierarchy relationship. It contains the leaf Id, the parent Id, the root Id, and the level.
Agreement Line Item	Represents a product or service line item associated with an agreement.
Agreement Lock	Represents a lock on the agreement.
Agreement Protection	Specifies protection settings for agreements generated by the Conga Contract Wizard and maintained by the Conga Contract Author.
Agreement Rule	Specifies a rule used to evaluate various agreement actions such as submit request.
Agreement Rule Condition	Specifies a condition used to evaluate an agreement rule.

Object	Purpose of the Object
Agreement Term Exception	Term Exceptions associated with a given Agreement.
Async Merge Call	Holds asynchronous merge calls.
Content Event	Holds content events for subscribers to handle.
Cycle Time Field	Holds the fields monitored for cycle time data capture.
Cycle Time Field Data	Holds the data captured on fields for cycle time reporting.
Cycle Time Group	Holds the groups monitored for cycle time data capture.
Cycle Time Group Data	Holds the data captured on groups for cycle time reporting.
Doc Assembly Component	Represents a single component in a document assembly rule.
Doc Assembly Rule	Represents a single rule in a document assembly ruleset.
Doc Assembly Ruleset	Represents a rule set that contains a list of rules to determine the sections to include in a dynamic template.
Document Agreement Clause	Agreement clauses in an agreement document.
Document Version	Represents a document version.
Document Version Detail	Holds the version details of a document.
Formula Field (Comply)	Represents a formula field used in agreement rules.
Merge Event	Holds document merge events for subscribers to handle.
Merge Event Detail	Holds document merge event details.
Query Template	Query Template for building SOQL queries and filters.
Query Template Filter	Filter expression used in a query.

Object	Purpose of the Object
Query Template Qualifier	Filter used to select a query template.
Related Agreement	To identify the different types of relationships between agreements.
Retention Policy	Holds object retention policies.
Search Filter (Comply)	Represents a search filter.
Temp Object (Comply)	A container for temporary objects and attachments.
Template	Contract language templates available in the system. These templates specify guidance, language, local settings, and include Word attachment templates representing the bulk of actual contracts that can be used when creating the first draft. Templates can include standard and non-standard language.
Template Clause Reference	Represents a clause reference in a template or a library.
Template Clause Reference Version	Holds Template Clause References for this version.
Template Datasource Filter	Represents a data source filter associated with a template.
Template Dynamic Section	Represents a dynamic section in a template.
Template Dynamic Section Version	Holds the template dynamic section details for a given template version.
Template Version	Holds various versions of this template and is a Master-detail to Template.
Term Exception	Term Exceptions master table.
Wizard	A runtime instance of the wizard design.
Wizard Design	Holds all the wizard design records.

Object	Purpose of the Object
Wizard Input Control	The individual question, instruction statements, and associated field attributes.
Wizard Rule	Individual rules which are applied within a ruleset.
Wizard Ruleset	A collection of rules that are applied to input control and steps. The system determines a ruleset at the level of input control expression formula, determine focus objects, record type, and step navigation rules.
Wizard Runtime Input	Input Control instance within a runtime wizard.
Wizard Step	The step that is designed and used in Wizard designs.

Conga Quote Management

Conga Quote Management allows you to create customized proposals including pricing and sales content. The right pricing and content is automatically selected and assembled based on business rules so that the sales representative can quickly get a compelling sales proposal to the prospect.

Object	Purpose of the Object
Document Collate Info	
Proposal Document Output Format	Holds default output format preferences by user profile and quote/proposal type for use in the quote/proposal document generation process.
Proposal Line Item	Represents a product or service line item associated with a quote/proposal
Proposal Request	Approval requests for a proposal
Quote/Proposal	Quote/Proposal object which holds dates and line items, many quotes/proposals may be related to a single opportunity, a quote/proposal may have many detail lines

Conga Quote Configuration Integration

Conga Quote and CPQ Integration integrates Conga Quote/Proposal Management with Conga CPQ. It enables the user to select products or services, configure and price them for simple or highly-complex scenarios, and produce high-quality customer-facing quotes and proposals.

Object	Purpose of the Object
Proposal Adjustment Line Item	Represents an adjustment to the price of a product/service.
Proposal Footnote	Represents the footnote available in a quote/proposal.
Proposal Incentive Benefit Data	Represents a single benefit data row associated with the incentive benefit for the quote/proposal.
Proposal Incentive Limit Data	Represents a single limit data row associated with the incentive limit for the quote/proposal.
Proposal Location	Represents the location available in a quote/proposal.
Proposal Product Attribute Value	Represents the attribute values for a product class in a proposal line item.
Proposal Rollup Data	Represents the dynamic rollup data for use in criteria.
Proposal Summary	Represents the summary of a product or service line item in a proposal.
Proposal Summary Group	Represents a summary of line items associated with a quote or proposal.
Proposal Tax Breakup	Represents the tax breakup of a product/service.
Proposal Usage Price Tier	Represents a single entry in a quote/proposal usage price tier.

Conga CLM Configuration Integration

Conga Contract and CPQ Integration integrates Conga Contract Lifecycle Management with Conga CPQ. It enables the user to select products or services, configure and price them for simple or highly-complex scenarios, and produce high-quality customer-facing agreements.

Object	Purpose of the Object
Agreement Adjustment Line Item	Represents an adjustment to the price of a product/ service.
Agreement Footnote	Represents the footnote available in an agreement.
Agreement Incentive Benefit Data	Represents a single benefit data row associated with the incentive benefit for the agreement.
Agreement Incentive Limit Data	Represents a single limit data row associated with the incentive limit for the agreement.
Agreement Product Attribute Value	Represents the attribute values for a product class in an agreement line item.
Agreement Rollup Data	Represents the dynamic rollup data for use in criteria.
Agreement Summary	Represents the summary of a product or service line item in an agreement.
Agreement Summary Group	Represents a summary of line items associated with an agreement.
Agreement Tax Breakup	Represents the tax breakup of a product/service.
Agreement Usage Price Tier	Represents a single entry in an agreement usage price tier.

Configuring Post-Installation Settings

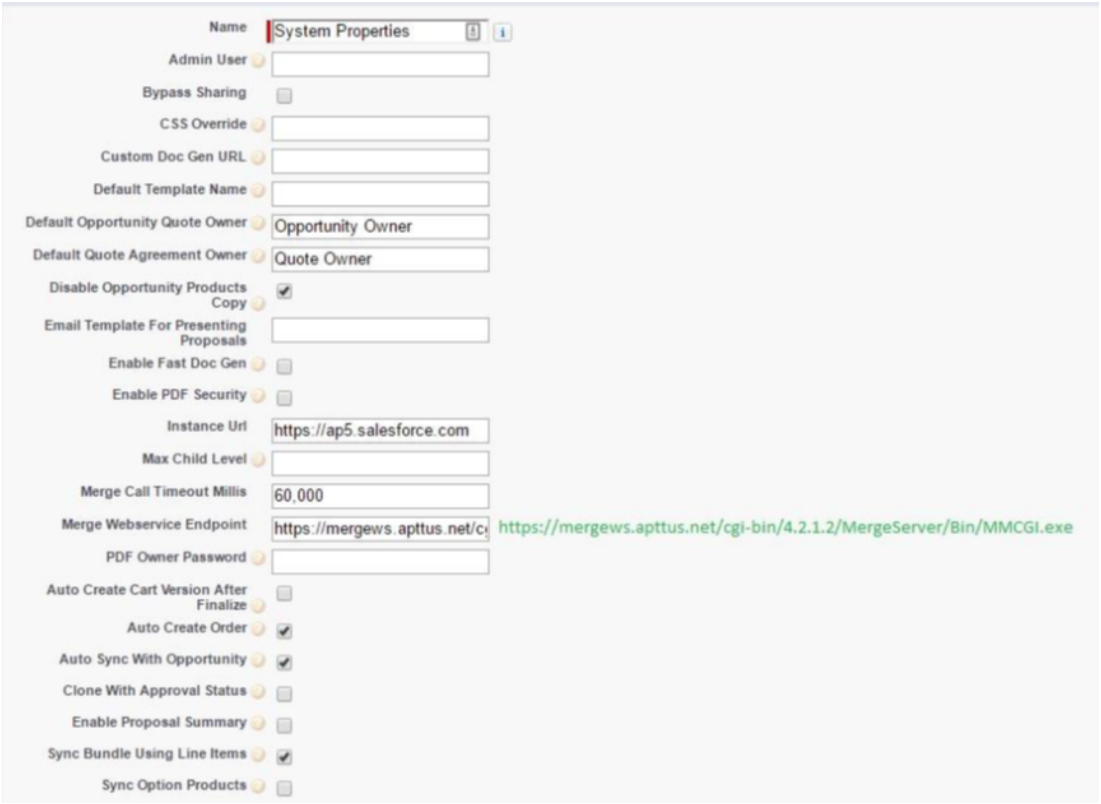
Post-installation settings are part of your specific business requirements which means your CPQ implementation will work with out-of-the-box installation, but you can customize the application per your specific implementation using custom settings.

Basic Post-Installation Configuration

These settings are mandatory settings in order for CPQ app to run.

Configuring Proposal System

1. Go to **Setup > Custom Setting > Proposal System Properties > Manage> New**
2. Refer to the following image and enter the values in the System property fields. (The values may vary according to your CPQ instance.)
3. Click **Save.**



Configuring System Properties

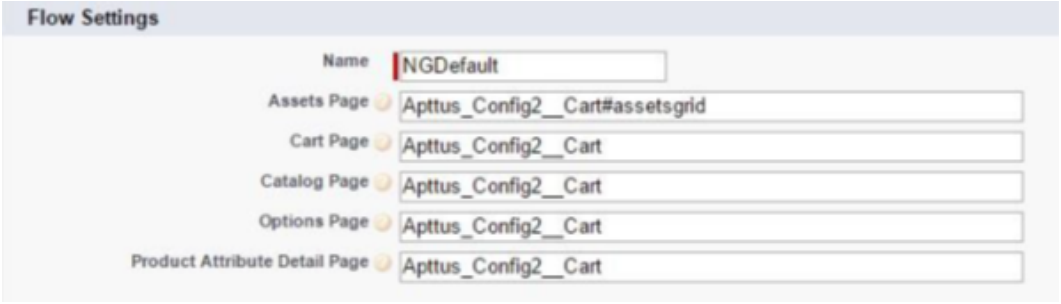
1. Go to **All tabs> Config Setting > System Properties**
2. Enter values in the fields.
3. Click **Save**.

Configuring Flow

1. Go to **All tabs> Config Setting > Flow Settings**
2. Enter the values in the fields. Refer to the image below for an example.

> ⓘ Do not use special characters in the name.

3. Click **Save**.



Configuring Custom Settings

1. Go to **Setup > Develop > Custom Setting > Config Select Products Settings > Manage**.
2. Find *Primary Settings*, Click **Edit**.
3. Enter a value in **Carousel Default Icon** ID (If you don't have any: create any notes & attachment, and get ID of that and enter here)
4. Click **Save.**

Running Custom Setting Maintenance Job

You must execute the Custom Setting Maintenance Job for all the flows after defining Custom Settings and when you define or update fields in Config Settings. For more information, refer to Running Maintenance Jobs.

Advance Post-Installation Configuration

- To customize the CPQ application, use the System Properties.
- To customize the product selection page, use the Catalog Page Settings.
- To customize the option selection pages, use the Bundle Page Settings.
- To customize the Installed Products page, use the Installed Products Settings.
- To customize Quote/Proposal, use the Proposal System Properties.
- To customize Comply, use the Comply System Properties.
- To support product and proposal summary in Salesforce Lightning, use the Object Summary Settings.

Configuring Application Management Settings

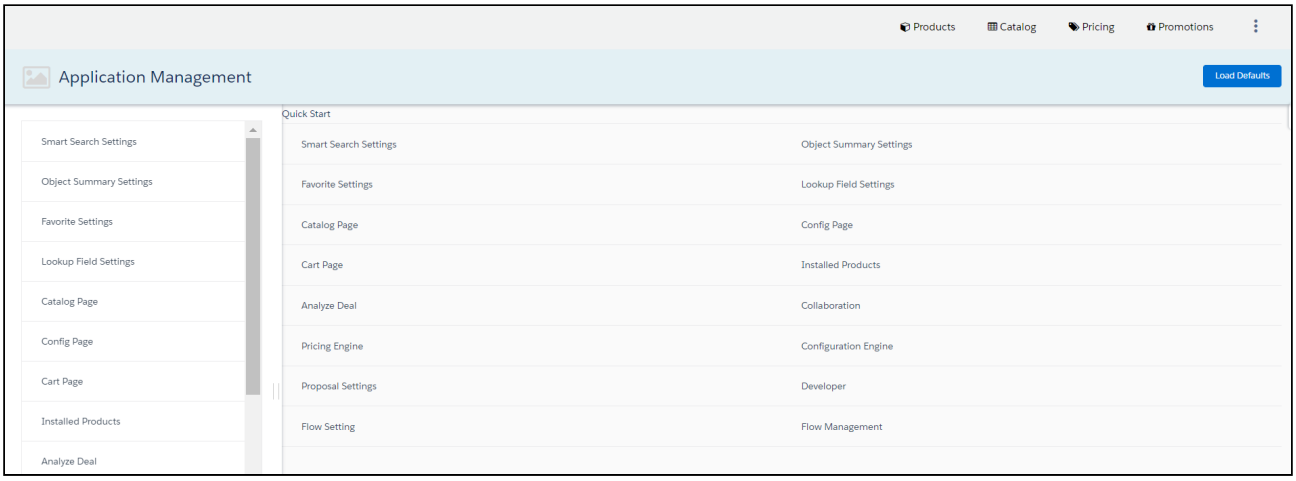
This section describes how you can configure Application Management settings.

The Application Management Settings allows you to configure and enable pages on the CPQ Admin console to easily configure different settings of the CPQ application from one single URL.

- **Launch from new Admin UI:** Settings can simply be modified by navigating to a new section in the New CPQ Admin user interface, allowing a logical grouping of the settings.
- **Logical grouping of settings**: By default, all settings related to pricing will be grouped together and items related to the Configuration Engine are grouped together.
- **Admin configurable settings:** You can also create your own group settings for customer specific settings that are not part of the standard CPQ settings.

Navigate to **Admin Console** and from the ⋮ drop-down list, click **Application Management**. Click **Load Defaults** to bring the custom settings for the default flow in the Application Management.

Using the **Load Defaults** functionality on the application management will auto-create the system settings and you must go with this approach.



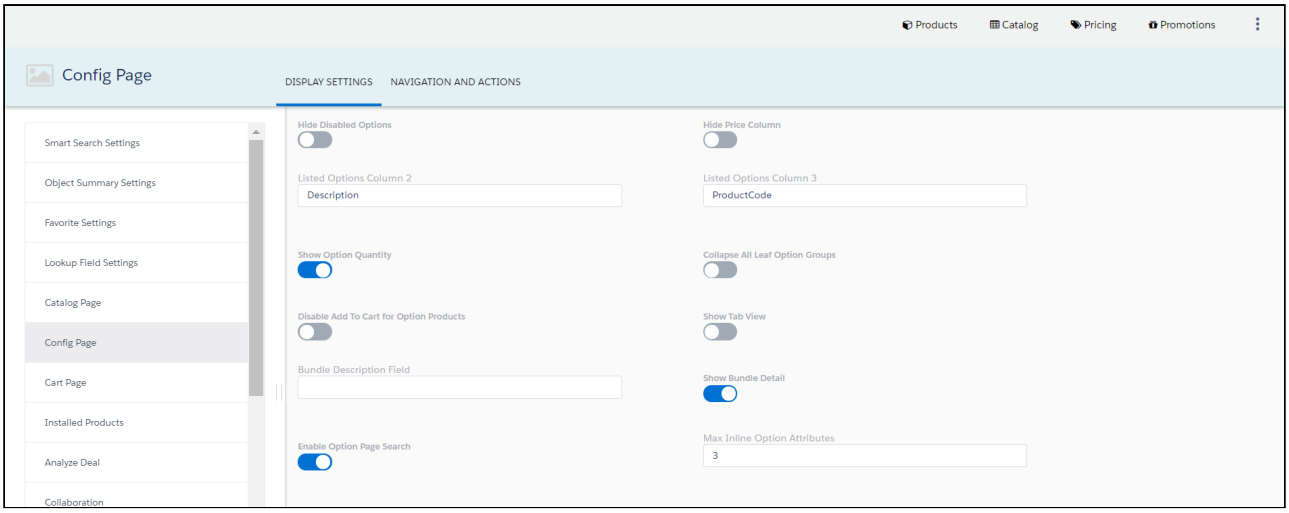
You must execute Custom Settings Maintenance Job after you update any Custom Settings. The changes you make are only effective if you execute the maintenance job. Otherwise, all the computations are processed with stale values. Refer to the Running Maintenance job topic.
You can use the link given next to the **Save** and **Cancel** button on every setting page.

Favorites Settings under Application Management lets you manage the favorite product related settings including the default image, the columns in the favorite dialog, and enabling or disabling the favorite settings using the slider. When you disable this setting,

the **Save as Favorite** option is not displayed in the cart. For more information on the configuration part, refer to Configuring Favorite Settings.

Lookup Field Settings under Application Management lets you manage the name of the record, an object of the lookup field, field name of lookup field, display columns, and filter criteria. For more information on the configuration part, refer to Configuring Lookup Field Settings.

For Quote, Product, and Favorites objects, you can define "Object summary" fields, which are shown when the user clicks on the link on the catalog such as Product Name or Favorites name. For more information on the configuration part, refer to Configuring Object Summary Settings.



> Disabled values are displayed with a dark gray background. CPQ does not allow you to access such values.

Note that the user with only System Administrator profile can access the Application Management functionality. If you log in through any other user, you will see this error message: *You do not have the level of access necessary to access the Application Management feature. Contact the administrator if access is necessary.*

> The Lookup Field tab available at **CPQ Admin > Application Management > Developer tab** is not functional with the current release.

Optional Configurations

Adding the Application Settings Management Objects as Tabs

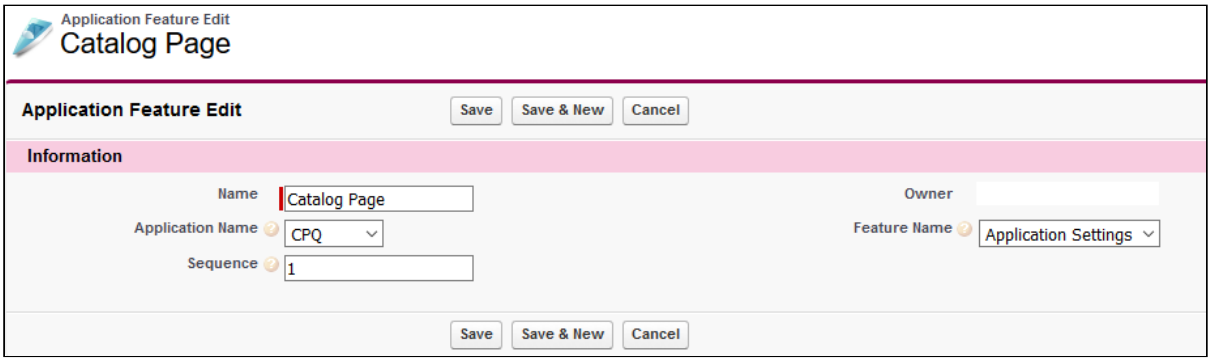
Go to **Create** > **Tabs** and create tabs for the following Objects.

Object	Description
Application Feature	This is a parent tab. The application settings available in Config Settings and Quote/Proposal System Properties are grouped under Application Features. For example, create an Application feature called as Catalog. You can group all the settings that affect the Catalog page under this application feature.
Setting Group	This is a related list in the Application Feature Visualforce page. Using this page, you can create logical groupings for each of the settings as well as the Flow in which these settings are available in. For example, for the Catalog page, you want to group settings by Display Settings and Navigation and Actions. The Display Setting group can have all the application settings that control the components displayed on the Catalog page, such as Hide Compare Products, Show Quantity. The Navigation and Actions Settings group have all the action and navigation settings such as, redirect to another page on button click without showing the spinner by using the Quick Redirect Action type.
Application Setting	This is a related list in the Setting Group Visualforce page. Each Application Setting has to be defined based on its behavior within a setting group. For example, you can group all application settings controlling the display on the Catalog page together. For example, the Show Quantity application setting will be a part of the Display Setting group.

Creating an Application Feature

When you navigate to the Application Feature tab and click **New** to add and save details in the fields. You can create multiple application features.

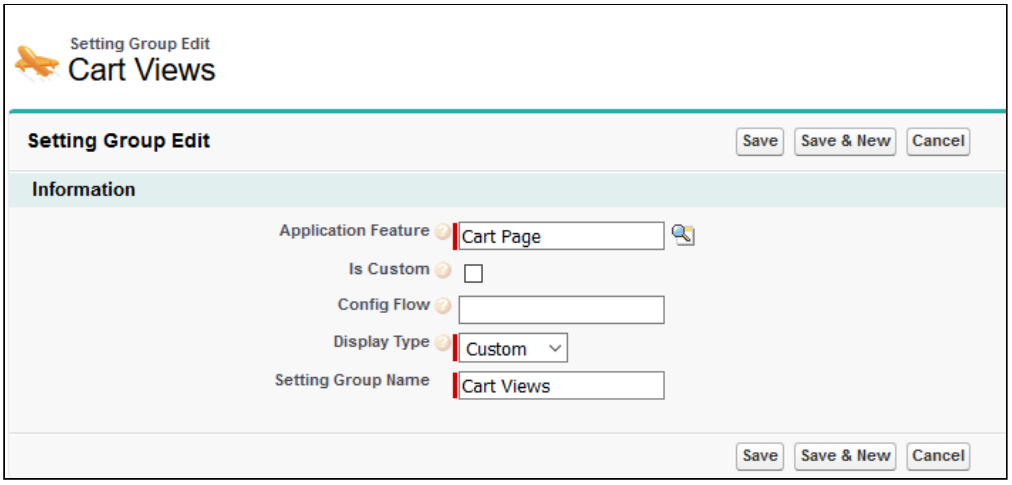
Field	Description
Name	Name that you want to give to the application feature. Let's name the Application feature as *Catalog Page*.
Application Name	The Application in which this feature setting will be applicable. By default the value is CPQ. If you want to add additional values, navigate to the Application Settings object and add additional fields to the Application Name picklist such as CLM.
Feature Name	The feature for which this Application Setting is defined. For example, Cart Page, Catalog Page, Flow Setting, Proposal Settings. You can add values to this picklist by navigating to the Application Settings object and add additional fields to the Application Name picklist. An example of a feature name could be Catalog page, Cart page, or just Application Settings.
Sequence	The sequence in which the Application Feature appears.

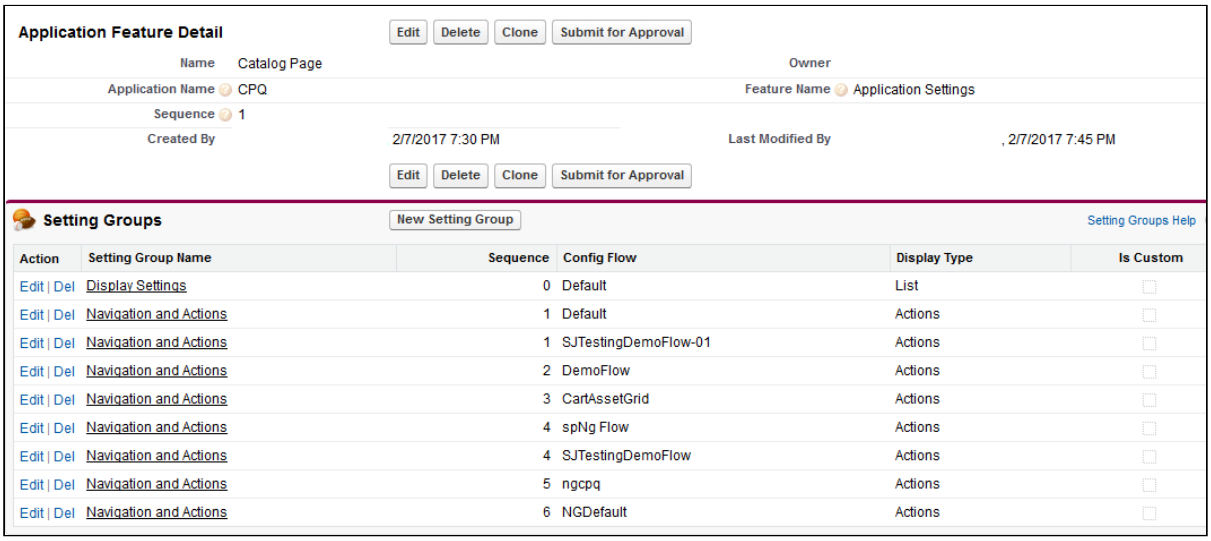


Creating a Setting Group

After you have created an application feature, use the Setting groups' related list from the Application Feature Visual force page to create a new Setting Group. You can create multiple Setting Groups within an Application Feature. Click **New** to add and save details in the following fields:

Field	Description
Sequence	The Sequence in which the Setting group will appear on the New CPQ Admin Page.
Application Feature	The name of the application feature, such as Cart Page.
Setting Group Name	The name you want to give to the Setting Group, such as Display Actions, Cart Views, and so on.
Config Flow	Specify the name of the flow in which the setting group is applicable and visible in.
Display Type	Specifies how the application settings defined in the Setting Group appear. Set the display type on the group based on the Application setting you define in the group. For example, all System settings would have display type as Lists, if all the application settings in the group control or specify custom display actions, the display type would be Actions and so on.
Is Custom	Specifies whether the Setting Group is a custom defined setting group.





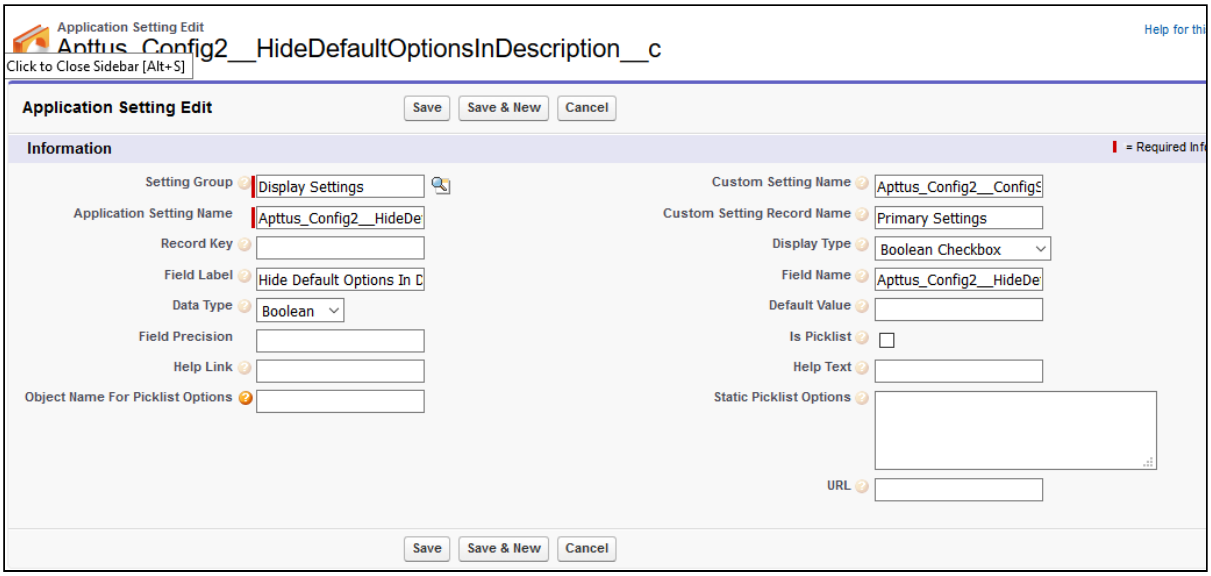
Creating an Application Setting

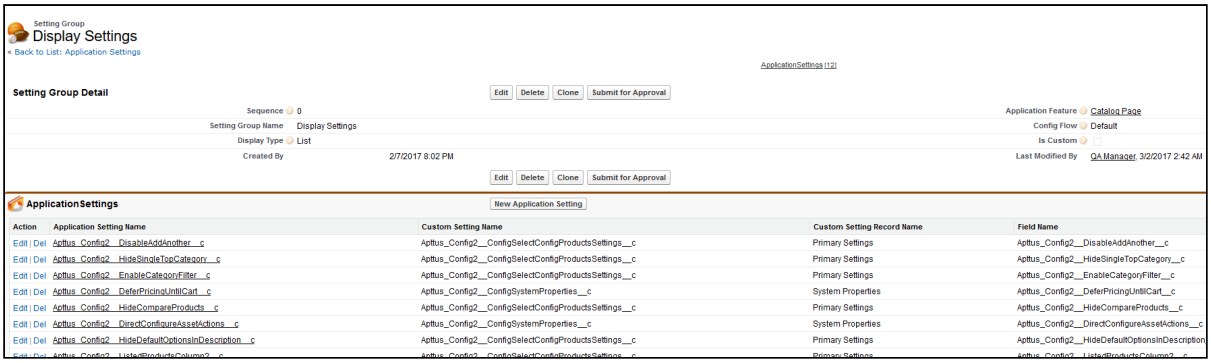
After you have created a Setting Group, use the Application Features related list from the Setting Groups Visualforce page to create a new Application Setting. You can create multiple Application Settings within a Setting Group. Click **New** to add and save details in the following fields:

Field	Description
Setting Group	A lookup to the setting group the application setting will belong to.
Application Setting Name	Specify a name for the application setting.
Record Key	Records associated with the custom setting will be fetched using a record key. When your setting Group is of type Actions or Columns, specify a record key. Record Key is used to identify records in the associated Custom Setting.
Field Label	Name of the application setting. For example, **Hide Default Options** In Description.
Data Type	This value can be Boolean, number, picklist, text. In our use case, Hide Default Options in Description is a checkbox so the data type will be boolean.

Field	Description
Field Precision	This field specifies the to specify the number of decimal places shown for currency, quantity, and percentage precision fields.
Object Name For Picklist Options	Specify the API Name of an object whose fields are displayed as available options if the setting is a picklist.
Custom Setting Name	Specify the API name of the custom setting which contains the application setting. For example, the Hide Default Options in Description Setting is available in General Page Settings of the Catalog Page which is essentially defined as Apttus_Config2__ConfigSelectConfigProductsSettings__c. So you'll pass the API name of the Setting containing the Application Setting.
Custom Setting Record Name	Name of the custom setting to which the application setting belongs to. For example, Hide Default Options In Description is a part of the Primary Setting. Specify the Record Name as Primary Setting. The values can be System Properties, Primary Settings, Config Custom Classes and so on.
Display Type	Defines how the Application setting is displayed in the CPQ Admin Console page. This value can be Boolean, Multiselect picklist, picklist, text, or radio button. In our use case, Hide Default Options in Description is a checkbox so the data type will be boolean.
Field Name	API name of the custom setting. For example, Apttus_Config2__HideDefaultOptionsInDescription__c.
Default Value	Specify a default value of the setting. For example for HideDefault Options, in Descriptionfield, the value can be set to *true*.
isPicklist	Is a boolean checkbox specifying whether the application setting is a picklist.
Static Picklist Options	Specify a comma-separated list of the picklist options if the application setting is a picklist.

Field	Description
URL	Specify a URL which provides more information for the custom setting.





Configuring Smart Search Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon (⋮) on the top-right corner and select **Application Management**.
3. Click **Smart Search Settings**.
4. Configure the following details.

Setting	Description
SEARCH CONNECTION SETTINGS	
Enhanced Search URL	Specify the URL end point for enhanced product search. With **Enhanced Search URL** populated with the correct URL, the system synchronizes all the changes at every hour and displays the results in the **Review Sync Status** section. If you do not specify the **Enhanced Search URL**, the system neither performs the sync nor updates the information in the **Review Sync Status** section.
API User Key	Specify the API key that consists of the client ID and is used for smart search related activities. This key is required before Search Activation/ Deactivation and resync. An error is displayed if you activate, deactivate or resync without the API User Key.
PRODUCT FIELD WEIGHTAGE	
Flow	Select a flow and enter the following details. • Field Name • Typeahead • Weightage

Configuring Favorite Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon (⋮) on the top-right corner and select **Application Management**.
3. Click **Favorite Settings**.
4. Enter details in one or more of the following sections, based on your organization's requirement.

Setting	Description
Disable Favorites	Disables the Favorites category on the catalog, **Save As My Fav** button on the cart and any UI elements related to Favorite Configurations on catalog, configuration and cart pages. By default, this checkbox is not selected.

Setting	Description
Listed Favorite Configuration Column1	Enter the API name of the favorite configuration field that you want to display on the catalog page for your favorite configuration records. You can enter a maximum of one API names. If you have a requirement of displaying one more field, use **Listed Favorite Configurations Column 2** custom setting.
Save as Favorite Dialog Columns	Enter comma-separated API names of the columns that you want to display on the **Save As My Favorite** dialog box on the cart.
Favorites Category Image	You can choose an image to upload for your group of favorite configuration category. This image is displayed under your favorites (as entered in **Favorites Display Label**) category on the catalog page. By default, no image is set.
Listed Favorite Configuration Column2	Enter the API name of the favorite configuration field that you want to display on the catalog page for your favorite configuration records. You can enter a maximum of one API name.

Configuring Catalog Page Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Catalog Page**.
4. Configure the following details.

Setting	Description
NAVIGATION AND ACTIONS	

Setting	Description
Flow	Select a flow and enter the following details. • Display As: • Action Area: • Action Style Class: • Is Enabled: • Always Display:
DISPLAY SETTINGS	
Hide Help Me Choose	Indicates whether to hide the Guided Selling link. By default this is not selected and the system shows Guided Selling link on the Catalog page for the respective categories.
Hide Configure Action	Hides the Configure action for a Product on the catalog page.
Hide Listed Products Price Column	Hides the price information column in listed product section. By default, this is not selected.
Listed Products Column3	(optional) Type the API name of the product field to display in the listed products section. Example Value: ProductCode
Search Category Default	Indicates the category to default to when searching for products using the search text box on the product selection page. The default value is All Categories. Select All Products to search all products regardless of categories.
Enable Category Filter	Filters the categories based on search results and displays in the Narrow Your Search area on the left. For example, if the search result has products from only two categories, the Narrow Your Search section displays only those two categories. You should always select Enable Category Filter when Hide Narrow Your Search is cleared.
Hide Compare Products	Disables the Compare Products feature in the catalog page.

Setting	Description
Hide Default Options In Description	Hides the default options listed below the product description.
Listed Products Column2	(optional) Type the API name of the product field to display in the listed products section. Example Value: ProductCode
Show Quantity Input	Displays the Quantity field on the catalog page.
Hide Single Top Category	Hides the top category. This property is useful only when you have a single root category and want to have a cleaner look for all of its child categories, without the root category.

Configuring Cart Page Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Cart Page**.
4. Configure the following details.

Setting	Description
DISPLAY SETTINGS	
Enable Auto Sequencing For Options	Enables auto sequencing for options. If you do not select this option, the Option Item Sequence in the shopping cart and the Line Itemsisbased on the user selection sequence rather than the Bundle Options sequence.
Product Display Max Length	Captures the maximum no of characters to be displayed for the product name on the cart page. If the product name exceeds the maximum length, the ellipsis is shown. The default value is 21.

Setting	Description
Expand Bundles in Cart	Expands all attributes and options of a bundle product in the cart. ⓘ The pagination feature is not supported along with Expand Bundles in Cart.
GroupBy Fields	Specify the API names of those line item fields that are of the type: Lookup, Formula Fields, and Text. These line item fields are shown under the Group By dropdown on the cart page. For example, for Group By PriceList, specify the API name of the field that is a lookup to the pricelist object. These fields are also used in the Cart View feature. For more information, see Viewing the Cart in Grid View.
Hide Subtotals In Cart	Indicates whether subtotals should be hidden in the cart view. The default value is false (show subtotals in the cart). When checked, the system hides subtotal lines in the Summary section, and still shows Category Total, One Time Total, and Grand Total.
Show Recommended Products Cart View	Displays the recommended products component in the Cart Detail View page.
Show Attributes in Cart	⚠ This setting is deprecated.
Hide Copy Action	Hides the copy action in the cart. The default value is false (copy icon is displayed).
Hide Grand Total	Hides the grand total line on the cart page.
View Cart Total Custom Fields	Enter the list of custom fields from the summary group object displayed in the custom view cart page. Each field API name should be separated by a new line or a comma.
NAVIGATION AND ACTIONS	

Setting	Description
Flow	Select a flow and enter the following details. • Display As: • Action Area: • Action Style Class: • Always Display:
CART LINE COLUMNS	
Flow	Select a flow and enter the following details. • Field Name Select the required columns that should be displayed on the cart for the user. The **Renewal Adjustment Type** and **Renewal Adjustment Amount** adjustment types are applicable only for Asset Renewal action. • Is Editable • Style • Style Class • Header Style
TOTALS COLUMNS	
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style
PRICE RAMP	
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style

Setting	Description
USAGE PRICE TIER	
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style
TIERED PRICE	
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style
RELATED PRICE	
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style
ADJUSTMENT LINE ITEM	
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style
MASS UPDATE	

Setting	Description
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style
AGREEMENT PRICE RULE	
Flow	Select a flow and enter the following details. • Field Name • Is Editable • Style • Style Class • Header Style

Configuring Analyze Deal Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon (⋮) on the top-right corner and select **Application Management**.
3. Click **Analyze Deal**.
4. Configure the following details.

Setting	Description
NAVIGATION AND ACTIONS	
Flow	Select a flow and enter the following details. • Action Label Name: • Display As: • Action Area: • Action Style Class: • Action Page: • Action Param: • Behavior: • Is Enabled: • Always Display:

Setting	Description
COST LINE ITEM	
Flow	Select a flow and enter the following details. • Field Name: • Is Enabled: • Style: • Style Class: • Header Style:
LINE ITEM WATERFALL	
Flow	Select a flow and enter the following details. • Field Name: • Is Enabled: • Is Price Point:
LINE ITEM METRICS	
Flow	Select a flow and enter the following details. • Field Name: • Is Enabled: • Style: • Style Class: • Header Style:
SUMMARY METRICS	
Flow	Select a flow and enter the following details. • Field Name: • Is Enabled: • Style: • Style Class: • Header Style:
SUMMARY WATERFALL	
Flow	Select a flow and enter the following details. • Field Name: • Is Enabled: • Is Price Point:

Configuring Flow Management Settings

This page contains the settings to determine the custom or out-of-the-box pages when an end user clicks **Configure Products** button. Once setup, these flow settings override the Visualforce pages setup in custom settings.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Flow Management**.
4. Enter details in one or more of the following sections, based on your organization's requirement.

Setting	Description
Flow Name	Type a mandatory name for the flow.
AssetsPage	Type the Visualforce page name for assets. Valid values are: • SelectInstalledProducts • SelectInstalledProductsEn
CartPage	Type the Visualforce page name for cart. Valid values are: • Apttus_Config2__CartDetailView • Apttus_Config2__CartView If you want to view the cart with the new Grid UI, name the cart page as *Apttus_Config2__Cart#cartgrid.*
CatalogPage	Type the Visualforce page name for product selection. Valid values are: • Apttus_Config2__SelectConfigProducts • Apttus_Config2__SelectConfigProducts FilterView

Setting	Description
OptionsPage	Type the options page to capture the options for a bundle product. Valid values are: • Apttus_Config2__ConfigureBundle • Apttus_Config2__SelectConfigOptions • Apttus_Config2__SelectConfigOptionsTabView • Apttus_Config2__SelectConfigOptionsListView • Apttus_Config2__SelectConfigOptionsTreeView • Apttus_Config2__SelectConfigOptionsDetailView • Apttus_Config2__SelectBundleOptions ⚠ Do not use the *SelectBundleOptions* page from your drop-down list, it is no longer supported.
ProductAttributeDetailPage	Type the Visualforce page name to capture product attribute detail information. Valid value is: • Apttus_Config2__ProductAttributeDetail3

Configuring Configuration Engine Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Configuration Engine**.
4. Configure the following details.

Setting	Description
CONSTRAINT RULES	
Disable Constraint Rules	Indicates whether constraint rules are disabled. This may be used to optimize performance when constraints are not used in the organization. The default value is false (enabled).

Setting	Description
Defer Constraint on Option Selection	Select this setting to process the constraint rules on option selection on the Configuration Page. If the selected the constraint rules deferred until the Sales Rep clicks **Go to Pricing** or any other action button available on the Configuration page.
Defer Validation Check In Bundles	Defers the validation check in the bundle configuration step. This is selected by default.
Show Checkbox In Rule Dialog	Enables check box to select multiple products in the Inclusion Constraint Rules dialog at once, instead of adding the products individually using Add to Cart. This setting works for *Inclusion Rule* with Min/Max Match Rule as *Include Min/Max*.
Max Constraint Rules Round Trip	This is the maximum number of round trips after which the constraint rule processing should stop. Round trip happens only when auto included products trigger more rules. The default value is 3.
Constraint Rule Execution Mode	Enter the mode to perform the constraint rule execution on the client-side or the server-side. By default, no value is specified and the system assumes the value as *Server*. Valid values are: • *Client* • *Server* • *CMS* This custom setting supports inclusion, exclusion, and validation rules on the client-side as well as the server-side. ⓘ In a Service CPQ flow, always execute constraint rules in Client mode.
ATTRIBUTES	
Cascade Shared Attribute Updates	Cascades the shared bundle attribute updates to options in the bundle. If this setting is unchecked, the bundle attributes will initially be cascaded to options in the bundle, but subsequent changes in the bundle attribute will not be cascaded to option attributes.

Setting	Description
Populate Attribute Extensions	Select this setting to indicate that the attribute values are populated in the attribute extension objects.
Product Attribute Extension Tables	Enter comma-separated API names of the lookup relationships, created with the **Product Attribute Value** object. For example, if you have 4 extension objects to **Product Attribute Value** object and their corresponding 4 lookup relationships, you must specify 4 API names in this field.
EXPRESSIONS	
Enable Field Expressions	Enables you to use Expression Builder for attributes, option groups, and options.

Configuring Developer Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Developer**.
4. Configure the following details.

Setting	Description
SYSTEM GLOBAL	
Instance URL	The Salesforce instance URL (for example, https://na7.salesforce.com/). The instance URL is required to navigate to custom pages in the managed package.
CSS Override	Type the name of the static resource to override CSS in the catalog, attributes, options, and cart pages.
Auto Update Category View	Indicates whether auto incremental updates to category view are enabled.
Keep Abandoned Carts	Indicates whether abandoned carts should be kept. The default value is false (abandoned carts are moved to the recycle bin).

Setting	Description
Bypass Shopping Cart	Displays the **Update Price** and **Finalize** buttons and hides the **Go To Pricing** button on all the new pages. On the configuration page, only the **Go To Pricing** button is disabled. You can still finalize the quote using the Mini cart.
FIELD PRECISION	
Percentage Field Precision	Type a number to specify decimal places for percentage precision. The default percentage precision is 2 decimal places.
Quantity Field Precision	Type a number to specify decimal places for quantity precision. The default quantity precision is 2 decimal places.
Currency Field Precision	Type a number to specify decimal places for currency precision. The default currency precision is 2 decimal places.
Term Field Precision	Type the precision value that you want to display for the **Selling Term** column on the shopping cart. For example, if your term calculation results to 4.553412 and you have set **Term Field Precision** to *3*, the **Selling Term** column displays 4.553. The default term precision is 5 decimals. This setting is applicable only for the New UI.
FINALIZATION & APPROVALS	
Run Misc Finalization Task In Async Mode	Indicates whether the secondary finalization task (attributes, usage tiers, etc) should be run asynchronously
Auto Sync On Cart Approval	Automatically synchronizes the cart when approved and finalized.
Auto Finalize On Cart Approval	Automatically finalizes the cart when the quote gets approved.

Configuring Object Summary Settings

In order to show the summary information for the Proposal (cart header) and Product in the Salesforce Lightning Experience, a new custom setting named **Object Summary Settings** is now available.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon (⋮) on the top-right corner and select **Application Management**.
3. Click **Object Summary Settings**.
4. Configure the following details.

For Proposal Summary

Field Name	Description
Name	Specify the name as: Proposal Summary
Object Name	Specify the object name as: *Apttus_Proposal__Proposal__c*
Display Fields	Specify the API names as: *Name,Apttus_Proposal__Proposal_Name__c,Apttus_Proposal__Account__c,PriceListid__c*

You can specify any Quote/Proposal field in a comma-separated list.

For Product Summary

Field Name	Description
Name	Specify the name as: Product Summary
Object Name	Specify the object name as: *Product2*
Display Fields	Specify the API names as: *Name,Description,Family*

You can specify any Product2 field in a comma-separated list.

On the catalog page, when you click **Proposal ID** on the top left corner, you see proposal summary with the fields configured in Proposal Summary.

Similarly, when you click **Product Name** on the catalog page, you see product summary page with the fields configured in Product Summary.

If there is no **Object Summary Setting** defined in the system, only Name will be displayed by default when you open the Proposal or Product summary dialog box.

For Favorite Configurations

Field Name	Description
Name	Specify the name as: *Favorite Configuration*
Object Name	Specify the object name as: *Apttus_Config2__FavoriteConfiguration__c*
Display Fields	Specify the API names as: *Name,Apttus_Config2__PriceListId__c,Apttus_Config2__ConfigurationId__c*

Configuring Lookup Field Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. Click the more icon (⋮) on the top-right corner and select **Application Management**.
3. Click **Lookup Field Settings**.
4. Click **New Lookup Field**.
5. Configure the following details.

Setting	Description
LOOKUP FIELD SETTINGS	
Name	Specify the name for the custom setting.
Object Name	Select one from the dropdown menu. • Collaboration Request • Line Item • Product Attribute

Setting	Description
Lookup Field Name	Select the name of the field from the dropdown menu. The values are populated based on the Object you select. This is the lookup field containing values such as *Enterprise*, *Standard*, and *Premium*.
Filter Criteria	Enter a valid expression to further filter down the search results for a lookup. The field size is 256 characters. Example: Product__c=Apttus_Config2__ProductAttributeValue__c.Product_Id__c This will fetch only such attributes whose IDs match with the selected products. Here, the ProductId is a custom formula field which fetches the Product Id of the Line Item object.
Lookup Display Columns	Select the fields to be displayed on the lookup dialog box.
Lookup Record Limit	Click the icon, enter the maximum number of records to query for the lookup field dropdown, and click **Save**. Default value is 200 to a maximum of 1000.
Junction Object Name	Click the icon, enter the name of the junction object which you would like to filter the lookup values, and click **Save**. This is a custom object that you can define to filter the values which do not exist on the Lookup Object itself.
Junction Field Name	Click the icon, enter the name of a field on the junction object which you would like to filter the lookup field by, and click **Save**.
Junction Default Flag	Click the icon, enter the name of a field API from the Junction object to determine the default value of a lookup field, and click **Save**. This field must be a checkbox.
Filter Criteria 2	This field is not available in the CPQ Admin UI. You must use **Setup** > **Build** > **Develop** > **Custom Settings** > **Lookup Field Settings** to configure it.

Configuring Config Page Settings

This page lists all the custom settings you may require to set up the options page. You can enter details in one or more of the following sections, based on your organizations' requirement.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Config Page**.
4. Configure the following details.

Setting	Description
DISPLAY SETTINGS	
Exclude Optional Products	Select this to exclude optional products when CPQ evaluated the bundle's structural requirements. When you enable this setting, CPQ excludes the optional products from the option group Min/Max criteria.
Hide Disabled Options	Hide the options that have been disabled by the exclusion rules. Such options will not be shown in the list of available options. Also, if all such disable options are under one option group, this group is also hidden on the configuration page.
Listed Options Column 2	Type the API name of the product field to display in the listed options section.
Show Option Quantity	Displays the Quantity in the body of the options pages. This setting controls the SelectConfigOptionsDetailView and ConfigureBundle page.
Disable Add To Cart for Option Products	Hides the Add To Cart and Configure buttons on the Catalog page for an option product. This is helpful to prevent your sales representative from selling an option product without its bundle product.
Show Bundle Detail	Shows the bundle details in the options page.
Hide Price Column	Disables the price column from the options page.

Setting	Description
Listed Options Column 3	Type the API name of the product field to display in the listed options section.
Collapse All Leaf Option Groups	Collapses all the leaf option groups in the Config Options Detail View page.
Show Tab View	Displays option groups as tabs instead of sections in the options page. You can enable the Show Tab View setting at a product level also. Select the check box on the product details page.
Bundle Description Field	Type the field names from the product object to be displayed as bundle description.
Enable Option Page Search	Select this setting to allow option search on the bundle configuration page. This setting enables a search bar on the Bundle Configuration page. The setting allows the user to search for options, attributes, and sub-bundles on the Configuration page.
Max Inline Option Attributes	Specify the maximum number of inline option attributes to be displayed in the attribute group. The default value is 3.
NAVIGATION AND ACTIONS	
Flow	Select the flow you want to configure. • Display As: • Action Area: • Action Style Class: • Always Display:
Display Type	The value of this field is set to *Attribute Item*. This indicates that you are configuring the actions available on the Attribute page.
Action	The name of the action available on the Attribute page.
Actions Label Name	The label of the corresponding **Action** action that is displayed on the Attribute page

Setting	Description
Display As	Select one of the following values: • *Action*: Select this to set the action as an action button • *Sidebar Action*: Select this to set the action as an action button on the sidebar of the Configuration page. You can define a maximum of 3 actions in the sidebar. Only the first 3 sidebar actions in the list are valid, CPQ does not display the remaining sidebar actions on the Configuration page at all. • *Task*: Select this to set the action as Task menu item in the Status bar.
Action Area	Select one of the following values to define the alignment of the buttons on the Attribute page: • *Center* - For center alignment • *Left* - For left alignment • *Right* - For right alignment • *More* - To add as an item under the More action button
Action Style Class	Enter one of the following values: • *apt-selected-btn*: To display the button in the drop-down menu with a white background and black font color. This also indicates the step that an end-user is on. • *apt-left-btn*: To display an arrow pointing left on the button. • *apt-right-btn*: To display an arrow pointing right on the button.
Is Enabled	Select this to enable the button.
Always Display	Select this to display the button on the Attribute page. The button is displayed even if you disabled the buttong using **Is Enabled** checkbox.

Configuring Installed Products Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Installed Products**.
4. Configure the following details.

Setting	Description
NAVIGATION AND ACTIONS	
Flow	Select a flow and enter the following details. • Display As: • Action Area: • Action Style Class: • Always Display:
COLUMNS	
Flow	Select a flow and enter the following details. • Field Name • Is Sortable • Style • Style Class • Header Style
GENERAL SETTINGS	
Direct Configure Asset Actions	This setting is no longer supported. Instead, you can use the **Allowed Actions** setting to configure asset actions at each Asset Line Item. Also, you can use the **Hide Asset Actions** setting to hide the action buttons from users on the Installed Products page.

Setting	Description
Submenu Actions	Enter comma-separated submenu actions that you want to display when the user clicks an action button on the Installed Products page. You can configure submenu actions only on the **Change** and **Relate** actions. The supported values are: • **Change:Configuration** displays the Configuration submenu under the Change action. This submenu enables users to configure an asset line item. • **Change:Quantity** displays the Quantity submenu under the Change action. This submenu enables users to initiate an action to add more licenses for an existing asset stream. • **Change:Split** displays the Split submenu under the Change action. This submenu enables users to split an asset into multiple split lines. • **Change:Merge** displays the Merge submenu under the Change action. This submenu enables users to merge multiple asset lines into a single asset. • **Relate:Component** displays the Component submenu as a drop-down under the Relate action. This submenu enables users to relate an asset component with a service.
Show Accounts Filter	Selecting this displays the account filter in the installed products page. This is currently not supported.
Filter Fields	Type the *API name* of Asset Line Fields to be displayed on the **Filter By** panel. Works with text fields, Picklist, Multi-picklist, Date, DateTime and Boolean. Commonly used fields are **Start Date**, **End Date**, **Lead Time Expiration**, and **Asset Status**. Usually, all fields under Asset Line Item object are supported but you must enter *ExpirationLeadTimeInDays* value in order to display Lead Time Expiration field in the Search pane.
Hide Co-Term	Hides the Coterminate with Increment panel on the Change Quantity (Increment Asset) page. The Coterminate with Increment panel is displayed by default.

Setting	Description
Show Assets	Select the account hierarchy that the user must see on the Installed products page. Select from the following settings: **Parent and Children**: The entire account hierarchy including the parent and children are displayed. **Parent:** Only parent accounts are displayed. **Children**: Only children accounts are displayed.
Hide Asset Actions	Type comma-separated values of Increment, Amend, Renew, and Cancel actions. Listed actions will be hidden from the users. You can configure asset actions at each Asset Line Item.
Max Renews Per Trip	Type the maximum number of renewals per trip. To prevent CPU timeout use a smaller number. The default is 20.
Amend Change Fields	Type comma separated line item field names whose values can turn an existing asset line into an amended asset line. This setting is no longer used. All the editable fields honor the flow settings configured for the Cart.
Default Renewal Term	Type a value for a renewal term. When the value is provided use this value as the renewal term for Renewing products. The value here denotes months to renew as the end date. Ideally, your **Selling Term** is equal to your **Default Renewal Term**.
Allow Backdated Termination	Enables you to terminate an asset by specifying an end date to any date prior to the current date. End date should be greater than the start date. Backdated Termination is supported out-of-the-box.
RENEWAL	
Renewal Lead Time	Indicates when the renewal quote will be created after an order is activated. By default, the value is 0, which indicates that the renewal quote is created immediately after the order is activated, with the same number of Line Items in the order. If you specify 30, the renewal quote is created 30 days before the Asset End Date.
Default Renewal Price List	Specify the Name of the Price List which is a mandatory field for Quote creation.

Setting	Description
Renewal Default Price Book	Enter a Renewal Price Book name which you want to associate with renewals.
Renewal Group Fields	Indicates how to group the Asset Line Items when a Renewal Opportunity is created. Specify the API names of the fields you want to use for grouping. • For Execution mode set to **Auto**, the system will group the renewal Opportunity by default by the **Auto Renew** Flag - One for the Asset Lines with **Auto Renew** as *True* and the other for Asset Lines with **Auto Renew** as *False*. • For Execution mode set to **OnDemand**, the system can group the renewal Opportunity by Account and the Price List. If the implementation teams want to group the renewal opportunities by other parameters on the Asset Line Item (for example, end date, product family, or any custom fields), they can do so by specifying a comma-separated list of API names of the fields in this section. It is recommended to limit the grouping to a maximum of 4 fields.
Renewal Execution Mode	Indicates if the Renewal of Asset Line Items must happen automatically or based on your specified conditions. • Enter **Auto** to automatically create the Renewal Opportunity on Order activation. Make sure you check **Auto Renew** on the product **PLI**, from the **Default** tab. • Enter **OnDemand** to create the renewal Opportunity before a certain lead time. You must enter a **Renewal Lead Time** for this mode to work successfully. To create a renewal Opportunity 90 days before the Asset Expiry, set the **Execution Mode** to *OnDemand* and **Renewal Lead Time** to *90*.
Renewal Business Object Type	Enter the Business object for which this renewal is taking place. Currently, *Proposal* and *Agreement* are supported.

Setting	Description
Cotermination Preference During Renewal	Select one or more of the following options to allow the user to determine renewal end dates. • UseProposalOrAgreementEndDate: This option enables users to renew an asset using the proposal or agreement end date. CPQ displays the **Use Proposal End Date** option if the user renews assets using the quoting flow and the Renewal Business Object Type is Proposal. CPQ displays **Use Agreement End Date** if the user renews assets using the agreement flow and the Renewal Business Object Type is Agreement. ⓘ If the user performs asset renewal through the Asset Manager flow or CSR flow, the **Use the Proposal End Date** or **Use Agreement End Date** option is not displayed on the Confirm Renewal intermediate page. • RetainCurrentAssetEndDate: This option enables users to retain the current asset end date while renewing an asset. • UseFarthestAssetEndDate: This option enables users to use the farthest asset end date while renewing assets. • UseRenewalDate: This option enables users to renew assets with a custom renewal date. You must select at least one value in this setting. If you have more than one value, you must also select a default value in the **Default Renewal Cotermination Option** setting. If you do not select any default value, the first of the selected values is considered as the default value. Depending on the selected setting, users can now bypass the Confirm Renewal intermediate page altogether. • If there is only selection for co-termination (apart from *UseRenewalDate*), the Confirm Renewal intermediate page is completely bypassed. On clicking Renew, CPQ creates the renewal line and refreshes the Installed Products page to show the detail in the Mini Cart. • If there is more than one selection for co-termination, CPQ displays the Confirm Renewal intermediate page to allow the user to make a selection. • If the selected option is *UseRenewalDate*, CPQ displays the Confirm Renewal pop-up to collect the user input.

Setting	Description
Default Renewal Cotermination Option	Select the default end date preference for renewal cotermination. If there is only one value in the **Cotermination Preferences During Renewal** setting, that option is selected by default. If there are more than one options, you can select a value as a default renewal cotermination option. Based on the default option, CPQ loads the selected renewal transactions on the Installed Products page, Confirm Renewal popup, or Confirm Renewal intermediate page.
MISCELLANEOUS	
Enable One Time Change	Select this flag to enable modifications or amendment to one-time assets. This is a global setting and cannot be restricted to apply for specific products. You can perform only amendments on one-time assets and not renewals. If you deselect this and modify the asset, the renewal quote is updated for the modified asset. If you set it to *false*, Renewal Opportunity and Renewal Quote are not created for the one-time only products, but if you modify the asset, the change is reflected in the Renewal Quote. If you set it to *true*, Renewal Opportunity and Renewal Quote is not created for one time. It is recommended that you treat one-time assets purely as one-time sales. Any modifications or renewals should not be encouraged for one-time assets.
Allow Mass Change	You can enable or disable mass changes for *must configure* assets with the help of this setting.
Create Renewal Opportunity	Indicates if you want to create renewal opportunities along with renewal quotes. By default, this check box is not selected. This is to prevent the system from running into locking issues when an account has a large number of assets and the batch process creates the renewal opportunities for all these assets. When set to true, the Create Renewal Opportunity scheduled job runs and creates the renewal opportunities along with renewal quotes. When set to false, the Create Renewal Opportunity scheduled job runs and does not create the renewal opportunities and creates only the renewal quote.

Setting	Description
Renew One Ramp	Consider a scenario where multiple ramped assets exist for a quote. In the previous releases, renewing any one of the ramped assets results in the creation of a renewal cart with the same number of ramps as the original proposal. Each ramp line is created from the original deal which implies that the uplift has to be defined from Ramp 1 for accurate prices in the subsequent ramps. Redefining and specifying details for each ramp can be cumbersome and as a user, you would expect that for a three-year ramped deal, the uplift specified on year 3 is applied on renewal to year 4. This setting enables you to renew only one ramp line item based on the new term in order of renewal term in the asset line item, default renewal term specified in the Installed Product Setting and Selling Term defined in the asset line item.
Account Hierarchy Batch Size	Indicates the number of Account Hierarchy records processed for each batch execution. Recommended value is 500. Valid values range from 1 to 2000.
COTERMINATION PREFERENCES	
End Date preferences for Cotermination	Enables you to decide what End Date preferences the user should see while coterminating incremental licenses using the **Change:Quantity** action. For information on coterminating assets, see Managing Asset Increment with Coterminate Lines. The supported values are: • RetainCurrentAssetEndDate: This option enables the user to retain the current asset end date for cotermination. • UseCustomDate: This option enables the user to enter a custom date for cotermination. • UseProposalEndDate: This option enables the user to use the proposal end date for cotermination. ⓘ If the user performs asset increment through the Asset Manager flow, the **Use the Proposal End Date** option is not displayed on the Change Quantity intermediate page.
Default Cotermination Option	From the **Default Cotermination Option** drop-down list, select the cotermination option that you want to set as default.

Setting	Description
RELATE CRITERIA	
Relate Action Criteria Fields	Select the asset line item fields to enable relate asset action. This setting shows all fields of Asset Line Item and Price Dimensions defined on asset line item. When the user selects two or more assets and clicks **Relate**, CPQ validates the selected asset line items against the Relate Action Criteria Fields. In case of any discrepancies, CPQ displays a validation error message to the user.
ASSET SOURCE	
Asset Source	Allows you to filter the assets to be displayed on the Installed Products page based on the **Ship** To or **Sold To** or **Bill To** fields of an account. By default, all the assets filtered with the **Sold To** field are displayed. Specify the API name of the Account field that you want to use as a source for displaying assets. For example, to filter the assets based on **Ship To** field, enter Apttus_Config2__ShipToAccountId__c. You can specify all the three account fields and the system considers an OR condition between these fields while filtering. This is useful when an account has different locations for shipping and billing and you want to display only those assets that have been shipped to a certain location.
PURCHASE ID CRITERIA	

Setting	Description
Purchase Id Criteria	Allows user to define a criteria to get coterminate related asset lines in the Coternimate with Increment section on the Change Quantity page during asset increment. The default purchase identification criteria are: Product, Charge Type, Option, and Account. From the **Available Purchase Identification Criteria** list, add the required criteria to the **Selected Purchase Identification Criteria** list. CPQ currently supports only Purchase Identifier and alphanumeric custom fields. For example, if you select Purchase Identifier, during increment of a selected asset, all other assets with the same Purchase Identifier are displayed on the Coternimate with Increment section. ⚠ If you add *Quantity*, *Selling Term*, or *End Date* to the **Selected Purchase Identification Criteria**, the user will receive an error on order creation or activation. Based on the selected purchase identification criteria, the total quantity of an asset spans multiple streams during the increment scenarios, on the Installed Products page (of a renewal quote).
SPLIT ASSET ACTIONS	
Split Asset Actions	Enables the user to perform split asset actions on the Define Split <asset name> page. Add the required split asset action from the **Available Split Asset Actions** to **Selected Split Asset Actions** list. Supported options are: • **Renew**: Enables users to perform the Split and Renew asset action. • **Swap**: Enables users to perform the Split and Swap asset action.
ASSET VIEWS	
You can create a view and share it with selected users to be displayed as a view on the Assets grid. Users can only select this view, but they cannot edit or delete it. Only you can edit or delete the view as an administrator.	
Flow	Select a flow for the view.
View	Select **New** to create a view.

Setting	Description
Name	Enter a name for the view.
Default View	Turn on this button to make this view as the default view.
Make Public	Turn on this button to make this view public.
User Types	Select one of the following user types: • User Profiles • User Roles • Users After selecting the user type, move the respective user from the **Available** list to the **Selected** list.
Field Name	Add the fields to be displayed as columns in the view for users on the Assets grid.
ASSET TERMINATION FIELDS	
Asset Termination Fields	Do not use this setting. Instead use the *Asset Termination* option from the **Display Type** picklist. For more information, see Configuring Display Columns Settings.
MERGE ACTION CRITERIA	
Merge Action Criteria Fields	Select the fields that must be common among the asset line items to be merged. The default fields are product, charge type, selling term, and price type.
EDITABLE FIELDS FOR CANCELLED LINES	

Setting	Description
Editable Fields for Cancelled Lines	Select the fields that you want to make editable for cancelled lines on the cart (during asset termination). You can select a custom currency field that is not based on a formula. All out-of-the-box currency fields such as base price and base extended price are not visible in the list of editable fields. Sensitive managed package fields that the user cannot edit on a cancelled line item are not visible in the multi-select picklist to the user. You can allow Sales Representatives to edit custom asset line item field values on the Confirm Termination page instead of waiting to go to the Cart page to edit these values. For example, they may want to capture the reason for terminating assets. CPQ carries these edited field values from the Confirm Termination page to the Cart page where Sales Representatives can make further changes to those field values, if required. ⓘ You must add the same fields to the *Asset Termination* **Display Type** of a specific flow in **Display Colum Settings**. For more information, see Configuring Display Columns Settings.

Configuring Collaboration Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon (⋮) on the top-right corner and select **Application Management**.
3. Click **Collaboration**.
4. Configure the following details.

Setting	Description
PARENT LINE ITEM COLUMNS	
Flow	Select a flow and enter the following details. • Field Name • Is Sortable • Style • Style Class • Header Style

Setting	Description
CHILD LINE ITEM COLUMNS	
Flow	Select a flow and enter the following details. • Field Name • Is Sortable • Style • Style Class • Header Style

Configuring Pricing Engine Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon (⋮) on the top-right corner and select **Application Management**.
3. Click **Pricing Engine**.
4. Configure the following details.

Setting	Description
GLOBAL PRICING	
Enable External Pricing	Indicates whether pricing will be done externally. The default value is false (disabled).
Defer Pricing	Performs pricing of products only after you click the Go to Pricing button; whereas, clearing this check box enables the system to perform pricing as and when you add or delete products. The Sales rep can use the **Update Price** button on the Configuration page to calculate pricing after updating fields. CPQ allows you to selectively turn on or off Defer Pricing at a flow level. You must set the **deferPricingUntilCart** parameter to True in the URL that you have configured on the **Configure Products** field of the respective quote. For more information, see Creating Custom Buttons for Different Flows.

Setting	Description
Pricing Profile	Type Basic, Advanced, or External. A pricing profile is Basic, when there are none of the following used: • Pricing Rules • Price Matrices • Related Pricing Setup • Bundles Note: If the Pricing Profile field is left blank, the default value is Advanced. ⓘ You can configure this setting based on a flow. In such case, the flow-based setting overrides the generic setting.
TOTALING AND SUBTOTALING ⓘ CPQ allows you to selectively skip totaling at a flow level. You must set the **isCartTotalingDisabled** parameter to True in the URL that you have configured on the **Configure Products** field of the respective quote. For more information, see Creating Custom Buttons for Different Flows.	
Totaling Group Type	Type the default value for summary group type picklist. Valid values are: • *Category*: When you add a product to the cart from the lowest leaf category where it is associated, the Category Hierarchy column in the **Line Items** related list is populated with its bread-crumb trail. This is useful to identify the exact category from where the product was added, in case if the product is associated with multiple categories. • *Product* The default value is *Category*. For Product-based totaling, the Product Totaling Hierarchy setting is required.
Adhoc/Product Totaling Hierarchy	Enter the list of product fields that represent the product totaling hierarchy. Each product field name must be separated by a new line. For Ad-hoc totaling, enter the line item fields that represent a totaling group. If empty, then CPQ uses Bundle Family by default.

Setting	Description
Disable Charge Type Totaling	Disables the totaling of charge types.
Compute Totals In Separate Step	Indicates whether the totaling should be performed in a separate step or it should be combined with the base pricing step. This setting ensures that the totaling, which is dependent on the number of lines, should be done as a separate remoting call. This setting helps to avoid the CPU time limit issue to a large extent.
ADVANCED PRICING	
Enable Aggregate Pricing	Indicates whether aggregate pricing is enabled. This may be used to optimize performance when aggregate price rules are not used in the organization. The default value is false (disabled).
Enable Matrix Pricing For Options	Indicates whether matrix pricing is enabled for options. This may be used to optimize performance when options do not use price matrices in the organization. The default value is false (disabled).
Max Adjustment Lines	Type the maximum number of discount lines allowed for a line item. The default value is one. You can create more than one adjustment only when the value is greater than one.
Enable Keyed Matrix Pricing	Select this if you have option products with large number of dimensions of type discrete. This also helps in reducing SOQL queries.
Enable Auto Reprice	Automatically reprices the cart. In the cart, when you modify pricing and tab out, the system automatically recalculates and displays the price.

Configuring Proposal Settings

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Application Management**.
3. Click **Proposal Settings**.
4. Configure the following details.

Setting	Description
DEFAULT	
Default Opportunity Quote Owner	Type the owner name to set the default owner for the quote/proposal created from an opportunity. Valid values are: • Opportunity Owner • Current User If not set, Opportunity Owner becomes the owner of the new quote/proposal.
Disable Opportunity Products Copy	Disables copying of opportunity products to the quote/proposal.
Default Quote Agreement Owner	Type *Quote Owner* or *Current User* to set the default owner for the agreement created from a quote/proposal. If not set, *Quote Owner* becomes the default owner of the new agreement.
Email Template For Presenting Proposals	Type the name of the email template to use for presenting proposals.
SYNCING	
Auto Sync With Opportunity	Automatically synchronizes the quote line items with the opportunity, when the proposal is finalized. This setting appears only if you have Conga Quote Configuration Integration package installed.
Sync Option Products	Synchronizes options along with bundles when you click Sync with Opportunity for your proposal. The default is to synchronize bundle only. This setting appears only if you have Conga Quote Configuration Integration package installed.
Sync Bundle Using Line Items	Synchronizes bundle products using proposal line items. The default uses proposal summary objects. This setting appears only if you have Conga Quote Configuration Integration package installed.
MERGE SERVICE	

Setting	Description
Instance Url	Type the Salesforce instance URL. For example: https://na7.salesforce.com This is required to navigate to custom pages in the managed package.
Merge Webservice Endpoint	Type the Conga merge webservice endpoint.
Merge Call Timeout Millis	Type a number to indicate timeout in milliseconds for the merge request. For example: 60,000
Max Child Level	Type the maximum level to generate the merge data for the proposal.
ESIGN	
Auto Select Attachment?	Select this setting to enable CPQ to select the most recent attachment automatically.
Auto-Select multiple recipients?	Select this setting to enable projects to specify a list of recipients that will be defaulted for eSignature. To use this option, projects will add the **DocuSign Default Recipients** related list on proposals and user will select the list for each proposal.
Auto Select Recipient Field Name	Enter the API name of the field which holds the value for the recipient to select automatically.
SECURITY	
PDF Owner Password	Type the password required to change permissions of the PDF document like printing or editing.
Enable PDF Security	Enabling PDF security lets users apply security settings to PDF documents and protect them with a password.

Configuring Custom Settings

This is the standard Salesforce Custom Settings page that lists all the legacy custom settings you may require to fulfill your specific business requirements. You can access these settings from **Setup > App Setup > Develop > Custom Settings.** Click **Manage** next to the setting you want to configure. In the case of Config System Properties and Installed Product Settings, you can configure certain fields differently for different flows, refer to Configuring Custom Settings for Different Flows.

Custom settings are executed in an order of precedence. For more information, refer to About Custom Setting Order of Precedence.

After you define or modify any custom setting, execute Custom Settings Maintenance Job.

> You must execute Custom Settings Maintenance Job after you update any Custom Settings. The changes you make are only effective if you execute the maintenance job. Otherwise, all the computations are processed with stale values. Refer to the Running Maintenance Jobs topic.

Some of the custom settings listed in this section may require you to contact Conga Support or Professional Services to guide you through using these settings for specific business use cases.

In your org, you may see custom settings that are not listed in this document—these custom settings are not functional and are marked as Deprecated (D).

- Base Private Properties
- Comply System Properties
- Config Asset Pricing Criteria Fields
- Config Asset Pricing Default
- Config Custom Classes
- Config Custom Display Actions
- Config Custom Display Columns
- Config Data Cache
- Config Field Set
- Config Flow
- Config Select Options Settings
- Config Select Products Settings
- Config System Properties
- Config User Preferences
- Installed Products Settings
- Lookup Field Settings

- Proposal System Properties

Base Private Properties

This custom setting holds the list of all the base library properties.

Setting	Description
Enable Usage Data	Select this to enable data usage.
Usage Data Endpoint	Enter the usage data endpoint.
Usage Data Key	Enter the usage data key
Usage Data Provider Class	Enter a list of comma-separated class names of Usage Data Providers specifically the class that implements Apttus_Base2.IUsageData interface.
Usage Event Key	Enter the usage event key.

Comply System Properties

This custom setting holds the lists all the Comply system properties. You can enter one or more of the following property details.

Setting	Description
Admin User	The admin user is the default owner of activities created by a user who is not allowed to be the owner (for example, customer portal user). Enter the admin user name as first name, last name.
Auto Enable PDF For Final Docs	Select this and the **Create PDF Attachment** check box is always selected when you choose to save as *Final - to be signed* from the check-in dialog. You may use this field when you want to finalize an agreement document.
Auto Enable Reconciliation	Select this and the *Reconcile Document* option is always selected when you go to check-in an agreement document.

Setting	Description
Agreement Number Field For Imported Docs	In this field, specify the API name of the field you want to use. For example, use`Apttus__Agreement_Number__c`. When a new document is imported into the system, it will include the agreement number in the top right corner of the header on each page, using the field selected above.For more information about when to use this field, see **Agreement Number/Header Configuration** section in Configuring Offline Agreement Window.
Allow PDF Select Override	This is only applicable when **Auto Enable PDF For Final Docs** is selected.
Allow Reconcile Selection Override	This is only applicable when **Auto Enable Reconciliation** is selected.
Auto Create Order	Select this check box if you want to create an order and asset as soon as an agreement is activated. This is useful when you want to integrate Assets with Contracts.
Auto Enable Private Indicator	Select this check box if you want to auto enable private indicator for documents. If you select this check box, the **Make this document private** check box for any agreement document is auto-selected.
Auto Insert Header Footer Data	Select this field to automatically insert the Agreement Number Field For Imported Docs field value to the header and the latest timestamp to the footer of an agreement document. This field is available for Generate, Import and Offline actions. For more information about when to use this field, see **Agreement Number/Header Configuration** section in Configuring Offline Agreement Window.
Allow Private Selection Override	Select this check box if you want to allow the user to override the private document selection. This setting is applicable only when **Auto Enable Private Indicator** is selected for documents.

Setting	Description
Auto Sync With Opportunity	Indicates whether the agreement will be automatically synchronized with the opportunity when the agreement is accepted. This setting appears only if you have Apttus Contract-Configuration Integration package installed.
Enable PDF Security	Enabling PDF security lets users apply security settings to PDF documents and protect them with a password.
Bypass Sharing	Indicates whether apex code can bypass record sharing during selective operations such as clone and deleting draft attachments.
Contract Summary Template	This field contains the name of the contract summary template. You may have to define a separate template to contain the contract summary details and mention the name of the template here.
Default Opportunity Agreement Owner	Type *Opportunity Owner* or *Current User* to set the default owner for the quote/proposal created from an opportunity. If not set, *Opportunity Owner* becomes the default owner.
Default Document Tags	Enter a comma-separated list of tags to make available for any agreement document as it is checked in from X-Author Contracts. Use of these tags requires **Contract Document Versioning** to be enabled. The **Document Finder** feature must also be configured in your org. For more information, see Configuring Document Finder.

Setting	Description
Document Naming Convention	Specify a value to apply a custom naming convention for all proposal documents at generation, check-in and signature events. The default naming convention is %:Name%_%action%_%templatename%_%timestamp% If this property is left blank, the system will use the default naming convention. The following attributes permitted when formulating a document naming convention are: • %checkintype% • %action% • %templatename% • %user% • %timestamp% • Proposal attributes such as %:Name%. Note: any variable prefixed by ':' represents a field on the Proposal object. Example proposal document name using the default naming convention: *SOW_Regenerated_SOW ABC_2015-08-07*
Document Structure FX2 For Imported Docs	Check this box to make document structure FX2 format for all Offline documents (created or imported). If not checked, all offline documents are created in the "pre-existing" format. ⓘ This property only applies to Offline agreements created from the Agreement or user Home page links in Salesforce. The format of Offline agreements created using X-Author for Contracts will still depend on user-input from X-Author.
Email Template For Checkin Notification	Type the email template for sending check-in notifications.
Enable Clause Approvals	This check box allows you to set up approval processes on clauses used in your document.

Setting	Description
Enable Document Versioning	Check this box to enable Document Versioning. ***All new agreement records created in your org will use Document Versioning after this setting is enabled***. Enable Version Control must be enabled for Document Versioning to work properly. Enabling Document Versioning changes the value of the **Version Aware** Agreement field to TRUE for all new agreements after the property is activated. The **Version Aware** field is a flag that tells Apttus Contract Management to use Document Versioning for a specific record. **Important Note**: It is recommended that once a record is flagged as Version Aware, you do not disable this field, as versioning will become undefined for the agreement record in question. Instead, ensure that records which should not use Document Versioning do not have the field enabled when they are created. For more information on how you can configure Document Versioning for your org, see Enabling Contract Document Versioning.
Enable Merge Call Debug	Enable Merge Service debugging.
Enable Multiple Checkout	Allows multiple checkout. This is applicable only when version control is in effect.
Enable PDF Security	Enabling PDF security lets users apply security settings to PDF documents and protect them with a password. See Enabling PDF Security for Quote/Proposal Documents for more information.
Enable Submit Merge Call	Submits merge calls for processing. The **Submit** and **Submit for Generate (Async)** button is displayed during document generation.

Setting	Description
Enable Template Versioning	Check this box to enable Template Versioning. *All new Templates created in your org will use Template Versioning after this setting is enabled.* For more information, see Configuring Template Versioning.
Enable Version Control	Check this box to enforce a check-in/check-out policy for agreement documents. This setting must be enabled when "Enable Document Versioning" is checked. Note: This property existed in CLM versions prior to 8, so this property may already be enabled.
Footer Datetime Format For Imported Docs	Specify the format in which date and time will be shown in the generated agreement. When a new document is imported into the system, it will include the Date in the bottom left corner of the footer on each page, in the format selected above. For more information, see **Setting the Date and Time Format for the Footer** section in Configuring Offline Agreement Window.
Instance URL	The Salesforce instance url for redirecting to custom pages. For example: https://na7.salesforce.com This is required to navigate to custom pages in the managed package.
Max Child Level	The maximum level of lookups and child objects available from the parent object to generate the merge data for. For example, **Agreement** is the parent object, **Agreement Fee** is the child object of **Agreement** and **Agreement Fee Adjustment** is the child of **Agreement Fee** and grandchild of **Agreement**.
Merge Call Timeout Millis	The timeout in milliseconds for the merge request. Type a number to indicate timeout in milliseconds for the merge request. For example: 60,000
Merge Webservice Endpoint	Type the Apttus merge webservice endpoint. For example: https://mergews.apttus.net/cgi-bin/MergeServer/Bin/mmcgi.exe
PDF Owner Password	Type the password required to change permissions of the PDF document like printing or editing.

Setting	Description
Publish Author Events	Indicates whether merge events that occur in X-Author, such as check-in or check-out, should be published. If enabled, a record is inserted in the Merge Event table on the Agreement record.
Publish Merge Events	Indicates whether merge events, such as generating an agreement document or creating offline document should be published. If enabled, a record is inserted in the Merge Event table on the Agreement record.
Sync Bundle Using Line Items	Synchronizes bundle products using agreement line items. The default uses agreement summary objects. This setting appears only if you have the Apttus Contract-Configuration Integration package installed.
Sync Option Products	Synchronizes options along with bundles. The default is to synchronize bundle only. This setting appears only if you have the Apttus Contract-Configuration Integration package installed.
Temp Email Template Inactive Hours	Use this property to set the orphan period of temporary email templates to use in conjunction with the **CleanupJobScheduler** APEX class. The default value of this property is set to 4 hours. For more information, see CLM for Administrators.
Use Agreement Locks for Versioning	Check this box to use Agreement locks for versioning instead of document-level locking: • If enabled, all agreement documents are locked (by the user checking out) when ***any one of them*** is checked out. • If disabled, only checked out documents are locked (by the user checking out)–any other agreement documents which have not yet been checked out can be checked out for editing.

Config Asset Pricing Criteria Fields

This custom setting holds the list of API names of fields from Asset Line Item object used in setting Asset Pricing Indicator. Criteria with a specific Line Status can also be defined. You can create multiple data sets as well.

Setting	Description
Criteria Field Names	Enter the list of API names of fields from the line item object used in setting the asset pricing indicator. Each field must be separated by a comma or a new line.
Line Status	Enter the list of line statuses to make the criteria valid only for the line item with line status mentioned in this field. Examples of valid values are *Renewed, Amended.* If you want to define a Line Status specific criteria, you must define two criteria: one with Line Status and another with blank Line Status for the same criteria fields.

Config Asset Pricing Default

This custom setting holds the default value for the asset pricing indicator associated with a line status. The value defined in the setting is only applicable on the line status defined. Create different data sets for different line statuses. Click **Manage** then click **New** to create a new record. Specify the exact value of line status in **Name**. Examples of valid values are *Renewed, Amended.* The list of valid line status values is defined in the Line Item object. Define the field mentioned in the table as desired and click **Save**.

Setting	Description
Default Asset Pricing Indicator?	Select this checkbox to enable asset pricing by default for Asset Related Line items associated with the given line status.

Config Custom Classes

This custom setting registers all custom classes that implement Callback classes for customization. Click **Manage** and click **Edit** next to *Config Custom Classes*. Enter API names of the custom classes that you created next to the Callback class you want to implement.

The following table lists the Callback classes and the global interfaces the Callback class implements.

Callback Class	Interface
Action Callback Class	CustomClass.IActionCallback3
Action Invoker Callback Class	CustomClass.IActionInvokerCallback
Action Params Callback Class	CustomClass.IActionParamsCallback
Adjustment Line Item Callback Class	CustomClass.IAdjustmentLineItemCallback
Adjustment Spread Callback	CustomClass.IAdjustmentSpreadCallback
Advanced Approval Callback Class	CustomClass.IAdvancedApprovalCallback
Asset Line Item Callback Class	Class CustomClass.IAssetLineItemCallback3
Asset Renewal Custom Callback Class	CustomClass.IAssetRenewalCustomCallback
Bulk Loyalty Point Callback Class	CustomClass.IBulkLoyaltyPointCallback
Cart Approval Callback Class	CustomClass.ICartApprovalCallback2
Deal Optimizer Callback Class	CustomClass.IDealOptimizerCallback2
Display Action Callback Class	CustomClass.IDisplayActionCallback
Formula Callback Class	CustomClass.IFormulaCallback ⓘ This callback has been deprecated.
Loyalty Cycle Callback Class	CustomClass.ILoyaltyCycleCallback
Loyalty Point Callback Class	CustomClass.ILoyaltyPointCallback

Callback Class	Interface
Metadata Callback Class	CustomClass.IMetadataCallback. This callback has been deprecated.
Option Filter Callback Class	CustomClass.IOptionFilterCallback
Pricing Callback Class	CustomClass.IPricingCallback2
Pricing Extension Callback Class	CustomClass.IPricingExtensionCallback
Product Attribute Callback Class	CustomClass.IProductAttributeCallback
Product Filter Callback Class	CustomClass.IProductFilterCallback
Recommendation Callback Class	CustomClass.IRecommendationCallback
Related Pricing Callback Class	CustomClass.IRelatedPricingCallback
Revalidation Callback Class	CustomClass.IRevalidationCallback
Shipping Callback Class	CustomClass.IShippingCallback
Tax Callback Class	CustomClass.ITaxCallback
Validation Callback Class	CustomClass.IValidationCallback

Config Custom Display Actions

This custom setting holds information about custom actions displayed in, for example, Installed Products view, Cart view.

Setting	Description
Action Area	Enter a value to select the action display area. Valid values are Left, Right, Center, and More. The value *More* indicates that the action is displayed under More button.
Action Label Name	Enter the name of the custom label for the action.

Setting	Description
Action Name	Enter the API name for the action.
Action Page	Enter the name of the custom action page. You can implement a custom action either as a Visualforce page or as a custom class.
Action Params	Enter the URL parameters for the action. You can pass multiple parameters by using the ampersand character to separate each parameter. Each parameter is expressed in the form param-name=param-value. For example, templateName=XYZ&draft=true
Action Style Class	Enter the style class name for the action.
Action Type	Enter the action type.
Always Display	Select this setting to display the action (grayed out) even when it is not enabled.
Behavior	Enter the behavior that the open page action takes.
Display As	Enter the action, task, or both to display the action on a page as action button or task.
Display Type	Enter the action display type. Valid values are: • Cart Line Item • Asset Line Item
Flow	Enter the flow associated with the action.
Is Enabled	Select this setting to enable the action for the display.
Sequence	Enter a number for the action display sequence.

Config Custom Display Columns

This custom setting holds information about fields displayed in, for example, Installed Products view, Cart view.

Setting	Description
Display Type	Enter the column display type. Valid values are: • Cart LineItem • Installed Product
Field Name	Enter the field API name for the column.
Flow	Enter the flow associated with the column.
Header Style	Enter the style used to display the header of a column, which is used primarily for adding inline CSS styles.
Is Editable	Select this setting to enable column value to be edited.
Is Price Point	Select this setting to determine type of a line item for use in Cost & Profitability Price Waterfall. If you select the setting, item is a Price Point. If you deselect the setting, item is an Adjustment.
Is Sortable	Select this setting to enable column sorting.
Sequence	Enter a number for the column display sequence.
Style	Enter the style used to display the column, which is used primarily for adding inline CSS styles.
Style Class	Enter the style classes that must be applied to dynamic columns.

Config Data Cache

Config Data Cache is used to store the list of line item fields that are used by various custom settings, like **Config Display Columns**, **View Cart Custom Fields**, and **Constraint Rule Criteria**. These fields are used in line item queries constructed for CPQ's internal use. This optimizes the queries to fetch specific fields instead of fetching all the fields. The administrator must not update this custom setting manually.

Setting	Description
Data	The delimited data.
Is Last	The last value indicator

Config Field Set

Config Field Set is used to store list of fields for special purpose.

Setting	Description
Data	Enter delimited data.
Is Last	Enter the last value indicator

Config Flow

This custom setting allows you to define a user interface flow for CPQ. You can use default pages available to you or create custom pages. Click **Manage** then click **New** to create a new flow. Define the field mentioned in the table as desired and click **Save**.

Setting	Description
Assets Page	Enter the name of the page you want to display as Asset page. Specify one of the following or create a custom page. • SelectInstalledProducts • SelectInstalledProductsEn
Cart Page	Enter the name of the page you want to display as Cart page. Specify one of the following or create a custom page. • CartView • CartDetailView

Setting	Description
Catalog Page	Enter the name of the page you want to display as Catalog page. Specify one of the following or create a custom page. • SelectConfigProducts • SelectConfigProductsFilterView
Option Page	Enter the name of the page you want to display as Option page. Specify one of the following or create a custom page. • ConfigureBundle • SelectConfigOptions • SelectConfigOptionsTabView • SelectConfigOptionsListView • SelectConfigOptionsTreeView • SelectConfigOptionsDetailView • SelectBundleOptions
Product Attribute Detail Page	Enter the name of the page you want to display as Product Attribute Detail page. Specify one of the following or create a custom page. • ProductAttributeDetail3

Config Select Options Settings

Configuration settings for option selection pages. The record should be named "Primary Settings".

Setting	Description
Bundle Description Field	Enter the field names from the product object to be displayed as bundle description.
Enable Option Page Search	Select this setting to allow option search on the bundle configuration page. This setting enables a search bar on the Bundle Configuration page. The setting allows the user to search for options, attributes, and sub-bundles on the Configuration page.

Setting	Description
Hide Disabled Options	Select this setting to hide the options that are disabled by the exclusion rules. Such options will not be shown in the list of available options. Also, if all such disable options are under one option group, this group is also hidden on the configuration page.
Hide Price Column	Select this setting to hide the price information column from the options page. Be careful in hiding this critical information.
Listed Options Column 2	Enter the API name of the product field to be displayed in the listed options section. A valid value could be like ProductCode.
Listed Options Column 3	Enter the API name of the product field to be displayed in the listed options section. A valid value could be like ProductCode.
Main Section Ratio	Enter the ratio to divide the options page. The ratio of the main sections in the page must be separated by a colon. The default value is 0:70:30. This is applicable to the SelectConfigOptionsDetail and ConfigureBundle pages.
Max Inline Option Attributes	Specify the maximum number of inline option attributes to be displayed in the attribute group. The default value is 3.
Option Field In Summary	Enter the Option field to be displayed on right hand side of configuration page.
Show Bundle Detail	Select this setting to show bundle price and quantity fields on options pages.
Show Option Quantity	Select this setting to display the Quantity in the body of the options pages. This setting controls the SelectConfigOptionsDetailView and ConfigureBundle page.

Config Select Products Settings

Configuration settings for product selection pages. The record should be named as "Primary Settings".

Setting	Description
Always Display Leaf Products	Select this setting to load and display all leaf nodes for a specific category. These leaf nodes have products associated with them. With this the setting, the user needs to just click the main category or any subcategory to load all sub-categories (and products) at one click.
Cache All Products	Select this setting to cache all products to avoid SOQL queries. This setting must be enabled when the number of product records is small. This is a Beta feature.
Carousel Category Level	Enter a number for the maximum depth allowed to be browsed in the Product Carousel. No value in this field allows infinite browse depth. A value of zero will restrict you to the root level.
Carousel Default Icon	Enter the Default Icon to be used for the categories in the Carousel.
Cart List Item Description Field	Enter the API field name of the **Long Description** field to be used in the Cart Detail View page.
Collapse All Leaf Option Groups	Select this setting to collapse all leaf option groups on the Config Options Detail View page.
Custom Action Label Name	Enter the name of the custom label for the action.
Custom Action Page	Enter the name of the custom action page.
Disable Add Another	Select this setting to disable the user from adding multiple instances of the same product.

Setting	Description
Disable Add To Cart for Option Products	Select this setting to hide **Add to Cart** and **Configure** buttons for the option products on catalog pages. This is a global setting to hide these buttons.
Disable Add To Cart On Click	Select this setting to disable the **Add To Cart** button on click until the server confirms the addition of the product.
Disable Favorites	Select this setting to disable saving the favorite, favorites category, and other favorites-related UI items.
Enable Category Filter	Filters the categories based on search results and displays them in the Narrow Your Search area on the left. For example, if the search result has products from only two categories, the Narrow Your Search section displays only those two categories. You should always select **Enable Category Filter** when **Hide Narrow Your Search** is cleared.

Setting	Description
Favorites Display Label	Enter a name for the group of favorite configuration records. This label is displayed as a category name on the catalog page. If you do not specify any name, the system gives *Favorites* as the default name. **Deprecated setting** Favorites Display Label Settings have been deprecated. Now you need to rely on overriding custom labels for renaming 'Favorites' with your choices such as Quick Buys, Most Used, and so on etc. For example, if you want to use the term "Quick Buy" for favorites, CPQ contains two custom labels whose values you must override. Go to Setup > Custom Labels. • Favorites > Click "New Local Translations/ Overrides" > Choose Language as "English" > Write "Quick Buys" in translated value area. • SaveAsFavorite > Click "New Local Translations/Overrides" > Choose Language as "English" > Write "Save as Quick Buy" in translated value area.
Favorites Icon Attachment Id	Enter the ID to the attachment of the Favorites icon.
Favorites Upload Image	You can choose an image to upload for your group of favorite configuration categories. This image is displayed under your favorites (as entered in the **Favorites Display Label**) category on the catalog page. By default, no image is set
Hide Breadcrumb	Hides the breadcrumb navigation element.
Hide Cart Header	Select this to hide the Cart Header for selected pages.
Hide Compare Products	Disables the Compare Products feature in the catalog page.

Setting	Description
Hide Configure Action	Hides the Configure action for a Product on the catalog page.
Hide Default Options In Description	Hides the default options listed below the product description.
Hide Help Me Choose	Indicates whether to hide the Guided Selling link. By default, this is not selected and the system shows Guided Selling link on the Catalog page for the respective categories.
Hide Line Item Attribute Details	Hides the Line Item Details section on the Product Attribute Detail page.
Hide Listed Products Price Column	Hides the price information column in the listed product section. By default, this is not selected.
Hide Narrow Your Search	Hides the Narrow Your Search section that displays the category tree on the left of the catalog page. This setting is deprecated for the New UI.
Hide Options in Quick Add Mode	Select this to hide option products in Quick Add mode
Hide Product Image	Hides the product image from the catalog page.
Hide Selected Product Charge Types	Select this to hide charge types for selected products from the tree view options page.
Hide Selected Products Column1	Hides the column that displays the product name.
Hide Single Top Category	Hides the top category. This property is useful only when you have a single root category and want to have a cleaner look for all of its child categories, without the root category.

Setting	Description
Listed Favorite Configurations Column1	Enter the API name of the favorite configuration field that you want to display on the catalog page for your favorite configuration records. You can enter a maximum of one API name. If you have a requirement of displaying one more field, use **Listed Favorite Configurations Column 2** custom setting.
Listed Favorite Configurations Column2	Enter the API name of the favorite configuration field that you want to display on the catalog page for your favorite configuration records. You can enter a maximum of one API name.
Listed Products Column2	(optional) Type the API name of the product field to display in the listed products section. Example Value: ProductCode
Listed Products Column3	(optional) Type the API name of the product field to display in the listed products section. Example Value: ProductCode
Listed Products Default Icon	Enter the ID to the attachment of the default Product Icon used for the Listed Products.
Main Section Ratio	Type the ratio in which you want to divide the page. The ratio of the main sections in the page should be separated by a colon. The default value is 20:60:20. This is applicable to the *SelectConfigOptionsDetail*, *ConfigureBundle*, and *SelectBundleOptions* page. Note: Do not use the *SelectBundleOptions* page from your drop-down list, it is no longer supported.
Minimize Cart	Displays the cart icon and hides the cart sidebar on the right-hand side of the catalog page.
Order Status Fields	Type the name of the line item fields (separated by comma) that contains information about promotion, delivery info, shipping info, and more.
Product Cache Fields1	Enter the first list of custom fields required for the product cache.

Setting	Description
Product Cache Fields2	Enter the second list of custom fields required for the product cache.
Product Cache Fields3	Enter the third list of custom fields required for product cache.
Product Cache Fields4	Enter the fourth list of custom fields required for product cache.
Read Only Location	Makes the Location field read-only.
Save as Favorite Dialog Columns	Enter comma-separated API names of the columns that you want to display on the **Save As My Favorite** dialog box on the cart.
Search Query Limit	Type a value to set the limit for a search query. For faster product search, type a smaller number. The default is 1000.
Selected Products Column2	(optional) Type the API name of the line item field to display in the selected products section. Example Value: ProductId__r.ProductCode
Selected Products Column3	(optional) Type the API name of the line item field to display in the selected products section. Example Value: ProductId__r.ProductCode
Selected Products Column4	(optional) Type the API name of the line item field to display in the selected products section. Example Value: ProductId__r.ProductCode
Show Checkbox In Rule Dialog	Enables check box to select multiple products in the Inclusion Constraint Rules dialog at once, instead of adding the products individually using Add to Cart. This setting works for *Inclusion Rule* with Min/Max Match Rule as *Include Min/Max*.
Show Favorite as First Category	Select this setting to display the favorites category on top in the list of product categories.

Setting	Description
Show Product Icon Cart Detail View	Displays the product icon on each line item in the Cart Detail View page. Note: This will disable the action item columns and show the actions under the product icon.
Show Quantity Input	Displays the Quantity field on the catalog page.
Show Recommended Products Cart View	Displays the recommended products component in the Cart Detail View page.
Show Selected Product All Charges	Displays all the charge types for selected products in the tree view options page.
Show Selected Products in Config Options	Displays the Selected Products widget in the Config Options Detail View page.

Config System Properties

This custom setting holds the Configuration & Pricing System properties. The record *System Properties* should already exist. If not, create a record named *System Properties.* Click **Manage** then click **New** to create a new record. Define the fields mentioned in the table as desired and click **Save**.

Setting	Description
Actions Column Position	Enter the position of the actions column in the cart page. Valid values are: • Left • Right
Adhoc/Product Totaling Hierarchy	Enter the list of product fields that represent the product totaling hierarchy. Each product field API name should be separated by a new line. Only applicable when product based totaling is in effect. Select the custom setting **Totaling Group Type**.

Setting	Description
Admin User	The admin user is the default owner of activities created by a user who is not allowed to be the owner, for example, Customer Portal user. Enter the admin user name as <first name>, <last name>.
Auto Execute Post-Pricing Step	Select this setting to enable CPQ to execute the post-pricing logic automatically after pricing. The setting is applicable only after the pricing logic is defined in the pricing extension callback.
Auto Execute Pre-Pricing Step	Select this setting to enable CPQ to execute the pre-pricing logic automatically before pricing. The setting is applicable only before the pricing logic is defined in the pricing extension callback.
Auto Finalize On Cart Approval	Select this setting to automatically finalize the cart when it is approved and ready for finalization.
Auto Refresh Usage Tier	Select this setting to enable modifiable usage tiers to automatically re-calculate tier price upon PLI change.
Auto Sync On Cart Approval	Select this setting to automatically synchronize the cart when it is approved and ready for finalization.
Auto Update Category View	Select this setting to enable auto incremental updates to category view.
Base Product Relation Field	Enter the API name of the product field that associates a child product with the base product.
Bypass Sharing	Select this setting to enable apex code to bypass record sharing.
Bypass Shopping Cart	Select this setting to display the **Update Price** and **Finalize** buttons and hide the **Go To Pricing** button on all the new pages. On the configuration page, only the **Go To Pricing** button is disabled. You can still finalize the quote using the Mini cart.

Setting	Description
Cart Edit Access Idle Timeout in Minutes	Enter a number to specify the number of minutes of idle time a configuration can be kept in edit mode.
Cart Theme	Cart Themes are deprecated. No changes are executed even if the values are defined in the setting.
Cascade Shared Attribute Updates	Select this setting to cascade the shared bundle attribute updates to options in the bundle. If this setting is not selected, the bundle attributes will initially be cascaded to options in the bundle, but subsequent changes in the bundle attribute will not be cascaded to option attributes.
Check Many Options	Select this to enable mass selection of options on the Configuration and Service Bundle page. This setting allows the Sales rep to select, clear the selection, and update options, and attributes before processing pricing and expression, and invoking rules.
Cleanup Invalid Rule Prompt	Select this setting to enable removal of line items prompted by currently invalid rules.
Compute Totals In Separate Step	Select this setting to enable CPQ to perform the totaling in a separate step or combine it with the base pricing step. This setting ensures that the totaling, which is dependent on the number of lines, should be done as a separate remoting call. This setting helps to avoid the CPU time limit issue to a large extent.
Constraint Rule Execution Mode	Enter the mode to perform the constraint rule execution on the client-side or the server-side. By default, no value is specified and the system assumes the value as *Server*. Valid values are: • *Client* • *Server* This custom setting supports inclusion, exclusion, and validation rules on the client-side as well as the server-side. ⓘ In a Service CPQ flow, always execute constraint rules in Client mode.

Setting	Description
CR Maintenance Governor Limits Threshold	Enter the percentage of governor limit to be used as a threshold for constraint rule maintenance job. When the maintenance job fulfils the threshold, CPQ divides the maintenance job in multiple batches and then continue to excecute the job. The value must be in range of 10 to 100. By default, the threshold will be 100% of the governor limit.
CSS Override	Enter the name of the static resource to override CSS in the catalog, attributes, options, and cart pages. For more information, see Configuring CSS Override. To define CSS for a specific flow, define the field in the dataset of that flow and leave the field blank in **System Properties**. If you define the **CSS Override** in both the datasets, only the value in **System Properties** is considered.
Currency Field Precision	Enter a number to specify decimal places for currency precision. The default currency precision is 2 decimal places. Currency Field Precision only applies to **Net Price**, irrespective of the currency. You can create a formula field to define the precision of the values of the amount fields based on different currencies.
Custom Asset Action Label Name	Enter a label name for a custom asset action.
Custom Asset Action Page	Enter the Visualforce name for the custom asset action page.
Custom Deal Guidance Page	Enter the name of the custom Deal Guidance page.
Custom Option Attribute Page	Enter the name of the custom Option Attribute page.
Custom Pre-Pricing Fields	Enter the list of custom fields from the line item object that determine the pre-pricing indicator.

Setting	Description
Custom Pricing fields	Enter the custom field API name from the line item object displayed on the cart page. To add more than one field name, separate the names by a new line or a comma. When you makes any changes to these custom fields, the system is set to reprice before clicking **Review and Finalize**.
Default Asset Pricing Indicator?	Select this setting to make asset pricing default for Asset Related Line Items.
Default Catalog Page	Enter the Visualforce page name for the default catalog page. Valid values are: • SelectConfigProducts • SelectConfigProductsFilterView
Defer Constraint on Option Selection	Select this setting to process the constraint rules on option selection on the Configuration Page. If the selected the constraint rules deferred until the Sales Rep clicks **Go to Pricing** or any other action button available on the Configuration page.
Defer Pricing	Select this setting to price the products before displaying the cart or after displaying the cart. The default is true (defer until cart is displayed). The Sales rep can use the **Update Price** button on the Configuration page to calculate pricing after updating the fields.
Defer Validation Check in Bundles	Select this setting to defer the validation check in the bundle configuration step. This is setting is selected by default.
Direct Configure Asset Actions	Select this setting to open the options page for the first bundle product that is configurable. The rest of the bundle products are added to the cart and displayed under the Selected Products section on the right-hand side panel. If the bundle product is set as *Must Configure*, an exclamation icon is displayed for incomplete configuration.

Setting	Description
Disable Charge Type Totaling	Select this setting to disable the totaling of charge types.
Disable Constraint Rules	Select this setting to disable constraint rules. This may be used to optimize performance when constraints are not used in the organization. The default value is false (enabled).
Disable Existing Asset Pricing	Select this setting to disable the existing asset pricing. If disabled, prices of existing assets are not rolled up.
Disable Pricing Query Optimization	Select this setting to disable pricing query optimization for the cart. This setting allows the system to load a large number of managed fields from the line item into the cache. By default, fewer fields are retrieved for line items in the cart to optimize the heap. This indicator is used to disable the optimization so that it supports the existing customization.
Disable SOSL Text Search	Select this setting to disable the SOSL catalog search. When disabled, CPQ uses SOQL catalog search.
Display Cart Actions As Dropdown	Select this setting to display the line item action buttons as a drop-down menu on the cart page.
Enable Adjustment Buckets	Select this setting to enable adjustments applied in the Mass Adjustments dialog box for a line item to be grouped in buckets, to be used as targets for applying further adjustments. This setting works only if you have set a finite value in the **Max Adjustment Lines** custom setting.
Enable Adjustment Spread	Select this setting to enable the adjustment on a bundle to spread to its options.
Enable Aggregate Pricing	Select this setting to enable aggregate pricing. This may be used to optimize performance when aggregate price rules are not used in the organization. The default value is false (disabled).

Setting	Description
Enable Auto Reprice	Select this setting to automatically reprice the cart when a price component is modified, without applying any pricing rules. This setting provides client-side computation facility. • On Cart Grid with custom setting **Enable Auto Reprice** set to ON, when you change any Quantity on cart line item and click **Reprice**, CPQ updates the pricing fields such as Extended Price, Net Price for the specific row. • Calculates totals (by charges and frequency) and grand totals • Calculates subtotals for category As a sales rep you can see estimated pricing without any delay using this setting. Estimated pricing means when quantity is changed and adjustment or discounts are provided, pricing calculation is done at the client-side.
Enable Auto Sequencing For Options	Select this setting to enable auto sequencing for options. If you do not select this option, the Option Item Sequence in the shopping cart and the Line Itemsisbased on the user selection sequence rather than the Bundle Options sequence.
Enable Base Price Adjustment	Select this setting to enable unit level price adjustment for the Base Price of the product in the cart.
Enable Base Price Rounding	Select this setting to enable CPQ to round list price and base price fields in the line item using the currency field precision setting.
Enable CartLocking for Concurrent Access	Select this setting to enable your cart for concurrent access to other sales representatives. For more information, see Enabling Cart Locking for Concurrent Access.
Enable Contextual Totals	Select this setting to display subtotals and totals section at the bottom of the cart page for Cart Grid UI. By default, this setting is not selected.

Setting	Description
Enable Custom Rounding	Select this setting to enable rounding of selling term and price in each price calculation step. • Rounding of Selling Term: This setting considers the precision of selling term defined in the **Term Field Precision** setting and rounds the selling term consistently both on the UI and in backend calculation. • Rounding of Pricing: This setting rounds the price in each price calculation step considering the precision defined for each currency, both on the UI and in backend calculations. The amount in price calculation steps (for list price, base price, adjusted price, and net price only after each step such as price matrix, price ruleset, line level manual adjustment, group adjustment) is calculated considering the precision of the currency. ⓘ Price rounding at backend occurs only for the base price, adjusted price, and net price when this setting is enabled.
Enable Defaulting For Products	Select this setting to enable defaulting for products. This may be used to optimize performance when product default rules are not used in the organization. The default value is false (disabled).
Enable Default Quantity For ABO Item	Select this setting to enable the default quantity rules for ABO items. When you enable this setting, CPQ supports only attribute-based default quantity during ABO operations.
Enable External Pricing	Select this setting to enable external pricing. The default value is false (disabled).
Enable Fast Cloning	Select this setting to enable fast cloning.

Setting	Description
Enable Field Expressions	Select this setting to enable users to use Expression Builder for attributes, option groups, and options. This setting is enabled by default.
Enable File	Select this setting to enable use of Files (Salesforce). This setting also enables preview on the Doc Gen page. After you select this setting, it is mandatory to run Migration, which moves the documents in Notes & Attachments from CPQ to Salesforce Files.
Enable Keyed Matrix Pricing	Select this if you have option products with a large number of dimensions of type discrete. This also helps in reducing SOQL queries.
Enable Location	Select this setting to enable users to select a location for a cart. The line items will have the current cart location assigned to them.
Enable Matrix Pricing For Options	Select this setting to enable matrix pricing for options. This may be used to optimize performance when options do not use price matrices in the organization. The default value is false (disabled).
Enable Notification Feed	Select this setting to enable push notifications. The notifications are enabled for the following features: • Quote Collaboration notifications. The notification pop-up appears when any collaboration request is submitted, completed, accepted, and merged in the parent cart. • To enable push notifications for async CPQ operations. For more information, see Configuring Progress Tracker for Async Operation.
Enable Optional Items	Select this setting to enable optional items in the cart.
Enable Paginated Grid	Select this setting to enable pagination for cart grid.

Setting	Description
Enable Price Matrix Audit Trail	Select this setting to enable CPQ to create audit trail entries when adjustments are applied on the Line Item through Price Matrices. The audit trail entries are created on the Adjustment Line Item object. ⓘ This feature does not work if you use adjustment buckets.
Enable Price Rule Audit Trail	Select this setting to enable CPQ to create audit trail entries when promotions are applied on the Line Item through Price Rules. The audit trails entries are created on the Adjustment Line Item object.
Enable Total Level Incentive	Select this setting to enable incentives at the total level.
Enhanced Search URL	Select this setting to enhance search URL endpoint.
Exclude Optional Products	Select this to exclude optional products when CPQ evaluated the bundle's structural requirements. When you enable this setting, CPQ excludes the optional products from the option group Min/Max criteria.
Expand Bundles in Cart	Select this setting to expand all attributes and options of a bundle product in the cart. ⓘ Pagination feature is not supported along with Expand Bundles in Cart.
FavoriteFilters	Enter the API names of fields of type picklist and multi-select picklist from favorite configuration object. These fields define the filters that can be used to categorize favorite configuration on catalog page. If you define default values for the picklists, they are displayed on the catalog page as well. Only the first five fields are displayed on the catalog page. Enter the API Names of the fields separated by a comma or by a new line.

Setting	Description
Field Expression Execution Mode	Indicates where the numeric expressions are evaluated. Following are the values: • Server - (Default) Enables the server-side execution of the numeric expressions. • Client - Disables the server-side execution of the numeric expressions.
Fixed Button Bar	Select this setting to display a fixed bar for actions button on the catalog, attributes, options, cart, and installed products page.
Generate Relative Url For Sites	Select this setting to enable CPQ to generate relative URL for links for a Communities user. This custom setting is deprecated.
Grid Render Threshold	This setting helps to improve the grid scroll performance, with a default value of 20. You must adjust the value based on the number of lines displayed on the Cart. ⓘ Earlier the scroll performance was managed using **APTS_VirtualizationThreshold**, which is now deprecated. You must use the Grid Render Threshold instead of **APTS_VirtualizationThreshold**.
GroupBy Fields	Specify the API names of those line item fields that are of the type: Lookup, Formula Fields, and Text. These line item fields are shown under the Group By drop down on the cart page. For example, for Group By PriceList, specify the API name of the field that is a lookup to the pricelist object. For more information, see Viewing the Cart in Grid View. ⓘ API names with relationship fields are not supported. For example, *Apttus_Config2_ProductIdr.name* is not supported. However, the regular Line Item Field API Names like *Apttus_Config2ProductId_c* are supported
Guide Page Default	Enter the Visualforce page name for the default guide page for product selection.

Setting	Description
Hide Asset Actions	Enter the actions to hide on the Installed Products page. Use a comma to separate values. For example, Increment, Amend, Renew, Cancel. These actions are tied with your Flow settings. For more information, see Defining Allowed Actions for Assets.
Hide Cart Views	Select this setting to hide Cart Views on the cart.
Hide Cart Views Creation	Select this setting to hide the **Create New View** menu option in the Cart View.
Hide Copy Action	Select this setting to hide the copy action in the cart. The default value is false (copy icon is displayed).
Hide Dates For One Time Charge	Select this setting to hide the dates for price list items of type One-Time Charge.
Hide Evergreen End Date	Select this setting to hide the end date for Evergreen Proposals.
Hide Grand Total	Select this setting to hide the grand total line on the cart page.
Hide Resolve Config Link	Select this setting to hide the link to the resolve configuration error page and instead display the error along with the constraint rule message.
Hide Subtotals In Cart	Select this setting to hide subtotals in the cart view. The default value is false (show subtotals in the cart). When selected, the system hides subtotal lines in the Summary section and still shows Category Total, One Time Total, and Grand Total.
Instance Url	Enter the Salesforce instance URL (for example, https://na7.salesforce.com). The instance URL is required to navigate to custom pages in the managed package.

Setting	Description
Keep Abandoned Carts	Select this setting to keep abandoned carts. The default value is false (abandoned carts are moved to the recycle bin).
Large Cart Spilt Criteria Fields	This setting is not functional. You must use the Split Cart Criteria Fields setting to define criteria for the Large Cart flow.
Large Cart Threshold	This setting is not functional. You must use the Split Cart Threshold setting to define criteria for the Large Cart flow.
Max Adjustment Lines	Enter the maximum number of discount lines allowed for a line item.
Max Allowed Lines For Mass Actions	Enter the maximum number of line items to be selected for mass actions on the cart. The default value for selecting the line items for mass action is 10. **Max Allowed Lines For Mass Actions** custom setting is overridden when **Perform Mass Actions in Parallel** custom setting is enabled by the user for selecting a large number of line items on the cart for mass action in one go.
Max Constraint Rules Round Trip	Enter the maximum number of round trips after which the constraint rule processing should stop. Round trip happens only when auto included products trigger more rules. The default value is 3.
Misc Charge Types	Enter the list of miscellaneous charge types. Each charge type should be separated by a new line. The charge types are displayed as picklist values when adding miscellaneous items to the cart. The default values are *Sales Tax* and *Shipping & Handling*.

Setting	Description
Multi Currency Management	Enter a value to define the currency precision in a multi-currency org. Valid values are *Platform* and *None*. • *Apttus:* This is the default value. If you choose this value, the system considers the currency precision settings that you have defined using Currency Field Precision. • *Platform:* If you choose this value, the system considers the precision settings for the currency defined at **Setup > Administer > Company Profile > Manage Currencies**. When you click **Manage Currencies**, the decimal places defined for your **Corporate** currency is considered. If you do not specify any value in the **Decimal Places**, the system considers the default value as 2.
Option Line Item Columns	Enter the field name from the line item object displayed on the Option Hierarchy page. To add more than one field name separate the names by a new line or a comma.
Option Pricing Chunk Size	Enter a number for the option pricing chunk size for bundles with a large number of options. The default value is 100. The SFDC governor limit affects the chunk size.
Option Product Columns	Enter the field name from the product object displayed on the Option Hierarchy page. To add more than one field name separate the names by a new line or a comma.
Options Page	Enter the Visualforce page name for the default options page. If no value is specified, the options page defaults to the icon view. Valid values are: • SelectConfigOptions • SelectConfigOptionsTabView • SelectConfigOptionsListView
Percentage Field Precision	Enter a number to specify decimal places for percentage precision. The default percentage precision is 2 decimal places.

Setting	Description
Perform Mass Actions in Parallel	Select this setting to perform mass actions on the cart. You can copy and remove multiple line items from the cart by mass action in one go.
Populate Attribute Extensions	Select this setting to indicate that the attribute values are populated in the attribute extension objects.
Pricing Batch Size	Enter a number to define the number of line items that can be processed in a single pricing call. Setting the Pricing Batch Size, the system runs pricing with the specified number of products as a batch, thus increasing performance. These batch calls to the database are governed by the Salesforce CPU time limit and hence the number assigned for Pricing Batch Size must be carefully evaluated. ⓘ You can configure this setting based on a flow. In such a case, the flow-based setting overrides the generic setting.
Pricing Profile	Enter *Basic*, *Advanced*, or *External*. A pricing profile is *Basic*, when there are none of the following used: • Pricing Rules • Price Matrices • Related Pricing Setup • Bundles If the Pricing Profile field is left blank, the default value is *Advanced*. ⓘ You can configure this setting based on a flow. In such a case, the flow-based setting overrides the generic setting.
Product Attribute Detail Page	Type the Visualforce page name for default attributes page. This is the custom page to capture product attribute details.

Setting	Description
Product Attribute Extension Tables	Enter comma-separated API names of the lookup relationships, created with the **Product Attribute Value** object. For example, if you have 4 extension objects to the **Product Attribute Value** object and their corresponding 4 lookup relationships, you must specify 4 API names in this field.
Product Display Max Length	Enter the maximum number of characters to be displayed for the product name on the cart page. If the product name exceeds the maximum length, the ellipsis is shown. The default value is 21.
Product Option Price Order	Enter the order of execution of option pricing (with adjustments) in conjunction with the price rule set and matrices. The valid values are: *First* and *Last*. By default, the value is *Last.* • *First* - The system applies the adjustments before the price rulesets and after price matrices. • *Last* (Default) - The system applies the adjustments after the price rulesets.
Product Sorting Fields	Enter the list of API names of fields from the Product object to be displayed in the sorting drop-down on the Catalog page. The Sales rep can use the **Sort** drop-down to select a field from the list to sort the products on the Catalog page.
Quantity Field Precision	Enter a number to specify decimal places for quantity precision. The default quantity precision is 2 decimal places.
Refine Search Checkbox Display Limit	Enter a number to specify the limit after which the Refine Your Search options are displayed as a picklist. By default, the options are displayed as checkboxes. The minimum and the maximum values are 0 and 20 respectively. Any number less than 0 is considered 0 and any number greater than 20 is considered 20.

Setting	Description
Related Price Scope	Enter a value to perform the related pricing calculations over the entire cart or confine the calculations to a Bundle product only. For example, if the price of an Option product is a percentage of another Option product in the same bundle, consider the following scenarios with **Related Price Scope** = *Cart* and *Bundle*, • If value = *Cart*, the Option price is based on the price of all the instances of the related product in the cart, whether it is in the same Bundle or a Standalone, or in another Bundle. • If value = *Bundle*, the Option price is based on the related product instances within the same Bundle.
Remove Invalid Rule Products	Select this setting to automatically remove line items auto-included by invalid rules on cart launch. If the setting is disabled, the user is prompted to take appropriate action.
Resolve Configuration Page	Enter the default Resolve Configure page name. On Resolve Configuration Errors/Warnings, this page is used. • *ResolveConfig* • *ResolveConfigProducts*
Revalidation Product Columns (D)	Type the API names of product object fields that you want to display on the revalidation pop-up on the cart page. You can separate each field API name either by a comma or by a new line. ⓘ This setting is deprecated.

Setting	Description
Rounding Mode	Enables currency rounding in pricing calculations. This setting rounds adjustments before calculating the base price. Enter one of the following values: • UP: Rounds the currency to the next number. For example, 21.2 is rounded to 22. • DOWN: Rounds the currency to the previous number. For example, 21.8 is rounded to 21. • HALF_UP: Rounds the currency to the next number if the decimal is equal to or greater than 5. For example, 21.5 to 21.9 is rounded to 22. • HALF_DOWN: Rounds the currency to the previous number if the decimal is equal to or smaller than 5. For example, 21.1 to 21.5 is rounded to 21. • HALF_EVEN: Rounds the currency to the nearest even number. For example, 23.5 is rounded to 24 and 22.5 is rounded to 22.
Run Misc Finalization Task in Async Mode	Indicator to check whether proposal line product attribute and usage tier records to be created with some delay after you **Finalize** a cart. In a scenario when the user creates Asset or generates a document immediately after finalizing the cart, there are chances that the record does not have attribute value or usage tiers. If this checkbox is selected, the usage tiers and attribute record for proposal line items are created in Async mode when you finalize a cart. Otherwise, the usage tiers and attribute record for proposal line items are created as soon as you finalize a cart.
Run Post Finalize Trigger In Async Mode	Indicates whether the finalization task should be run asynchronously.
Run Validation Callback On Add	Select this to invoke the Validation Callback upon upon clicking **View Cart** on the Mini Cart.

Setting	Description
Same Day Cancellation	Select this option if you want the Asset Cancellation applicable on the same day. For example, while performing Termination on an Asset you enter the Termination Date as 5/11/2016. If you select Same Day Cancellation, the cancellation will be effective from 5/11/2016. Otherwise, the cancellation will be in effect a day later which is 5/12/2016. Same Day Cancellation is applicable by default. Clear the checkbox to apply cancellations after a day.
Save On All Actions	Saves the configuration when you click any action on the cart page.
Search Category Default	The default search category. Indicates the category to default to when searching for products using the search text box in the product selection page. The valid values are the following: *All Categories All Products* The default value is *All Categories*. Choose *All Products* to search all products regardless of categories.

Setting	Description
Selling Term Calculation Method	Indicates the method to calculate the selling term for line items on the Cart page. The valid values are: • Calendar: The selling term is calculated based on the calendar. This is the default method. • Billing Preference: The selling term is calculated based on the billing preferences. You can price transactions based on the term calculated according to the user's preferred start day instead of it always being the first of the month. When the Conga Billing package is installed, you must set the **Selling Term Calculation Method** setting to Billing Preference. CPQ then calculates the selling term based on the billing preferences. For more information, see Billing Preferences. The old values supported for this field were *Normal* or *Precise*. Whenever you upgrade CPQ packages, the last value of *Calendar* or *Billing Preference* (added before upgrade) must be retained. If the value of this field changes to *Normal* or *Precise* (old values) after an upgrade, manually change it to *Calendar* or *Billing Preference*. The old values of this field (Normal and Precise) are not automatically removed from existing installations.
Service Line Split Criteria	Defines the criteria to clone the service line.

Setting	Description
Service Price Distribution Method	Enter a value to define the mode used for service pricing. The valid values are: • **Rollup:** Considers all entries of asset components and service line items and the rolls up the final value. Enter this value when the customer has a need to pool and price related equipment with different service prices. • **Allocate:** If values for different services are different and user clicks **Relate** and adds another option, CPQ takes the price of the matrix entry and then distributes to the entire service price. Enter this value when the customer has a need to create a service line only for similarly priced related equipment line items. In this case, the dimensions from the first set of the related line items are used to price the service.

Setting	Description
Show Admin Sidebar	Hides the sidebar for the following Admin pages: • CPQConsole.page • CategoryManager.page • ClassificationHierarchy.page • ConstraintRuleActionCriteriaEdit.page • ConstraintRuleConditionCriteriaEdit.page • CriteriaUpdate.page • DisplayActionSettings.page • DisplayColumnSettings.page • FeatureSet.page • HierarchyViewUpdate2.page • IconUploader.page • ManageGuidedSearchRule.page • MultipleConstraintRulesAdmin.page • PriceListItem.page • PriceRule.page • ProductConsole.page • ProductConstraintView.page • ProductDefaultRuleCriteriaEdit.page • ProductFilterMaintenance.page • RelateAttributeToProduct.page • RelateFeatureToProduct.page • RelateProductToBundle.page • RelateProductToCategory.page • RelateProductToFootnote.page • SystemProperties.page
Show Attributes in Cart	⚠ This setting is deprecated.
Show Header	Displays the header tabs and the sidebar on the custom pages.
Show Info In Header	Indicates whether to show information message from auto include or auto-exclude in the header. The default value is false (Hide).
Show Radio Buttons for Asset	Enable to display radio buttons for asset actions Renew, Increment, Amend, and Cancel.

Setting	Description
Show Tab View	Displays option groups as tabs instead of sections in the options page. You can enable the Show Tab View setting at a product level also. Select the checkbox on the product details page.
Skip Constraint Decision Field	Indicates that fields on the line item skips the Constraint Rule check. Enables bypassing constraint and finalize quote even if there are constraint error.
Skip Review	Skips the review step for items in the cart.
Split Cart Criteria Fields	Defines the criteria to break the summarized view of items added during a large cart process. List the API names of the line item fields separated by a comma or new line. Enter the field names whose values do not change after the records are created for a cart.
Split Cart Threshold	Indicates the number of line items, when exceeded a cart is created. When you define the threshold as 20 and the cart has 30 line items, a cart is created for the remaining 10 line items. This setting is applicable for *Split* quotes. The default batch size for order line item creation and order activation is 100 unique bundle lines. This means 100 unique products on the cart (each product can have multiple line items).
Static Criteria Fields	Defines the criteria to break the summarized view of items added during a large cart process. List the API names of the line item fields separated by a comma or a new line. Type the field names whose value does not change after the records are created for a cart.
Term Field Precision	Type the precision value that you want to display for the **Selling Term** column on the shopping cart. For example, if your term calculation results to 4.553412 and you have set **Term Field Precision** to *3*, the **Selling Term** column displays 4.553. The default term precision is 5 decimals. This setting is applicable only for the New UI.

Setting	Description
Totaling Group Type	The totaling group preference. The valid values are the following: • *Category*: When you add a product to the cart from the lowest leaf category where it is associated, the Category Hierarchy column in the **Line Items** related list is populated with its bread-crumb trail. This is useful to identify the exact category from where the product was added, in case if the product is associated with multiple categories • *Product* The default value is *Category*. For Product-based totaling, the **Product Totaling Hierarchy** setting is required.
Update View Category Batch Size	Defines the number of categories to update views in a single transaction. Enter a number to define the batch size.
Update View Product Batch Size	Defines the size of the batch to update views of products in a single transaction. Enter a number to define the batch size.
Update View Use Dml Limit	Indicates whether the DML limit is used to determine the workload of a batch to update views of products.
Use Button to Save Selection	Indicates whether to wait for button click before processing option selection and validation.
Use Enhanced CSS	Enables you to include the enhanced CSS file on CPQ pages.
View Cart Custom Fields	Enter the list of custom fields from the line item object displayed in the custom view cart page. Each field API name should be separated by a new line or a comma.
View Cart Custom Fields 2	Enter the list of additional custom fields from the line item object displayed in the custom view cart page. Each field API name should be separated by a new line or a comma.
View Cart Page	Type the Visualforce page name for default cart page.

Setting	Description
View Cart Total Custom Fields	Enter the list of custom fields from the summary group object displayed in the custom view cart page. Each field API name should be separated by a new line or a comma.

When a line item of **Charge Type** = *One Time* or a proposal that is *Evergreen* is created, you can choose to hide the dates of the line item or in the proposal. Similarly, if the Line Item has **Auto Renewal** marked as *true*, you can choose to hide the dates on those line items.

The behavior listed below is true when you split the bundle ramps.

Scenario	Flags selected	Result
If **Price Type** is *One Time* on Line Item	Hide Dates For One Time (If selected)	Hide **Start Date** and **End Date** on Line Item.
	Hide Dates For One Time (If not selected)	**Start Date** and **End Date** are visible.
If **Price Type** is *Recurring* or *Usage*	Hide Evergreen End Date (If selected) and AutoRenewalType = Evergreen on Line Item	**End Date** is hidden.
	Hide Evergreen End Date (If not selected) and AutoRenewalType = Evergreen on Line Item	**End Date** should behave the same way it behaves for recurring.

Config User Preferences

This custom setting holds Configuration and Pricing user preferences.

Setting	Description
Catalog Products Per Page	Enter the number of catalog products that you want to display on the Catalog page. Valid values are: 10, 20, 50, and 100.
Category Preference	Enter the Category preferences for the user or profile. Enter the list of category names separated by comma.
Collapse Error Message	Select this setting to collapse error messages.

Setting	Description
Collapse Info Message	Select this setting to collapse info messages.
Collapse Quick Add Filter	Select this setting to collapse the quick add filter by default.
Collapse Warning Message	Select this setting to collapse warning messages.
Flow	Enter the name of the default flow associated with the user.
Groups Per Page	Number -
Items Per Page	Enter the number of browsed or searched products to be displayed on the following pages: • On the product selection page • On the cart grid for showing line items (if pagination is enabled) • On the promotions pop-up • On the Mini-Cart • On the assets grid
Logging Level	Enter the logging level for JavaScript code. Valid values are: Debug, Info, and Error.
Option Items Per Page	Enter the number of displayed option products to show in the cart page. Valid values are 5, 10, 15, 20, and 25.
Selected Comparison Products	Enter values to maintain selection of compared products, between the Catalog and the Compare Features page.
Selected Products Per Page	Enter the number of selected products that you want to display on the configuration pages. Valid values are 5, 10, 15, 20, and 25.

Installed Products Settings

Custom Settings for the Installed Products Page in CPQ.

Setting	Description
Account Hierarchy Batch Size	Enter the number of Account Hierarchy records processed for each batch execution. Recommended value is 500. Valid values range from 1 to 2000.
Alert Asset Related To Renewal Cart	Select this setting to display a warning on the Installed Products page when the user tries to manually renew an asset associated with a system-generated renewal quote or renewal agreement. The setting is not selected by default.
Allow Backdated Termination	This setting is disabled by default, that means, user cannot terminate an asset by specifying an end date to any date prior to the current date. End date should be greater than the start date. Select this setting to enable user to specify a termination date earlier than today. This may have additional impact on billing because an invoice for the period may have already been generated.
Allow Mass Change	Select this setting to enable user to perform mass changes for *must configure* assets.
Amend Change Fields	Enter comma separated line item field names whose values can turn an existing asset line into an amended asset line. This setting is no longer used. All the editable fields honor the flow settings configured for the Cart.
Apply Adj To Current Contract Term	Select this setting to keep the discounting limited to the current contract term. When you enable this setting, CPQ applies adjustments and calculates the net price and net unit price of the assets based on the current contract term only. When you disable this setting, CPQ applies adjustments and calculates the net price and net unit price of the assets based on the original start date of the asset.
Asset Currency Field Precision	Enter the number of decimal places for the asset currency fields. (used by the formatfields function)

Setting	Description
Asset Source	Enter the account source which the asset line items will be retrieved from. This setting allows you to filter the assets to be displayed on the Installed Products page based on the **Ship To** or **Sold To** or **Bill To** fields of an account. By default, all the assets filtered with the Sold To field are displayed. Specify the API name of the Account field that you want to use as a source for displaying assets. For example, to filter the assets based on **Ship To** field, enter Apttus_Config2__ShipToAccountId__c. You can specify all the three account fields and the system considers an OR condition between these fields while filtering. This is useful when an account has different locations for shipping and billing and you want to display only those assets that have been shipped to a certain location.
Asset Termination Fields	Do not use this setting. Instead use the *Asset Termination* option from the **Display Type** picklist. For more information, see Configuring Display Columns Settings.

Setting	Description
Base Price Defaulting Method For Renewal	Enter a value to define how CPQ must default the base price when an asset is renewed. Valid Values are: • Net Unit Price (default if not specified): CPQ renews the asset line with the net unit price. If no value is specified for this setting, this is the default behavior of CPQ during renewal. • Higher Of Contract Unit Price / Net Unit Price: CPQ renews the asset with the higher value of contract unit price of net unit price of the asset. • Lower Of Contract Unit Price / Net Unit Price: CPQ renews the asset with the lower value of contract unit price of net unit price of the asset. ⓘ Other asset operations still use the net unit price for loading the asset line items. CPQ applies this setting to bundle and its options similarly (that means, a bundle cannot have a higher price while its options have a lower price). You can define this setting at a flow level; however, the flow-level setting overrides the setting defined at **Primary Settings**.

Setting	Description
Cotermination Preferences During Renewal	Enter one or more of the following values (comma separated values, without space) to allow the user to determine renewal end dates. • UseProposalOrAgreementEndDate: This option enables users to renew an asset using the proposal or agreement end date. CPQ displays the **Use Proposal End Date** option if the user renews assets using the quoting flow and the **Renewal Business Object Type** is Proposal. CPQ displays the **Use Agreement End Date** option if the user renews assets using the agreement flow and the **Renewal Business Object Type** is Agreement. ⓘ If the user performs asset renewal through the Asset Manager flow or CSR flow, the **Use the Proposal End Date** or **Use Agreement End Date** option is not displayed on the Confirm Renewal intermediate page. If the user performs asset increment through the Asset Manager flow, the **Use Agreement End Date** option is not displayed on the Change Quantity intermediate page. • RetainCurrentAssetEndDate: This option enables users to retain the current asset end date while renewing an asset. • UseFarthestAssetEndDate: This option enables users to use the farthest asset end date while renewing assets.

Setting	Description
	• UseRenewalDate: This option enables users to renew assets with a custom renewal date. You must enter at least one value in this setting. If you have more than one value, you must also select a default value in the **Default Renewal Cotermination Option** setting. If you do not select any default value, the first of the selected values is considered as the default value. Depending on the selected setting, users can now bypass the Confirm Renewal intermediate page altogether. • If there is only selection for co-termination (apart from *UseRenewalDate*), the Confirm Renewal intermediate page is completely bypassed. On clicking **Renew**, CPQ creates the renewal line and refreshes the Installed Products page to show the detail in the Mini Cart. • If there is more than one selection for co-termination, CPQ displays the Confirm Renewal intermediate page to allow the user to make a selection. • If the selected option is *UseRenewalDate*, CPQ displays the Confirm Renewal pop-up to collect the user input.
Create Renewal Opportunity	Select this setting to enable CPQ to create a renewal opportunity for system generated asset renewal quotes. This setting indicates if you want to create renewal opportunities along with renewal quotes. By default, this check box is not selected. This is to prevent the system from running into locking issues when an account has a large number of assets and the batch process creates the renewal opportunities for all these assets. When this setting is true, the Create Renewal Opportunity scheduled job runs and creates the renewal opportunities along with renewal quotes. When this setting is false, the Create Renewal Opportunity scheduled job runs and does not create the renewal opportunities and creates only the renewal quote.
Default Cotermination Option	Enter the default end date preference for cotermination.

Setting	Description
Default Renewal Cotermination Option	Enter the default end date preference for renewal cotermination. If there is only one value in the **Cotermination Preferences During Renewal** setting, that option is selected by default. If there are more than one options, you can select a value as a default renewal cotermination option. Based on the default option, CPQ loads the selected renewal transactions on the Installed Products page, Confirm Renewal popup, or Confirm Renewal intermediate page.
Default Renewal Price List	Enter the name of the Price List which is a mandatory field for Quote creation.
Default Renewal Term	Enter a value for a renewal term. When the value is provided use this value as the renewal term for Renewing products. The value here denotes months to renew as the end date. Ideally, your *Selling Term* is equal to your *Default Renewal Term*.
Editable Fields for Cancelled Lines	Enter comma-separated API names of the fields that you want to make editable for cancelled lines on the cart (during asset termination). You can enter a custom currency field that is not based on a formula. You cannot enter out-of-the-box currency fields such as base price and base extended price as editable fields.

Setting	Description
Enable One Time Change	Select this setting to enable modifications or amendment to one-time assets. This is a global setting and cannot be restricted to apply for specific products. You can perform only amendments on one-time assets and not renewals. If you deselect this and modify the asset, the renewal quote is updated for the modified asset. If you set it to false, Renewal Opportunity and Renewal Quote are not created for the one-time only products, but if you modify the asset, the change is reflected in the Renewal Quote. If you set it to true, Renewal Opportunity and Renewal Quote is not created for one time. It is recommended that you treat one-time assets purely as one-time sales. Any modifications or renewals should not be encouraged for one-time assets.
End Date preferences for Cotermination	Enter comma separated values (without space) to enable the user to select one of the following end dates while coterminating incremental licenses. • RetainCurrentAssetEndDate • UseCustomDate • UseProposalEndDate ⓘ If the user performs asset increment through the Asset Manager flow, the **Use the Proposal End Date** option is not displayed on the Change Quantity intermediate page.
Filter Fields	Enter the API name of Asset Line Fields to be displayed on the **Filter By** panel. Works with text fields, Picklist, Multi-picklist, Date, DateTime and Boolean. Commonly used fields are **Start Date**, **End Date**, **Lead Time Expiration**, and **Asset Status**. Usually, all fields under Asset Line Item object are supported but you must enter *ExpirationLeadTimeInDays* value in order to display Lead Time Expiration field in the Search pane.

Setting	Description
Hide Co-Term	Select this setting to hide the co-term end date on renew. This setting hides the Coterminate with Increment panel on the Change Quantity (Increment Asset) page. The Coterminate with Increment panel is displayed by default.
Max Renewal Group Per Job	Enter the number of Max Renewal Groups processed by one batch job. This setting is used to limit the length of batch job to avoid too many running batch jobs.
Max Renewal Line Item Per Cart	Enter the number of Max Renewal Line Items to be processed in one cart. This setting is used to limit the size of product configuration to avoid exceeding CPU and SOQL limits.
Max Renews Per Trip	Enter the maximum number of renewals per trip. To prevent CPU timeout use a smaller number. The default is 20.
Merge Action Criteria Fields	Enter comma-separated asset line item fields that must be common among the asset line items to be merged. The default fields are product, selling term, charge type, and end date.
Purchase Identification Criteria	Enter comma separated asset line item fields that determine the eligibility for cotermination and quantity cumulation. This setting allows user to define a criteria to get coterminate related asset lines in the Coternimate with Increment section on the Change Quantity page during asset increment. The default purchase identification criteria are: Product, Charge Type, Option, and Account. CPQ currently supports only Purchase Identifier and alphanumeric custom fields. For example, if you select Purchase Identifier, during increment of a selected asset, all other assets with the same Purchase Identifier are displayed on the Coternimate with Increment section. Based on the selected purchase identification criteria, the total quantity of an asset spans multiple streams during the increment scenarios, on the Installed Products page (of a renewal quote).

Setting	Description
Relate Action Criteria Fields	Enter comma separated asset line item fields to enable relate asset action. This custom setting shows all fields of Asset Line Item and Price Dimensions defined on asset line item. When the user selects two or more assets and clicks **Relate**, CPQ validates the selected asset line items against the Relate Action Criteria Fields. In case of any discrepancies, CPQ displays a validation error message to the user.
Renewal Business Object Type	Enter the Business object for which this renewal is taking place. Valid values are: *Proposal* and *Agreement*
Renewal Default Price Book	Enter a Renewal Price Book name which you want to associate with renewals.
Renewal Execution Mode	This setting indicates if the Renewal of Asset Line Items must happen automatically or based on your specified conditions. • Enter **Auto** to automatically create the Renewal Opportunity on Order activation. Make sure you check **Auto Renew** on the product **PLI**, from the **Default** tab. • Enter **OnDemand** to create the renewal Opportunity before a certain lead time. You must enter a **Renewal Lead Time** for this mode to work successfully. To create a renewal Opportunity 90 days before the Asset Expiry, set the **Execution Mode** to *OnDemand* and **Renewal Lead Time** to *90*.

Setting	Description
Renewal Group Fields	Enter the API names of the fields you want to use for grouping. This setting indicates how to group the Asset Line Items when a Renewal Opportunity is created. For Execution mode set to **Auto**, the system will group the renewal Opportunity by default by the **Auto Renew** flag - One for the Asset Lines with **Auto Renew** as *True* and the other for Asset Lines with **Auto Renew** as *False*. For Execution mode set to **OnDemand**, the system can group the renewal Opportunity by Account and the Price List. However, if the implementation teams want to group the renewal opportunities by other parameters on the Asset Line Item, they can do so by specifying a comma-separated list of API names of the fields in this section. It is recommended to limit the grouping to a maximum of 4 fields.
Renewal Lead Time	Enter a value when the renewal quote must be created after an order is activated. By default, the value is 0, which indicates that the renewal quote is created immediately after the order is activated, with the same number of Line Items in the order. If you specify 30, the renewal quote is created 30 days before the Asset End Date.
Renew One Ramp	Select this setting to renew only one ramp line item based on the new term in order of renewal term in the asset line item, default renewal term specified in the Installed Product Setting and Selling Term defined in the asset line item. Consider a scenario where multiple ramped assets exist for a quote. In the previous releases, renewing any one of the ramped assets results in the creation of a renewal cart with the same number of ramps as the original proposal. Each ramp line is created from the original deal which implies that the uplift has to be defined from Ramp 1 for accurate prices in the subsequent ramps. Redefining and specifying details for each ramp can be cumbersome and as a user, you would expect that for a three-year ramped deal, the uplift specified on year 3 is applied on renewal to year 4.

Setting	Description
Show Accounts Filter	Select this setting to display accounts filter in the Installed Products page. This is currently not supported.
Show Assets	This setting determines whether current, parent, or child assets of the current asset's account are displayed. Enter comma separated values of Parents and/or Children. If empty, only context account assets are shown. Examples: • parents • parents,children
Show Service Coverage	Select this setting to display service coverage for a primary service.
Split Asset Actions	Enter the actions that can be performed on a split asset. This setting enables the user to perform split asset actions on the Define Split <asset name> page. Valid values are: • Renew: Enables users to perform the Split and Renew asset action. • Swap: Enables users to perform the Split and Swap asset action.

Setting	Description
Submenu Actions	Enter comma-separated submenu actions that you want to display when the user clicks an action button on the Installed Products page. You can configure submenu actions only on the **Change** and **Relate** actions. The supported values are: • **Change:Configuration** displays the Configuration submenu under the Change action. This submenu enables users to configure an asset line item. • **Change:Quantity** displays the Quantity submenu under the Change action. This submenu enables users to initiate an action to add more licenses for an existing asset stream. • **Change:Split** displays the Split submenu under the Change action. This submenu enables users to split an asset into multiple split lines. • **Change:Merge** displays the Merge submenu under the Change action. This submenu enables users to merge multiple asset lines into a single asset. • **Relate:Component** displays the Component submenu as a drop-down under the Relate action. This submenu enables users to relate an asset component with a service.

Lookup Field Settings

Controls lookup field behavior in CPQ. Lookup field settings provide an ability to filter lookup records through filter criteria based on the immediate parent on which the lookup field is defined.

Setting	Description
Enable Quick View	Select this to enable CPQ to pre-populate the values in lookup fields on the Cart page and Mass Update popup. If you disable the setting, the Sales Rep needs to search the values. CPQ performance is optimized if you disable this setting as the lookup values are retrieved on the Cart page or Mass Update popup only when the Sales Rep clicks the search icon. Conga recommends disabling the setting for lookup fields with a large number of lookup records and also only enable the setting for a maximum of 2 lookup fields that are present on the cart.

Setting	Description
Filter Criteria	Enter a formula to filter lookup values. The formula must be a valid expression to further filter down the search results for a lookup. The field size is 256 characters. Formulas are built with left as a lookup or junction and right as the object. Example: Product__c=Apttus_Config2__ProductAttributeValue__c.Product_Id__c This formula fetches only such attributes whose IDs match with the selected products. Here, the ProductId is a custom formula field that fetches the ProductId of the Line Item object.
Filter Criteria 2	Enter a formula to be used as an extension to **Filter Criteria**. Contents of this field are concatenated to the content of **Filter Criteria** without any validation while evaluating. There is no space between the expressions when they are concatenated because the leading and the trailing spaces are trimmed. Avoid splitting the expression where space is required. Split the expression mid-way between a word to avoid trimming the space at the beginning or end of a word. For example, **Filter Criteria** = *AccountId_c = '00137000002ttoF' AN* **Filter Criteria 2** = *D TaxExempt_c = 'Yes'*. After the fields are concatenated the expression becomes *"AccountId_c = '00137000002ttoF' AND TaxExempt_c = 'Yes"* . *Otherwise, when Filter Criteria = AccountId_c = '00137000002ttoF' AND* **Filter Criteria 2** = *TaxExempt_c = 'Yes'*. After the fields are concatenated the expression becomes *"AccountId_c = '00137000002ttoF' ANDTaxExempt_c = 'Yes"* and results in a syntax error. Note that there is no space between *'AND'* and *'TaxExempt_c'*
Junction Default Flag	Enter the name of a field API from the Junction object to determine the default value of a lookup field. This field must be a checkbox.
Junction Field Name	Enter the name of a field on the junction object which you would like to filter the lookup field by.
Junction Object Name	Enter the name of the junction object which you would like to filter the lookup values. This is a custom object that you can define to filter the values which do not exist on the Lookup Object itself.

Setting	Description
Lookup Display Columns	Enter the names of fields to be displayed on the lookup dialog box. Field names must be separated by comma without spaces. The columns of the Lookup popup fields from the Lookup Object will be displayed by populating with the fields APIs from the Lookup Object such as Name__c.
Lookup Field Name	Enter the name for the lookup field. This is the API name of the Lookup field on the Line Item or Product Attribute Value Object defined under the Object Name.
Lookup Record Limit	Enter the maximum number of records to query for the lookup field drop-down. Default value is 200 to a maximum of 1000.
Object Name	Enter the API name of the object where the lookup field exists. Valid values are: • *Apttus_Config2__LineItem__c* • *Apttus_Config2__ProductAttributeValue__c* • *Apttus_Config2__ProductConfiguration__c.* ⓘ You can define lookup fields with filter criteria from Product Configuration object in Lookup Fields Settings. This allows you to use lookup fields with filter criteria in features like Mass Update

Following are the scenarios where lookup field settings are required:

Make your product attribute as hidden

While configuring a product, along with its attributes, the product attribute lookup will now display only relevant attributes for the products which you have just selected on the catalog page.

Perform the following steps to hide the Product ID attribute.

Prerequisite: You must create a custom formula field, named *Product Id* to fetch the Product Id of the Line Item object.

1. Navigate to **Product Attribute Groups** by clicking **All Tabs**.
2. Select the appropriate Product Attribute Groups and click **Edit**.
3. Select the Is **Hidden** checkbox for Product Id and click **Save**.

To customize the lookup fields, create a new record under **Custom Settings** > **Lookup Field Settings** > **Manage** and fill in the requisite details.

Field	Description
Name	Specify the name for the custom setting.
Display Columns	Specify the API names of fields to be displayed on the lookup dialog box, separated by a comma.
Filter Criteria	Enter a valid expression to further filter down the search results for a lookup. The field size is 256 characters. When the expression exceeds the size, use **Filter Criteria 2**. Example: Product__c=Apttus_Config2__ProductAttributeValue__c.Product_Id__c This will fetch only such attributes whose IDs match with the selected products. Here, the ProductId is a custom formula field which fetches the Product Id of the Line Item object.

Field	Description
Filter Criteria 2	This field is used as an extension to Filter Criteria. Contents of this field are concatenated to the content of "Filter Criteria" without any validation while evaluating. ⚠ There is no space between the expressions when they are concatenated because the leading and the trailing spaces are trimmed. Avoid splitting the expression where space is required. Split the expression mid-way between a word to avoid trimming the space at the beginning or end of a word. For example, **Filter Criteria =** *AccountId_c = '00137000002ttoF' AN* **Filter Criteria 2 =***D TaxExempt_c = 'Yes'*. After the fields are concatenated the expression becomes "*AccountId_c = '00137000002ttoF' AND TaxExempt_c = 'Yes'*". Otherwise, when **Filter Criteria =** *AccountId_c = '00137000002ttoF' AND* **Filter Criteria 2 =***TaxExempt_c = 'Yes'*. After the fields are concatenated the expression becomes "*AccountId_c = '00137000002ttoF' ANDTaxExempt_c = 'Yes'*" and results in a syntax error. Note that there is no space between '*AND*' and '*TaxExempt_c*'
Lookup Field Name	Specify the API name for the lookup field. Example: Product_Edition__c This is the lookup field containing values such as *Enterprise*, *Standard*, and *Premium*.
Object Name	Specify the object name from which the lookup relationship is created. Example: Apttus_Config2__ProductAttributeValue__c

Configuring Users and Queues for Quote Collaboration

You need to configure lookup fields for users and queues so that Quote Collaboration pop up shows the relevant users and queues while assigning the collaboration request.

For configuring User, ensure that you set up the lookup field (under **Custom Settings** > **Lookup Field Settings** > **Manage**) with the following values:

Field	Description
Name	CollaborationRequest__c.User.OwnerId
Display Columns	Name
Filter Criteria	UserType = 'Standard' AND IsActive = true
Lookup Field Name	OwnerId.User
Object Name	Apttus_Config2__CollaborationRequest__c

For configuring Queue, ensure that you set up the lookup field with the following values:

Field	Description
Name	CollaborationRequest__c.Group.OwnerId
Display Columns	Name
Filter Criteria	Type = 'Queue'
Lookup Field Name	OwnerId.Group
Object Name	Apttus_Config2__CollaborationRequest__c

Proposal System Properties

Quote/Proposal System properties.

Setting	Description
Admin User	The admin user is the default owner of activities created by a user who is not allowed to be the owner (for example, customer portal user). Enter the admin user name in the following format: Format = first name, last name
Auto Create Cart Version After Finalize	Select this setting to enable CPQ to create a new version of the cart automatically after it is finalized.
Auto Create Order	Select this setting to enable CPQ to create an order and assets automatically when the proposal is accepted.
Auto Select Attachment?	Select this setting to enable CPQ to select the most recent attachment automatically.
Auto-Select multiple recipients?	Select this setting to enable projects to specify a list of recipients that will be defaulted for eSignature. To use this option, projects will add the **DocuSign Default Recipients** related list on proposals and user will select the list for each proposal.
Auto Select Recipient Field Name	Enter the API name of the field which holds the value for the recipient to select automatically.
Auto Sync With Opportunity	Select this setting to enable CPQ to synchronize the proposal automatically with the opportunity when the proposal is accepted. This setting automatically synchronizes the quote line items with the opportunity when the proposal is finalized. This setting appears only if you haveConga Quote Configuration Integration package installed.
Bypass Sharing	Select this setting to enable apex code to bypass record sharing.
Clone With Approval Status	Select this setting to enable CPQ to retain approval status on the cart and line items associated with the cloned quote/proposal.

Setting	Description
Create Agreement With Approval Status	Select this setting to enable CPQ to retain the approval status on the quote. When an agreement is created from the quote, the approval status on the quote is transferred to agreement line items associated with the quote.
CSS Override	Enter the name of the static resource to override CSS on the Proposal document generation page.
Custom Doc Gen URL	Enter the custom URL that is accessed when using the custom button on the DocGen page.
Default Opportunity Quote Owner	Enter the owner name to set the default owner for the quote/proposal created from an opportunity. Valid values are: Opportunity Owner Current User If not set, *Opportunity Owner* becomes the owner of the new quote/proposal.
Default Quote Agreement Owner	Enter the owner name to set the default owner for the agreement created from a quote/proposal. Valid values are Quote Owner Current User If not set, *Quote Owner* becomes the owner of the new agreement.
Default Template Name	Enter the name of the default email template that must be used when sending emails.
Disable Opportunity Products Copy	Select this setting to disable copying of opportunity products to the quote/proposal.

Setting	Description
Document Naming Convention	Enter a value to apply a custom naming convention for all proposal documents at generation. The following attributes are permitted when formulating a document naming convention: • %action% • %templatename% • %user% • %timestamp% • %version% • Proposal attributes such as %:Name%. ⓘ Any variable prefixed by ':' represents a field on the Proposal object. If the property contains a null value (i.e., if left blank), the current default naming convention will be used: "%:Proposal_Name_c%_%templatename%_%timestamp%". Example of a proposal document name using the default naming convention: SOW_Regenerated_SOW ABC_2019-08-07
Email Template For Presenting Proposals	Enter the name of the email template that must be used for presenting proposals.
Enable Document Preview in DocuSign App	Select this checkbox to enable document preview in DocuSign application.
Enable Fast Doc Gen	Select this setting to enable the automatic generation of the proposal document upon the click of the **Send Proposal** button on the Quote Details page.

Setting	Description
Enable File	Select this setting to enable use of Files (Salesforce). This setting also enables preview on the Doc Gen page. After you select this setting, it is mandatory to run Migration, which moves the documents in Notes & Attachments from CPQ to Salesforce Files.
Enable PDF Security	Select this setting to enable users to apply security settings to PDF documents and protect them with a password.
Enable Proposal Email Editing	Select this setting to enable editing of the Proposal email template.
Enable Proposal Summary	Select this setting to enable the proposal summary object.
Enable Submit Merge Call	Select this setting to submit merge calls for processing. If this setting is enabled, the **Submit** button is displayed during document generation.
ESignatureStrictMode	RESERVED Select this setting to disable buttons for Select Attachments / Recipients.
Instance Url	Enter the Salesforce instance URL.
Large Doc Process Batch Size	Indicates the number of repeating child items to process in a single batch transaction. This setting is applicable for *Enterprise* quotes. The default batch size for order line item creation and order activation is 100 unique bundle lines. This means 100 unique products on the cart (each product can have multiple line items).
Large Doc Threshold	Enter a number of the line items in the proposal when exceeded indicates that the proposal is a large document. Enter *-1* to deactivate the threshold.

Setting	Description
Max Child Level	Enter the maximum level to generate the merge data for the proposal.
Merge Call Timeout Millis	Enter the timeout in milliseconds for the merge request. For example: 60,000
Merge Webservice Endpoint	Enter the Apttus merge webservice endpoint.
PDF Owner Password	Enter the password required to change permissions of the PDF document like printing or editing.
Sync Bundle Using Line Items	Select this setting to use proposal line items to synchronize bundle products. By default, CPQ uses proposal summary objects. This setting appears only if you have Conga Quote Configuration Integration package installed. ⓘ This field is not supported for the quotes where **QTC Profile** is set to *Split*. Regardless of the value of the field, proposal line items are used to sync Spilt Quote with Opportunity.
Sync Option Products	Select this setting to enable CPQ to synchronize the option products with the bundle. This setting synchronizes options with bundles when you click **Sync with Opportunity** for your proposal. This setting appears only if you have Conga Quote Configuration Integration package installed.

Configuring Custom Settings for Different Flows

You can create different data set of Config System Properties and Installed Product Settings for different flows in the org. When you create such a data set, the definition of these properties in the data set is only valid in the Quotes configured using that flow. You must create the data set with the same name as the flow.

The following fields can be defined differently for different flow:

- Split Cart Threshold

- Split Cart Criteria Fields
- CSS Override
- Service Line Split Criteria
- Constraint Rule Execution Mode
- Pricing Execution Mode
- Custom Pre Pricing Fields
- Base Price Defaulting Method For Renewal
- Hide Asset Actions
- Check Many Options

To create a new data set for Config System Properties

1. Go to **Setup > App Setup > Develop > Custom Settings.**
2. Click **Manage** next to Config System Property.
3. Click **New**.
4. In the **Name** field, enter the name of the flow for which you want to create the data set.
5. Configure the settings according to your needs. For information on the settings, refer to the topic Configuring Custom Settings.
6. Click **Save**.

For more information about configuring Installed Product Settings for a different flow, refer to Defining Allowed Actions for Assets.

About Custom Setting Order of Precedence

There are different 3 ways you can define a custom setting, from default custom setting dataset, flow-based custom setting dataset, and URL parameters of the **Configure Products** button. There is an order of precedence based on where the custom settings are defined that CPQ follows while executing them. The order of precedence of custom settings execution is described below.

- Settings defined in the default custom settings dataset are executed in absence of flow-based dataset or URL parameters. They are also executed if an invalid definition of the setting is configured in flow-based dataset or URL parameters.
- Settings defined in the flow-based custom setting dataset overrides the settings in the default custom settings dataset. These are executed in the flow they are defined for.
- Settings defined in the URL parameters in the **Configure Products** button override both default and flow-based custom settings dataset.

Configuring Field Set Settings

The Field Set Settings page allows the System Administrator to configure the field sets.

You can enter details in one or more of the following sections, based on your business requirement.

Setting	Description
Name	Select a Field Set from the drop-down list. The available Field Sets are as follows: • Collab Create Fields • Collab Merge Fields • Line Item Fields to Revalidate Invalid Price List Items • Line Item Fields to Revalidate Product Structures • Related Line Item Fields • Asset Line Fields For Selected Assets
FieldName	Select a field that you want to add to a Field Set. For example, you may want to add Record ID, Created Date to the Collab Create Fields. These fields are populated in the drop-down lists from the corresponding objects. For example, Asset Line Item object for the asset field set. Click **+** symbol to add a field and **-** symbol to remove a field from the field set. Click Up and Down arrows in front of each field to set a sequence in which fields to be displayed in the field set list. Click **Save** to set your configuration.

Each field set is meant for a specific business scenario.

Field Set	Scenario
Collab Create Fields	In a multi-tier collaboration scenario, specify the additional fields whose value you want to copy over from parent line item to child line item when creating a collaboration request. For more information, see About Quote Lifecycle Collaboration.
Collab Merge Fields	In a multi-tier collaboration scenario, specify the additional fields whose value you want to copy over from child line item to parent line item when merging a collaboration request. For more information, see About Quote Lifecycle Collaboration.
Line Item Fields to Revalidate Invalid Price List Items	When you select the Price List Item field for the Line Item Fields to Revalidate Product Structures set, then you can have a validation based on the price list item. For more information, see About Quote Lifecycle Collaboration.
Line Item Fields to Revalidate Product Structures	When you select the Price List Item field for the Line Item Fields to Revalidate Product Structures set, then you can have a validation based on the price list item. For more information, see Revalidating the Product Configuration.

Field Set	Scenario
Related Line Item Fields	Select the fields that you want to display on the Related Line Items pop-up. This pop-up is displayed when the user hovers the mouse on a service name (bundle or option), clicks the more icon ( ), and selects **Related Line Items** on the Cart. For example, when you select the **Weightage Type** field for the Related Line Item Fields set, then you can define the **Percentage** and **Amount** values for the Related Asset Line Items pop-up. The Related Asset Line Items are associated with a Service Product. For more information on the Service CPQ, see Managing Services.
Asset Line Fields For Selected Assets	Select the fields from Asset Line Item object that you want to display on the Selected Assets or Manage *<Service_Option_Name>* pop-up. You can display a maximum of 10 fields in the pop-up. This pop-up is displayed when the user clicks the **Selected Assets** link on the Service Catalog or when the user selects a service option and clicks the icon on the Service Config page. The asset name is displayed next the checkbox on the Manage *<Service_Option_Name>* pop-up by default. You need not add the **Asset Name** field in the **Asset Line Fields For Selected Assets** setting. For example, when you select the **Location** field for the **Asset Line Fields for Selected Assets** setting, then the user can view the location at which the product was sold to the customer. For more information on the Service CPQ, see Managing Services.

Smart Search Setup

You can use the Smart Search feature to search for products in large and complex product catalogs, with multiple levels of categories and stricter rules. The smart search typeahead mechanism and faceted search helps narrow the search results and still honor product visibility and availability rules.

You must setup remote site before you configure settings for Smart Search.

Setting up Remote Site Settings

Navigate to **Setup > Administration Setup > Security Controls > Remote Site Settings** and click **New Remote Site** to create the following records:

Record number	Remote Site Name	Remote Site URL
1	SmartSearch	https:// *<SMART_SEARCH_URL>* Smart Search URL can be any of the following: • AZURE PREVIEW: smartsearch-pre-appgw.apttuscloud.io (Nov 2016 SP1 GA - Version 9.1168 and above versions) For any fresh package installation, Azure URL defaults.
2	SFDC	• For production environment, enter https://login.salesforce.com • For sandbox environment, enter https://test.salesforce.com
3	OauthToken	• For production environment, enter https://login.salesforce.com/services/oauth2/token • For sandbox environment, enter https://test.salesforce.com/services/oauth2/token

Creating a Certificate

Navigate to **Setup > Administration Setup > Security Controls > Certificate and Key Management** and click **Create Self-Signed Certificate** to create a new certificate with the following details:

1. Enter *JWT* as the label for the certificate. The Unique Name defaults to *JWT.*
2. Select the **Exportable Private Key** checkbox.

3. Choose Key Size as *2048*.
4. Select the **Active** checkbox.
5. Click **Save**.

Once you have created a certificate, go to your certificate record and click Download Certificate to store the certificate on your local system. You will need the certificate while creating a connected app.

Creating a Connected App

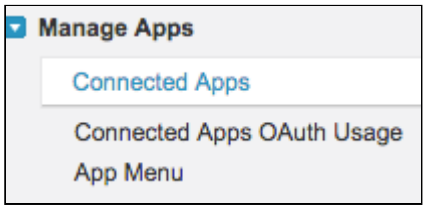
1. Navigate to **Setup > App Setup > Create > Apps.**
2. Scroll down and search for **Connected Apps** related list. Click **New** to create a new app.
3. Enter the basic information, **Connected App Name** and **API Name** as *ApttusSmartSearch.*
4. In the API (Enable OAuth Settings) section, select **Enable OAuth Settings** checkbox.
5. In the Callback URL, enter https://***<login OR test>***.salesforce.com/services/authcallback/***<org-id>***/ApttusSmartSearch
   In ***<login OR test>*** enter login or test depending on your environment. In **<org-id>**, enter your org ID.
6. Select the **Use digital signatures** check box. Choose the file from generated certificate.
7. For **Selected OAuth Scopes**, select the following:
   - Access and manage your data (api)
   - Access and manage your Chatter data (chatter_api)
   - Perform requests on your behalf at any time (refresh_token, offline_access)
8. Click **Save**.

Approving your Connected App

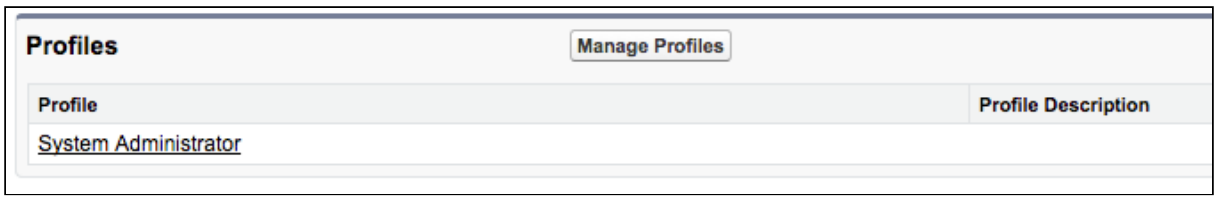
Before approving your connected app, you must decide which Oauth policy you would like to have. This policy decides which permitted users can work on Smart Search's admin page. Admin approved users are pre-authorized. You can also authorize other users and for such users, the steps mentioned in Approving your Connected App procedure are mandatory.

Step 1: Managing Profiles for Connected App

1. Navigate to **Setup > Administration setup > Manage Apps > Connected Apps**.



2. Next to the ApttusSmartSearch connected app record, click **Edit.**
3. Under the OAuth Policies section, in Permitted Users drop-down list, choose *Admin Approved Users are Preauthorized* and click **Save**.
4. Scroll down and search for Profiles section. Click **Manage Profiles** and select the profiles that you want to give access to the Smart Search App.



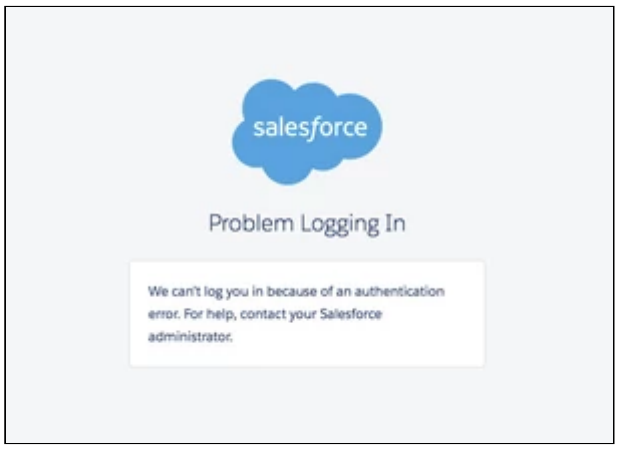
5. Click **Save.**

Step 2: Approving your Connected App

1. Navigate to **Setup > App Setup > Create > Apps.**
2. Scroll down and search for **Connected Apps** related list.
   Click your *ApttusSmartSearch* record and save the values of the Consumer Key, Consumer Secret, and Callback URL fields in a text file. You will need these keys to set up the last step for smart search.
3. Go to your browser URL, paste the following URL and click **Enter**:
   https://***<login OR test>***.salesforce.com/services/oauth2/authorize?response_type=code&client_id=**<consumer-key>**&redirect_uri=**<callback-url>**

   Use the correct domain for **<login or test>**, enter your consumer key and callback URL which you have fetched in the above step in **<consumer-key>** and **<callback-url>**. Note that `<callback-url>` needs to be encoded.

4. Click **Allow** to give permission to your connected app.
5. You will see the following page, which is the expected behavior of the app. You can now close the browser tab.



Smart Search Settings

You must follow the steps below to configure Smart Search.

> ⚠ Please contact Conga Product Support team to enable the Smart Search capability for your org.

Prerequisite

Setup the remote site for Smart Search. Refer to Smart Search Setup.

Smart Search Config Settings

Go to **Config Settings** > **Smart Search Settings**. This page lists all the smart search specific settings that you need to set up.

Setup Apttus Intelligent Cloud Search Connection Settings

Please contact Conga Product Support/PS/CSM team to obtain the URL and API User Key.

Setting	Description
Enhanced Search URL	Specify the URL end point for enhanced product search. With **Enhanced Search URL** populated with the correct URL, the system synchronizes all the changes at every hour and displays the results in the **Review Sync Status** section. If you do not specify the **Enhanced Search URL**, the system neither performs the sync nor updates the information in the **Review Sync Status** section.
Api User Key	Specify the API key that consists of the client ID and is used for smart search related activities. This key is required before Search Activation/Deactivation and resync. An error is displayed if you activate, deactivate or resync without the API User Key.

Add and Review Product Field Weightage

The weightage allocation enables the Smart Search to control the ranking of the search results based on customer needs. The total for the weightage is 100% and this can be split into any number of Product Fields that participate in the search criteria. For example, 70% weightage to product name and 30% weightage to product code means that if for a search text there is a match with a product name as well as a product code, the product name match will be ranked higher in the result. This capability allows the customers to fine tune the search results and also include custom fields that may be included in the search criteria.

To define the criteria, the admin is required to define the map as follows. The map of fields and corresponding weightage can be different for each flow.

1. Select the desired flow using **Flow** drop down.
2. Select the desired field using **Field Name** drop down. Select the desired weightage in percentage using **Weightage** drop down.
3. Click **Add Row**, if you need to add an additional field. You can add product fields such as Product Name, Description, Product Code, and more. Ensure that total weightage sum of all the fields equals to 100%.
   The system supports fields of type text, picklist, rich text, and number. It can also be a mix of all these types of fields. You can add a maximum of 6 fields.
4. Click **Save**.

Push Data into Search Engine

This is a one-time Activation step. This enables the system to set up new indexes and push the product data that runs the catalog and search.

Click **Search Activation.**

Refresh Data in Search Engine

This step is required to be executed any time there is change to the base line setup or “search meta-data”. This includes:

- New fields are added to the weightage criteria.
- New fields are added to the Refine Search filter.
- New fields are added to the product catalog listings.

Click **Search Metadata Refresh.** You need to refresh the metadata only when you have made changes to the product schema or data.

Remove Data in Search Engine

This step is required when you want to deactivate the Smart Search functionality in your org and purge data from our servers.

Click **Deactivate Search**.

Review Sync Status

This section displays the details about the last Sync job performed for Smart Search Settings. The fields such as **Sync Message**, **Last Attempted Sync**, **Last Successful Sync** and **Current Sync Status** depict the necessary information about the Sync job. The Review Sync Status is changed when you activate Smart Search (by clicking **Search Activation**), run the batch job and deactivate the Smart Search (by clicking **Deactivate Search**).

Switch to Azure Smart Search

You can now switch the search engine from *Solr* to *Azure* for faster search results. Create a new record under **App Setup > Develop > Custom Settings > Config Smart Search Settings > Manage** with name *SmartSearchSync.* Inside this record, specify the **Search Environment** as *Azure.*

> ⚠ **Note**

By default, the value in **Search Environment** is set to *Azure* for new package installation. For upgrade (if you are already using Smart Search), you must deactivate the smart search, change the search environment and then activate the smart search.

Upgrading CPQ

This section provides information on upgrading CPQ to the latest version from the previous two releases.

If you have not installed CPQ, you can contact Conga Support to request for an installation link, then perform the standard installation as described in Installing CPQ Packages.
Before you upgrade the CPQ packages in your org to May '22 release, please note the following points:

- If the CPQ package in your org is on any of the following releases, you must upgrade it to the Winter '20 patch (Winter20.02.26) before upgrading it to May '22 release:

Release	Package
Spring '20 or earlier release	Packages older than **12.0.1719 | 12.1719**
Summer '20 patch older than Summer20.02.26	Packages older than **12.1.1787.99 | 12.1787.99**
Winter '20 patch older than Winter20.02.26	Packages older than **12.2.1839.48 | 12.1839.48**

- If the CPQ package in your org is on any of the following releases, you can directly upgrade it to May '22 release:

Release	Package
Latest Summer '20 patch (Summer20.02.26 or later)	**12.0.1719 | 12.1719** or later
Latest Winter '20 patch (Winter20.02.26 or later)	**12.2.1839.48 | 12.1839.48** or later
Spring '21 and any Spring '21 patches	**13.0.1882 | 13.1882** or later
Summer '21 and any Summer '21 patches	**13.1.1921 | 13.1921** or later

Release	Package
December '21 and any December '21 patches	**13.2.1969 | 13.1969** or later

- Preparing for Upgrade
- Upgrading to CPQ May '22 Release
- Performing the Post-Upgrade Tasks
- Preparing for Upgrade (UDJS)

Preparing for Upgrade

Before you upgrade to CPQ May '22 release, you must ensure the following:

The Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, It ensures all the Conga published managed packages are on the latest versions for the registered orgs. To register your org for push upgrade, see Registering for Conga Push Upgrade.

- You go through CPQ Features by Release to know about the new features, enhancements, and deprecated features in CPQ since your existing release. After you upgrade CPQ to May '22 release, you cannot roll back to any previous release.
- You have the supported platforms and system requirements.
- You have access to the **Install Center** on the Conga Community Portal.
- You have administrator privileges to your Salesforce org.
- You need not back up your configurations. All configurations you performed since you installed your existing release will remain intact after the upgrade.
- You must delete the **UsageDataJobScheduler** job once before upgrading to May22.09.27 release or later.

To delete the UsageDataJobScheduler job

Before you upgrade **Conga Base Library** for May22.09.27 release or any succeeding release, perform the following steps.

1. Go to **Setup** > **Administration Setup** > **Monitoring** > **Apex Jobs**.
2. Search for the **UsageDataJobScheduler** job, CPQ displays the status of the job as *Queued*.

3. Open another instance of the org in a different browser tab, and go to **Setup** > **Administration Setup** > **Monitoring** > **Scheduled Jobs**.
4. Search and delete the **UsageDataJobScheduler** job from Scheduled Jobs. On the Apex Jobs page, the status of the **UsageDataJobScheduler** job should now change to *Aborted*.
5. Install the package version **3.0.233.2 |** 3.233.2 or higher of the **Conga Base Library** package.
6. After the latest package is installed, check if the **UsageDataJobScheduler** job gets scheduled again on both the Apex Jobs and Scheduled Jobs pages. If not, the latest package has successfully disabled the scheduling of this job and has deprecated it.

Upgrading to CPQ May '22 Release

This section describes step-by-step instructions to upgrade from Summer '21 and December '21 to May '22 release.

The Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, It ensures all the Conga published managed packages are on the latest versions for the registered orgs. To register your org for push upgrade, see Registering for Conga Push Upgrade.

Upgrading CPQ from December '21 to May '22 Release

1. Go to **Setup** > **Installed Packages** and ensure that your current Salesforce org has the packages of CPQ December '21 release or later, which are the *minimum* required versions for upgrading to May '22 release. Refer to the latest December '21 release or respective patch Release Notes.
2. Ensure that you have the following packages and dependent packages to upgrade to May '22 release. These packages are required to utilize the new features and enhancements of May '22 release. These are the *minimum* required versions; later versions are also supported.

Package	**Latest Certified Version** *(Name / Number)*
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.1 | 13.300.1

Package	Latest Certified Version *(Name / Number)*
Conga Base Library **(New)** (Required if you are using Conga Configuration & Pricing)	3.0.232 | 3.232
Conga Billing **(New)** (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration **(New)** (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing **(New)**	14.0.1995 | 14.1995
Conga Contract Lifecycle Management **(New)**	13.0.669 | 13.669
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer **(New)**	8.0.0029 | 8.29
Conga CPQ Setup **(New)** (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup **(New)** (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer **(New)**	7.0.0007 | 7.7
Conga DocuSign Api **(New)**	8.0.0117 | 8.117

Package	Latest Certified Version *(Name / Number)*
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration **(New)**	11.0.0079 | 11.79
Conga Quote Configuration Integration **(New)** (Required if you are using CPQ and Proposal Management)	14.0.0391 | 14.391
Conga Quote DocuSign Integration **(New)**	4.0.0021 | 4.21
Conga Quote Echosign Integration **(New)** (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16
Conga Quote Management **(New)** (Required if you are using Proposal Management)	12.0.0256 | 12.256

3. Perform the upgrade. The upgrade procedure is the same as the installation procedure. For detailed information on installing CPQ managed packages, see Installing CPQ Packages.
4. After the upgrade is complete, perform the post-upgrade tasks.

Upgrading CPQ from Summer '21 to May '22 Release

1. Go to **Setup** > **Installed Packages** and ensure that your current Salesforce org has the packages of CPQ Summer '21 release or later, which are the *minimum* required

versions for upgrading to May '22 release. Refer to the latest Summer '21 release or respective patch Release Notes.

2. Ensure that you have the following packages and dependent packages to upgrade to May '22 release. These packages are required to utilize the new features and enhancements of May '22 release. These are the *minimum* required versions; later versions are also supported.

Package	**Latest Certified Version** ***(Name / Number)***
Conga Approvals **(New)** (Required if you are using Approvals)	13.0.300.1 | 13.300.1
Conga Base Library **(New)** (Required if you are using Conga Configuration & Pricing)	3.0.232 | 3.232
Conga Billing **(New)** (Required if you are using Conga Billing)	8.0.296 | 8.296
Conga CLM Configuration Integration **(New)** (Required if you are using CLM)	14.0.0173 | 14.173
Conga Configuration & Pricing **(New)**	14.0.1995 | 14.1995
Conga Contract Lifecycle Management **(New)**	13.0.669 | 13.669
Conga CPQ Api	13.2.0127 | 13.127
Conga CPQ Approvals (Required if you are using Approvals on CPQ objects)	12.1.0024 | 12.24
Conga CPQ Maximizer **(New)**	8.0.0029 | 8.29

Package	Latest Certified Version *(Name / Number)*
Conga CPQ Setup **(New)** (Required if you are using CPQ Admin Console)	14.0.128 | 14.128
Conga Deal Maximizer Setup **(New)** (Required if you are using Deal Maximizer)	8.0.0052 | 8.52
Conga Deal Maximizer **(New)**	7.0.0007 | 7.7
Conga DocuSign Api **(New)**	8.0.0117 | 8.117
Conga Order Management	1.0.0000 | 1.0
Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
Conga Quote Approvals	6.5.0005 | 6.5
Conga Quote Asset Integration	6.5.0014 | 6.14
Conga Quote CLM Integration **(New)**	11.0.0079 | 11.79
Conga Quote Configuration Integration **(New)** (Required if you are using CPQ and Proposal Management)	14.0.0391 | 14.391
Conga Quote DocuSign Integration **(New)**	4.0.0021 | 4.21
Conga Quote Echosign Integration **(New)** (You must install the latest Adobe Sign package to use this package)	3.0.16 | 3.16

Package	Latest Certified Version *(Name / Number)*
Conga Quote Management **(New)** (Required if you are using Proposal Management)	12.0.0256 | 12.256

3. Perform the upgrade. The upgrade procedure is the same as the installation procedure. For detailed information on installing CPQ managed packages, see Installing CPQ Packages.
4. After the upgrade is complete, perform the post-upgrade tasks.

Performing the Post-Upgrade Tasks

After you upgrade to this CPQ release, consider the following options and requirements:

Object Maintenance

After upgrading your packages, you must run the object maintenance tasks to synchronize all your previous configurations.

To access the maintenance pages, click the **All Tabs** icon (+) to display all tabs and click the link for each required maintenance. When you run maintenance jobs after an upgrade, you must run them in the order below, in the instances described for each:

- **Product Filter Maintenance** - After every upgrade, you must run this task, which collects all product filter field values related to a category. The maintenance job updates the required records for all the categories every time you run it. You must also run the maintenance task every time there is a change in products or a change in the filter field values.
- **Category Maintenance** - After any product association, removal or hierarchy change, you must run a Category Maintenance job. This de-normalizes the hierarchy into a custom object for reporting and totaling purposes. If category maintenance is not run, the end-user may see incorrect totals on the shopping cart page.
- **Constraint Rule Maintenance** - This should be run after any changes are made to Constraint Rule conditions. If it isn't run, products may not be included or excluded as expected.
- **Bundle Maintenance** - When options are added or removed from a bundle, you must use the Update Bundle Components page to run a bundle maintenance job that synchronizes all the bundles and options.

- When you run bundle maintenance for all bundles, be sure to only click the button once to avoid scheduling multiple unnecessary bundle maintenance jobs. To track the progress of the job, you should go to **Setup > Administration Setup > Monitoring > Apex Jobs**. You see the bundle jobs and whether they have been completed.
- **Attribute Maintenance** - After an attribute group is associated with a category, run this job to associate the attribute with the products that belong to the category.
- **Criteria Maintenance** - After any pricing change (rules, dimensions or any criteria change), you must run a Criteria Maintenance job.

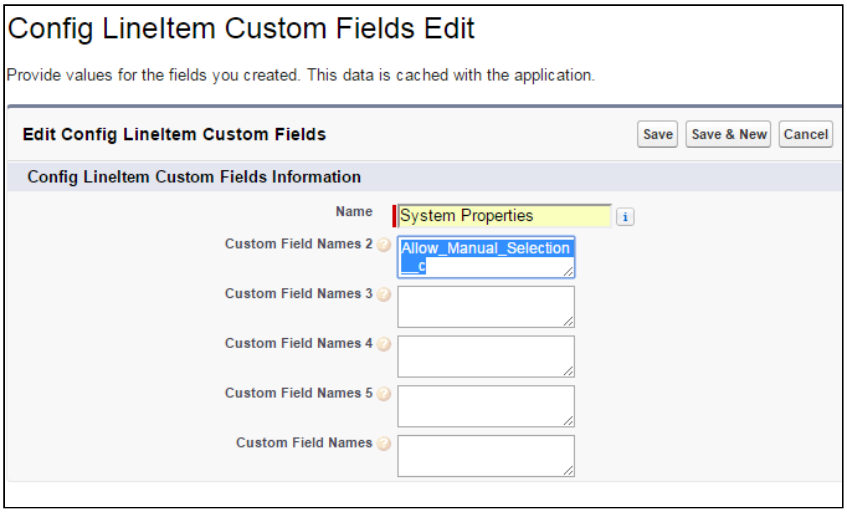
  > Ensure that you run the Criteria Maintenance job only if you have active rulesets or have made any updates to the active rulesets.

Adding Custom Fields to Config LineItem Custom Fields

If your implementation uses the custom callback class related to Pricing, Related Pricing, Asset Line Item, and Validation, then the API names of the custom fields referred in the custom callback class must be defined in the following custom setting:

Config LineItem Custom Fields

If the length of the text field exceeds the first column, use the next text field for this.



> Do not append namespace before the field name in case you are entering the API names of product type fields in any of the Custom Field Name fields. The following table shows the correct and incorrect format of the API names.

Correct	Incorrect
Apttus_Config2_ProductIdr.ProductType_c	Apttus_Config2_ProductIdr.Apttus_Config2ProductType_c

Correct	Incorrect
Apttus_Config2_OptionIdr.ProductType_c	Apttus_Config2_OptionIdr.Apttus_Config2ProductType_c

Custom Labels for Custom Actions

After upgrading, it is recommended that you create custom labels for any custom actions. Note that you must do it for each flow or action combination, otherwise none of the buttons are displayed on the cart.

For more information, see Configuring Display Actions Settings.

Preparing for Upgrade (UDJS)

Before you upgrade to CPQ May '22 release, you must ensure the following:

The Conga Push Upgrade is an automated tool that upgrades packages available in your Salesforce org (Production or Sandbox) to the latest versions. In addition, It ensures all the Conga published managed packages are on the latest versions for the registered orgs. To register your org for push upgrade, see Registering for Conga Push Upgrade.

- You go through CPQ Features by Release to know about the new features, enhancements, and deprecated features in CPQ since your existing release. After you upgrade CPQ to May '22 release, you cannot roll back to any previous release.
- You have the supported platforms and system requirements.
- You have access to the **Install Center** on the Conga Community Portal.
- You have administrator privileges to your Salesforce org.
- You need not back up your configurations. All configurations you performed since you installed your existing release will remain intact after the upgrade.
- You must delete the **UsageDataJobScheduler** job because it is no longer supported and may cause an error.

To delete the UsageDataJobScheduler job

The Apex Jobs page displays that the **UsageDataJobScheduler** job failed with an error. This happens because the **UsageDataJobScheduler** job is no longer supported. Before you

upgrade **Conga Base Library** for this release, perform the following steps to avoid the error.

1. Go to **Setup** > **Administration Setup** > **Monitoring** > **Apex Jobs.**
2. Search for the **UsageDataJobScheduler** job, CPQ displays the status of the job as *Queued*.
3. Open another instance of the org in a different browser tab, and go to **Setup** > **Administration Setup** > **Monitoring** > **Scheduled Jobs**.
4. Search and delete the **UsageDataJobScheduler** job from Scheduled Jobs. On the Apex Jobs page, the status of the **UsageDataJobScheduler** job should now change to *Aborted*.
5. Install the latest package of **Conga Base Library**. For more information on the package version name and number, see Upgrading CPQ.
6. After the latest package is installed, check if the **UsageDataJobScheduler** job gets scheduled again on both the Apex Jobs and Scheduled Jobs pages. If not, the latest package has successfully disabled the scheduling of this job and has deprecated it.

Creating an Opportunity in Salesforce

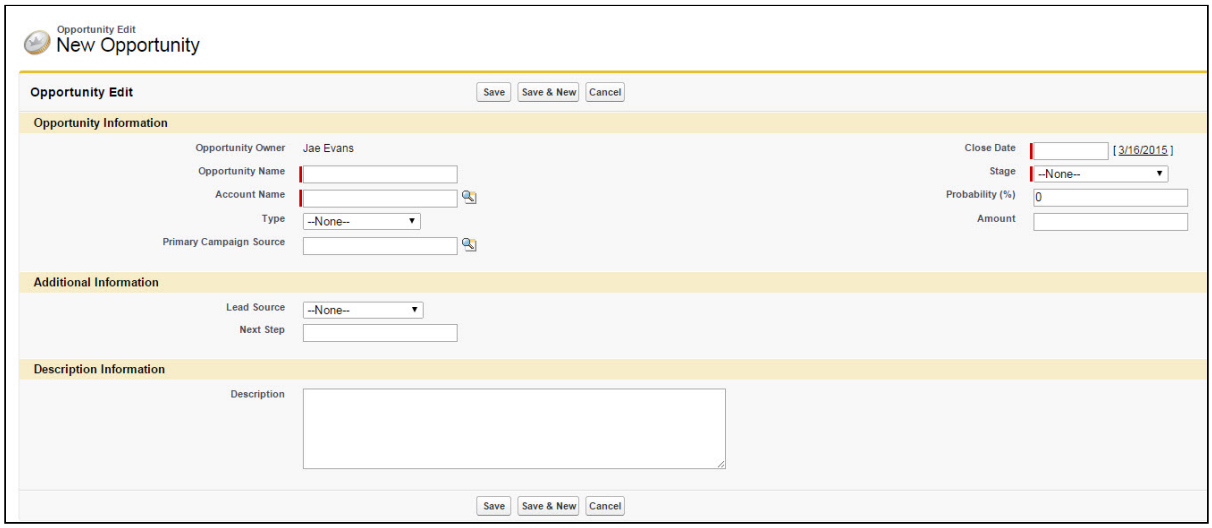
Opportunities are the qualified contacts or accounts that you have already talked to and have entered into your sales cycle. Adding opportunities to Salesforce builds your pipeline and increases your sales forecast. You will use the Salesforce Opportunities tab to create and find your opportunities.

The CPQ process begins as soon as you create an opportunity.

> ⓘ Opportunities are standard Salesforce objects, not custom objects. For more information on opportunities, see Salesforce documentation.

To create an opportunity

1. Log in to Salesforce.com.
2. Navigate to the **Opportunities** tab, and click **New**.



3. Enter the **Opportunity Name, Account Name, Close Date,** and **Stage**.
4. Enter additional information as required.
5. Click **Save**.

You have created an opportunity. For complete information about creating opportunities in Salesforce, refer to Create Opportunities and Opportunity Fields in the Salesforce documentation.

Creating a Shopping Cart Experience

A CPQ administrator configures CPQ to create a shopping cart experience for the user.

Consider that ABC company sells products related to computers. Products including monitor, keyboard, mouse, Operating System, tools are sold by the ABC company. Each of these products can be sold as a stand-alone product or together as a bundle. For example, all of these products can be integrated as a bundle called computer where each of these products are options. Furthermore, each product has attributes associated with it. Attributes are features for a product. For example, attributes for a laptop are color, RAM, Operating System used and so on.You can configure attributes to derive pricing or enhance search options.

Each of these products has a price associated with it. Pricing can be different for each of these products sold as a stand-alone or when you buy a computer.

A sales representative can simply select a product when it is categorized rather than to search for the product from the huge array of products. A shopping cart experience for a user is enhanced when the user gets to search for his products based on some search criteria. The search criteria can be based on features of a products.

The sales of a product can be improved if user is advised to buy another product when he selects one product. For example, if a user buys a monitor, he will also be interested to buy a keyboard.

Consider that there is a holiday and the ABC company decides to promote a new product. User get discount of x% for all products from October to December.

All of this is configured by using CPQ.

Perform the following to configure CPQ:

1. Defining the products and services
2. Configure price for products
3. Configure rules that define pricing discounts and offers
4. Configure rules that guide user to buy products
5. Define search mechanism to search based on attributes
6. Run the maintenance jobs to update all changes made to products, bundle, category, and pricing.

Running Maintenance Jobs

As an administrator, you can run the following maintenance jobs.

Job	Definition
Custom Setting Maintenance	This job generates static resources for custom settings.
Category Maintenance	This job creates a de-normalized view for Categories with related products.
Criteria Maintenance • Update Pricing Fields • Update Expression Fields • Update Constraint Fields	These jobs optimize the query time by creating indexes for the fields that are referred to in the criteria or expressions.
Bundle Maintenance	This job synchronizes all the bundles and options.

You must execute an appropriate maintenance job after applying structural changes to the definition of categories, products, or criteria to ensure that the Sales rep does not see

stale definition in CPQ application. A stale definition may cause errors in the console or application.

Prerequisites

- Go to **Setup** > **Administration Setup** > **Email Administration** > **Deliverability**. In the **Access to Send Email** section, ensure that the **Access Level** is set to **All Email**.
- Define **Update View Product Batch Size** and **Update View Category Batch Size** in Config System Properties.

To run a Custom Settings Maintenance job

CPQ uses custom settings to process computation on various CPQ pages. The Custom Settings Maintenance job generates static resources of the custom settings to reduce the time to fetch the values. Because the values are static, you must run the Custom Settings Maintenance Job every time you make any modification in the custom settings. Otherwise, the changes may not reflect in CPQ and the computations may get processed with stale values, resulting in incorrect configuration and pricing.

1. Go to All Tabs () > **Custom Settings Maintenance**.
2. Select the flow you want to run the maintenance for or select *All* to run maintenance for all the flows.
3. Click **Update**.

To run a Category Maintenance job

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Category Maintenance**. The Category Maintenance window is displayed.
3. From the **Hierarchy** drop-down, select a particular hierarchy or *All* and click **Update View**. After the maintenance job is 100% complete, the **Status** column under **Batch Jobs** changes to Completed.
4. Close the Category Maintenance window.

This executes an asynchronous batch job that maintains the hierarchy. After you click **Update View**, the administration task is complete and an updated history for all the batch jobs is displayed.

To use manual incremental update for category views

After making changes to the existing category definition you can run the Category Maintenance batch job only for changes rather than running for the entire definition. For example, if your category has 10 products and you added 5 more, you can run the Category Maintenance batch job only for the 5 additional products. To execute the Category Maintenance batch job for the changes, you must define an Admin Property that is used by CPQ to isolate the changes in the category.

Perform the following steps to add an admin entry:

1. Click the All tabs icon () and click **Admin**.
2. Click New to create a new Admin entry.
3. In the **Name** field, enter *APTS_UpdateViewProductClasses*. Leave the fields **Value** and **Code** blank.
4. Click **Save**.

Now, if you change the category definition, the **Code** field in the Admin property is populated with the Product Classification records of the newly added products. These records are created after you associate products to the category. When you execute the Category Maintenance batch job, the maintenance batch job is only executed for the changes and not the entire definition. In case, you want to execute a maintenance batch job for the entire definition, clear the **Code** field in the Admin Entry and then execute the Category Maintenance batch Job.

> You must run the Category Maintenance batch job after every CPQ version upgrade.

To specify the batch size for the Category Maintenance batch job

You can use the *APTS_UpdateViewApiBatchSize* admin entry to specify the batch size for the Category Maintenance batch job when the job is executed using the exposed global Apex method *CPQWebService.updateHierarchyViews()*. The default batch size for the Category Maintenance batch job set in this global method is 2000, which is used when you have not defined the *APTS_UpdateViewApiBatchSize* admin entry. The behavior of the Category Maintenance batch job is same when the job is executed from the UI.

You must specify a batch size value only when you notice limit exceptions in batch jobs because of complex categories with a lot of hierarchy levels. In such cases, you must decrease the batch size to execute the batch jobs successfully.

> The maximum batch size for the Category Maintenance batch job set in the *CPQWebService.updateHierarchyViews()* global method is 2000. You can specify any number between 1 to 2000 in the *APTS_UpdateViewApiBatchSize* admin entry depending on the number of categories in the org. If you specify a value greater than 2000, CPQ will use 2000 as default. The actual batch size value to be used depends on the complexity of the categories and hence is specific to your requirements. You must arrive at the exact number through the trial and error method.

Perform the following steps to add an admin entry:

1. Click the All tabs icon () and click **Admin**.
2. Click **New** to create an admin entry.
3. In the **Name** field, enter *APTS_UpdateViewApiBatchSize*.
4. In the **Value** field, enter a value between 1 and 2000.
5. Leave the **Code** field blank.
6. Click **Save**.

To run a Criteria Maintenance job

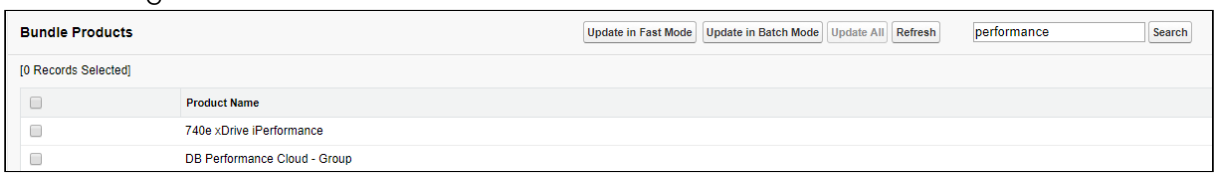
1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. Click the more icon () on the top-right corner and select **Criteria Maintenance**. The Criteria Maintenance window is displayed.
3. Click one of the following buttons.
   - **Update Pricing Fields**: Use this command to update all Pricing Criteria fields.
   - **Update Expression Fields**: Use this command to update all Expression Criteria fields.
   - **Update Constraint Fields**: Use this command to update all Constraint Criteria fields.

   After the maintenance job is 100% complete, the **Status** column under **Batch Jobs** changes to Completed.
4. Close the Criteria Maintenance window.

To run a Bundle Maintenance Job

1. Click All Tabs icon() and click **Bundle Maintenance**.

2. From **Product Name**, select the name of the parent bundle, whose sub-bundles have been changed.



3. Do one or more of the following:
   - To update the selected bundle, click **Update In Batch Mode**.
   - To update all the bundles in the list, click **Update All**.
   - To update basic bundles with a limited number of options, click **Update in Fast Mode**.

   > Run Bundle Maintenance in Batch Mode instead of Fast Mode if the bundle has a large number of options and sub-bundles. Bundle Maintenance in Fast mode can only be used for a limited number of options and sub-bundles. If that limit is exceeded an error is displayed.

   The system displays a confirmation message indicating that the process completed successfully.

The asynchronous task that maintains the product bundle structure is executed.

To run maintenance of view record for individual bundles

1. Navigate to the **CPQ Console** tab and click **Manage Products**.
2. Select an existing bundle to which you want to add or remove a bundle or an option.
3. Click **Product Console**.
4. From the **Catalog** section, click **Manage Bundles/Options**.
5. After making changes to the product association, click **Update View**.

The view record of the selected bundle is updated.

To run Attribute Maintenance Job

In the Classic UI, you must run the Attribute Maintenance job after you update attributes. You must execute this job after you associate attribute groups to a category or define a Product Attribute Rule (PAR) or Attribute Value Matrix (AVM) with criteria. It is not required to execute the job if you use CPQ Admin UI; CPQ executes the batch job automatically.

Follow the steps below in Classic UI:

1. Click All Tabs icon ( ) and click **Attribute Maintenance**.
2. Click one of the following buttons:

- **Update Attributes**: Use this command to update the attribute groups.
- **Update Attribute Rule Views**: Use this command to update PAR and AVM.

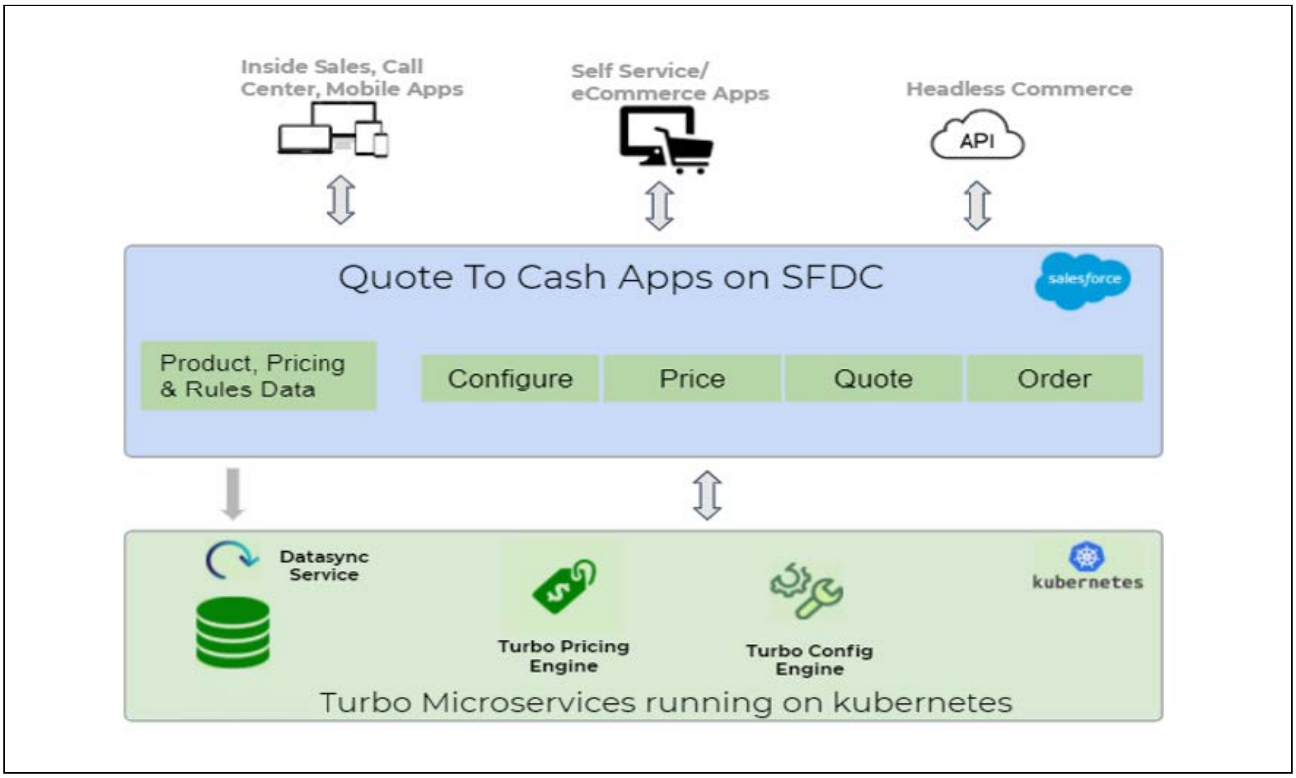
About TurboEngines

Conga TurboEngines is a concurrent processing engine provided by Conga that comprises various microservices that process product configurations (**TurboConfig**), pricing calculations (**TurboPricing**), and other product-related business data, such as promotions. Conga TurboEngines offload the computation workload from the Salesforce platform to the **Conga Flexible Compute Platform** to reduce the processing time on the cart. Processing the computation workload in the Conga Flexible Compute Platform reduces the interaction costs and the quote turnaround time specifically during peak load or large transactions.

TurboEngines scale on the following dimensions:

- Number of users
- Size of transaction
- The complexity of the product and rules

TurboEngines also provide a critical component called **TurboEngines Data Sync** services that provide a high-performance mechanism to sync pricing and config master data at regular, scheduled intervals (or on-demand) between Salesforce and the Conga Flexible Compute Platform. Data is pushed to TurboPricing and TurboConfig consumer endpoints and made available for processing to take advantage of the performance improvements offered by the TurboEngines platform.



For more information on TurboEngines, refer to TurboEngines documentation.

In this section following topics are described:

- About TurboConfig
- About TurboPricing

About TurboConfig

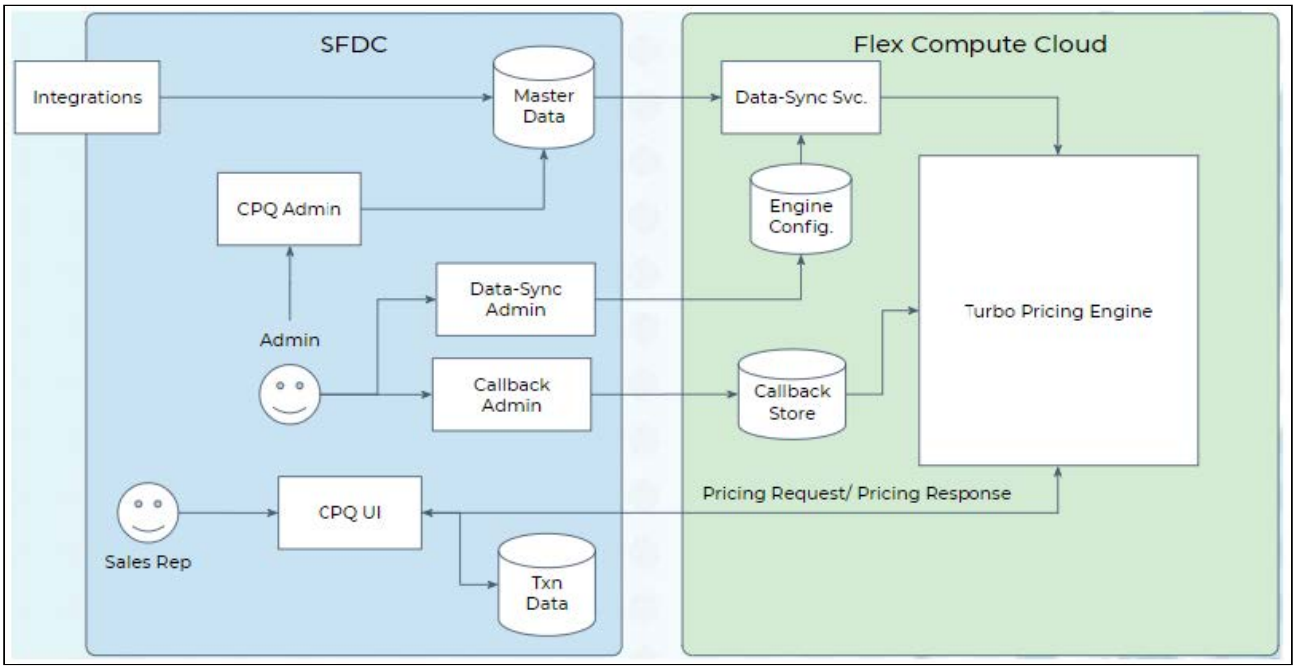
TurboConfig is a configuration engine created to process product configuration rules when products and bundles are configured on a cart and when finalizing the quote. TurboConfig offloads the computation workload from the Salesforce platform to the Conga Flexible Compute Platform built using microservices to reduce the processing time of the configuration rules. Computation workload includes the processing of rules defined on the products. For example, in a TurboConfig enabled flow, when the Sales rep adds the product or the favorite configuration to the cart, the constraint rules associated with them are offloaded to the Conga Flexible Compute Platform to process. TurboConfig engine executes the rules, maintains rule states, and avoids unnecessary line item processing.

TurboConfig is recommended when you have a large number of rules or highly complex configuration rules to be applied while selecting a product or configuring a bundle.

About TurboPricing

TurboPricing is a pricing engine built using microservices to reduce the processing time on the cart. You can enable TurboPricing to offload complex pricing computation workload from the Salesforce platform to the Conga Flexible Compute Platform. It reduces time to submit prices to customers, improves user experience, and improves user adoption with a more responsive user interface.

The following diagram shows how data flows between a Salesforce org and Conga Flexible Compute Platform:



Using Shield Platform Encryption

You can add a layer of security to the sensitive data in CPQ using Salesforce Shield Platform Encryption. CPQ support Salesforce Shield Platform Encryption with the following limitations:

- Encryption for number fields is not supported
- Encryption for account and contact is not supported

For more information, see How Shield Platform Encryption Works.

Enabling CPQ for Partner Community Users

This section describes how to enable CPQ for partner community users.

Prerequisites

- The *Partner Community User* profile must be available in your org at **Setup** > **Manage Users** > **Profiles**.
- The *Partner Community* user license must be available for *Partner Community User*.
- The **Enable Profile User Interface** setting must be enabled at **Setup** > **Manage Users** > **User Management Settings**.

To Enable CPQ for Partner Community users

1. Configure a profile.
2. Configure field-level security.
3. Configure permission set.
4. Share settings.
5. Configure partner community user.
6. Configure partner community site.
7. Configure a custom button on the quote.

Configuring a Profile

1. Go to **Setup** > **Manage Users** > **Profile**. The Profiles page is displayed.
2. Click the Partner Community User link. The Partner Community User page is displayed.
3. Click **Clone**. The Clone Profile page is displayed.
4. In the **Profile Name** field, enter a name for the profile. For example, Partner Community User Cloned.
5. Click **Save**. The Partner Community User Cloned profile is created. You must complete the following configuration.

Assigned Apps

1. Click **Assigned Apps**. The Assigned Apps page is displayed.
2. Click **Edit**, select the **Visible** checkbox for the following apps:

- Apttus Approvals Management (Apttus_Approval__ApttusApprovalsManagement)
- Apttus Approvals Setup (Apttus_Approval__ApttusApprovalsSetup)
- Apttus Batch Updates (Apttus_Config2__ApttusBatchUpdates)
- Apttus Billing Setup (Apttus_Config2__ApttusBillingSetup)
- Apttus Contract Management (Apttus__ApttusContractManagement)
- Apttus Contract Management (Lightning) (Apttus__ApttusContractManagement_Lightning)
- Apttus Contract Wizard (Apttus__ApttusConfigurableWizard)
- Apttus CPQ Admin (Apttus_CPQAdmin__ApttusCPQAdmin)
- Apttus Incentive Setup (Apttus_Config2__ApttusIncentiveSetup)
- Apttus Lightning Component Admin (Apttus_Base2__Configurator)
- Apttus Order Management (Apttus_Config2__ApttusOrderManagement)
- Apttus Pricing Setup (Apttus_Config2__ApttusPricingSetup)
- Apttus Product Setup (Apttus_Config2__ApttusProductSetup)
- Apttus Proposal Management (Apttus_Proposal__ApttusProposalManagement)
- Apttus Proposal Management (Lightning) (Apttus_Proposal__ApttusProposalManagement_Lightning)
- Apttus X-Author™ For Excel (Apttus_XApps__XAuthorExcel)
- TurboEngines Admin (Apttus_Base2__TurboEngineAdmin)

3. Click **Save**.

Object Settings

1. Click **Object Settings**. The Object Settings page is displayed.
2. Click the required object name link. The <object> page is displayed.
3. Click **Edit**. Configure the following settings:

Object Name	Tab Settings	Page Layouts	Object Permissions
Accounts	Default On	Varies by Record Type	Read, Create, Edit
Contacts	Default On	Varies by Record Type	Read, Create, Edit
DocGen Packages	Tab Hidden	Varies by Record Type	Read
Documents	Tab Hidden	--	Read
Opportunities	Default On	Varies by Record Type	Read, Create, Edit

Object Name	Tab Settings	Page Layouts	Object Permissions
Products	Default On	Product Layout (No Price)	Read
Proposals	Default On	Varies by Record Type	Read, Create, Edit

4. Click **Save**.

Apex Class Access

1. Click **Apex Class Access**. The Apex Class Access page is displayed.
2. Click **Edit**.
3. From the **Available Apex Classes** list, select CPQ-related classes.
4. Click **Add** to move them to the **Enabled Apex Classes** list.
5. Click **Save**.

Visualforce Page Access

1. Click **Visualforce Page Access**. The Visualforce Page Access page is displayed.
2. Click **Edit**.
3. From the **Available Visualforce Pages** list, select CPQ-related pages.
4. Click **Add** to move them to the **Enabled Visualforce Pages** list.
5. Click **Save**.

Custom Permissions

1. Click **Custom Permissions**. The Custom Permissions page is displayed.
2. Click **Edit**.
3. From the **Available Custom Permissions** list, select the following permissions:
    - Apttus.XA_CheckinAsFinal
    - Apttus.XA_CheckInWithoutRedlines
    - Apttus.XA_CheckInWithRedlines
    - Apttus.XA_Contracts
    - Apttus.XA_Template
    - Apttus.XA_UnlockReadOnlyClause
    - Apttus.XA_UnprotectDocument
4. Click **Add** to move them to the **Enabled Custom Permissions** list.
5. Click **Save**.

Custom Metadata Types

1. Click **Custom Metadata Types**. The Custom Metadata Types page is displayed.
2. Click **Edit**.
3. From the **Available Custom Metadata Types** list, select CPQ-related metadata types.
4. Click **Add** to move them to the **Enabled Custom Metadata Types** list.
5. Click **Save**.

Custom Setting Definitions

1. Click **Custom Setting Definitions**. The Custom Setting Definitions page is displayed.
2. Click **Edit**.
3. From the **Available Custom Setting Definitions** list, select the following and other CPQ-related definitions:
    - Apttus*.*
    - echosign_dev1.*
4. Click **Add** to move them to the **Enabled Custom Setting Definitions** list.
5. Click **Save**.

Configuring Field-Level Security

1. Go to **Setup** > **Create** > **Objects**. The Custom Objects page is displayed.
2. Click the Quote/Proposal object link. The Quote/Proposal (Managed) page is displayed.
3. In the **Custom Fields & Relationships** related list, click the Price List field link. The Price List (Managed) page is displayed.
4. Click **Set Field-Level Security**. The Set Field-Level Security page is displayed.
5. Select the **Visible** checkbox for the required profile and click **Save**. For example, Partner Community User Cloned.
6. Repeat step 3 to 5 for the following fields and other required fields for the Quote/Proposal object:
    - Approval Stage
    - Approval Status
    - Opportunity
    - Account
    - Proposal Name
7. Repeat step 2 to 6 for the CPQ-related custom objects.

ⓘ If you add or remove any field or object, you must configure field-level security for those fields and objects.

Configuring a Permission Set

1. Go to **Setup** > **Manage Users** > **Permission Sets**. The Permission Sets page is displayed.
2. Click **New**. The Create page is displayed.
3. Enter the following details:
    a. **Label**: Enter a label for the permission set. For example, CPQ Sales User – Partner.
    b. **API Name**: The field is automatically populated.
    c. **Description**: Enter a description for the permission set.
    d. **License**: Select the Partner Community license for this permission set.
4. Click **Save**. The CPQ Sales User – Partner permission set is created. You must complete the following configuration.

Assigned Apps

1. Click **Assigned Apps**. The Assigned Apps page is displayed.
2. Click **Edit** and select the **Visible** checkbox for the following apps:
    - Apttus_Proposal.Apttus Proposal Management (Apttus_Proposal__ApttusProposalManagement)
    - Apttus_Proposal.Apttus Proposal Management (Lightning) (Apttus_Proposal__ApttusProposalManagement_Lightning)
3. Click **Save**.

Object Settings

1. Click **Object Settings**. The Object Settings page is displayed.
2. Click the required object name link. The <object> page is displayed.
3. Click **Edit**. Configure the following settings:

Object Name	Object Permissions	Tab Settings
Accounts	Read	--
Adjustment Line Items	Read, Create, Edit, Delete	--
Admin	Read	--

Object Name	Object Permissions	Tab Settings
Agreement Action Conditions	Read	--
Agreement Clauses	Read, Create	--
Agreement Documents	Read, Create	--
Agreement Explorer	Read, Create, Edit, Delete	--
Agreement Line Items	Read, Create, Edit, Delete	--
Agreement Locks	Read	--
Agreement Price Rules	Read, Create, Edit, Delete	--
Agreement Price Tiers	Read, Create, Edit, Delete	--
Agreement Protection	Read	--
Agreement Rule Conditions	Read	--
Agreement Rules	Read	--
Agreements	Read, Create	--
Agreements	Read	--
Agreement Term Exceptions	Read, Create, Edit, Delete	--
App Assignments	Read	--
App Files	Read	--
Application Features	Read	--
Applied Expression Infos	Read, Create, Edit, Delete	--
Applied Rule Action Infos	Read, Create, Edit, Delete	--

Object Name	Object Permissions	Tab Settings
Applied Rule Infos	Read, Create, Edit, Delete	--
Apps	Read	--
Asset Attribute Value History	Read, Create, Edit, Delete	--
Asset Attribute Values	Read, Create, Edit, Delete	--
Asset Line Item History	Read, Create, Edit, Delete	--
Asset Line Items	Read, Create, Edit, Delete	--
Asset Transaction History	Read, Create, Edit, Delete	--
Asset Usage Price Tier History	Read, Create, Edit, Delete	--
Asset Usage Price Tiers	Read, Create, Edit, Delete	--
Async Merge Calls	Read, Create, Edit, Delete	--
Attribute Group Translations	Read, Create, Edit, Delete	--
Attribute Value Matrices	Read	--
Attribute Value Matrix Entries	Read, Create	--
Bundle Components	Read	--
Categories	Read	--
Category Hierarchies	Read	--
CategoryTranslations	Read	--
Charge Group Members	Read	--
Charge Groups	Read	--
Charge Types	Read	--

Object Name	Object Permissions	Tab Settings
Collaboration Requests	Read, Create, Edit, Delete	--
Constraint Rule Action Expressions	Read	--
Constraint Rule Actions	Read	--
Constraint Rule Conditions	Read	--
Constraint Rules	Read	--
Content Events	Read, Create, Edit, Delete	--
Cost Adjustments	Read	--
Cost Items	Read	--
Cost Line Items	Read	--
Cost Models	Read	--
Cycle Time Field Data	Read, Create, Edit	--
Cycle Time Fields	Read, Create, Edit	--
Cycle Time Group Data	Read, Create, Edit	--
Cycle Time Groups	Read, Create, Edit	--
Doc Assembly Components	Read	--
Doc Assembly Rules	Read	--
Doc Assembly Rulesets	Read	--
Document Agreement Clauses	Read	--

Object Name	Object Permissions	Tab Settings
Document Collate Infos	Read	--
Document Version Details	Read, Create, Edit	--
Document Versions	Read, Create, Edit	--
External Order Adjustment Items	Read, Create, Edit, Delete	--
External Order Summary	Read, Create, Edit, Delete	--
External Order Summary Items	Read, Create, Edit, Delete	--
Favorite Configurations	Read, Create, Edit, Delete	--
Features	Read	--
Feature Sets	Read	--
Field Expressions	Read	--
Footnote	Read	--
Formula Fields	Read, Edit	--
Formula Fields (Comply)	Read	--
Frequency/UOM Conversion Rates	Read, Create	--
Guided Questions	Read	--
Guided Search Rule Entries	Read	--
Guided Search Rule Filters	Read	--
Guided Selling Rules	Read	--
Help Docs	Read	--

Object Name	Object Permissions	Tab Settings
Line Item Rollups	Read, Create, Edit, Delete	--
Line Items	Read, Create, Edit, Delete	--
Merge Event Details	Read, Create, Edit, Delete	--
Merge Events	Read, Create, Edit, Delete	--
Opportunities	Read, Create, Edit	--
Payment Terms	Read	--
Permission Set Relationships	Read	--
Price Breakups	Read	--
Price Dimensions	Read	--
Price Escalators	Read	--
Price List Categories	Read	--
Price List Items	Read, Create, Edit	--
Price Lists	Read, Create, Edit	--
Price Matrices	Read	--
Price Matrix Entries	Read	--
Price Rule Entries	Read	--
Price Rules	Read	--
Price Rulesets	Read	--

Object Name	Object Permissions	Tab Settings
Product Attribute Group Members	Read, Create, Edit	--
Product Attribute Groups	Read, Create, Edit	--
Product Attribute Matrix Views	Read, Create, Edit	--
Product Attribute Rule Actions	Read, Create, Edit	--
Product Attribute Rules	Read, Create, Edit	--
Product Attribute Rule Views	Read, Create, Edit, Delete	--
Product Attributes	Read, Create, Edit	--
Product Attribute Values	Read, Create, Edit	--
Product Classifications	Read, Create, Edit	--
Product Configurations	Read, Create, Edit, Delete	--
Product Constraint Entries	Read, Create, Edit, Delete	--
Product Constraints--	Read, Create, Edit, Delete	--
Product Constraint Views	Read, Create, Edit, Delete	--
Product Default Rule Filters	Read, Create, Edit, Delete	--
Product Default Rules	Read	--
Product Default Values	Read, Create, Edit, Delete	--
Product Features	Read, Create, Edit, Delete	--
Product Feature Values	Read, Create, Edit, Delete	--
Product Filter Views	Read, Create, Edit, Delete	--

Object Name	Object Permissions	Tab Settings
Product Footnotes--	Read, Create, Edit	--
Product Group Members	Read, Create, Edit	--
Product Groups	Read, Create, Edit, Delete	--
Product Hierarchy Views	Read, Create, Edit, Delete	--
Product Information	Read, Create, Edit	--
Product Option Components	Read, Create, Edit	--
Product Option Groups	Read, Create, Edit	--
Product Option Prices	Read, Create, Edit	--
Products	Read	--
Product Translations--	Read, Create, Edit, Delete	--
Proposal Adjustment Line Items	Read, Create, Edit, Delete	--
Proposal Document Output Formats	Read	--
Proposal Footnotes	Read, Create, Edit, Delete	--
Proposal Incentive Benefit Data	Read, Create, Edit, Delete	--
Proposal Incentive Limit Data	Read, Create, Edit, Delete	--
Proposal Line Items	Read, Create, Edit, Delete	--
Proposal Location	Read, Create, Edit, Delete	--
Proposal Product Attribute Values	Read, Create, Edit, Delete	--

Object Name	Object Permissions	Tab Settings
Proposal Related Line Items	Read, Create, Edit, Delete	--
Proposal Requests	Read, Create, Edit, Delete	--
Proposal Rollup Data	Read, Create, Edit, Delete	--
Proposals	Read, Create, Edit	Visible
Proposal Summary	Read, Create, Edit, Delete	--
Proposal Summary Groups	Read, Create, Edit, Delete	--
Proposal Tax Breakups	Read, Create, Edit, Delete	--
Proposal Usage Price Tiers	Read, Create, Edit, Delete	--
Published Favorites	Read, Create, Edit, Delete	--
Query Template Filters	Read	--
Query Template Qualifiers	Read	--
Query Templates	Read	--
Related Account Locations	Read	--
Related Agreements	Read, Create, Edit	--
Related Asset Line Items	Read, Create, Edit, Delete	--
Related Incentives	Read, Create, Edit, Delete	--
Related Line Items	Read, Create, Edit, Delete	--
Related Opportunity Line Item Colls	Read, Create, Edit, Delete	--
Related Opportunity Line Items	Read, Create, Edit, Delete	--

Object Name	Object Permissions	Tab Settings
Related Price List Items	Read, Create, Edit, Delete	--
Related Products	Read, Create, Edit, Delete	--
Retention Policies	Read, Create, Edit, Delete	--
Revenue Recognition Policies--	Read, Create, Edit, Delete	--
Revenue Split/Merge Policies	Read, Create, Edit, Delete	--
Revenue Split Policy Entries	Read, Create, Edit, Delete	--
Rollup Data	Read, Create, Edit, Delete	--
Saved Searches	Read, Create, Edit, Delete	--
Search Attribute Values	Read	--
Search Filters (Comply)	Read	--
Search Filters (CPQ)	Read	--
Service Locations--	Read, Create, Edit, Delete	--
Setting Groups	Read, Create, Edit, Delete	--
Smart Search Activity History	Read, Create, Edit, Delete	--
Summary Groups	Read, Create, Edit, Delete	--
Temp Display Columns	Read, Create, Edit, Delete	--
Temp Filters	Read, Create, Edit, Delete	--
Temp Incentive Benefit Data	Read, Create, Edit, Delete	--
Temp Incentive Limit Data	Read, Create, Edit, Delete	--

Object Name	Object Permissions	Tab Settings
Template Clause References	Read, Create, Edit, Delete	--
Template Clause Reference Versions	Read, Create, Edit, Delete	--
Template Datasource Filters	Read, Create, Edit, Delete	--
Template Dynamic Sections	Read, Create, Edit, Delete	--
Template Dynamic Section Versions	Read, Create, Edit, Delete	--
Templates	Read	--
Template Versions	Read, Create, Edit, Delete	--
Temp Objects	Read, Edit	--
Temp Objects (CPQ)	Read, Create, Edit, Delete	--
Temp Relate Data	Read, Create, Edit, Delete	--
Temp Renew Asset Groups	Read, Create, Edit, Delete	--
Temp Renew Asset Line Items	Read, Create, Edit, Delete	--
Temp Renews	Read, Create, Edit, Delete	--
Temp Rollup Data	Read, Create, Edit, Delete	--
Term Exceptions	Read	--
Totaling Groups	Read, Create, Edit, Delete	--
Transfer Prices	Read, Create, Edit, Delete	--
Usage Price Tiers	Read, Create, Edit, Delete	--
User Views	Read	--

4. Click **Save**.

Apex Class Access

1. Click **Apex Class Access**. The Apex Class Access page is displayed.
2. Click **Edit**.
3. From the **Available Apex Classes** list, select CPQ-related classes.
4. Click **Add** to move them to the **Enabled Apex Classes** list.
5. Click **Save**.

Visualforce Page Access

1. Click **Visualforce Page Access**. The Visualforce Page Access page is displayed.
2. Click **Edit**.
3. From the **Available Visualforce Pages** list, select CPQ-related pages.
4. Click **Add** to move them to the **Enabled Visualforce Pages** list.
5. Click **Save**.

Sharing Settings

1. Go to **Setup** > **Security Controls** > **Sharing Settings**. The Sharing Settings page is displayed.
2. Click **Edit**. The Organization-Wide Sharing Defaults Edit page is displayed.
3. From the **Default Internal Access** drop-down, select a required value.
4. From the **Default External Access** drop-down, select a required value.
5. Select the **Grant Access Using Hierarchies** checkbox for the required object.
6. Click **Save**.

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Product	Public Read/Write	Public Read Only	Checked
Location	Public Read/Write	Public Read/Write	Checked
Address	Public Read Only	Public Read Only	Checked
Adhoc Approval Process	Public Read/Write	Public Read Only	Checked

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Agreement Document Output Format	Public Read/Write	Public Read Only	Checked
Agreement Price Rule	Public Read/Write	Public Read Only	Checked
Agreement Protection	Public Read/Write	Public Read Only	Checked
Agreement Rule	Public Read/Write	Public Read Only	Checked
Agreement Template	Public Read/Write	Public Read Only	Checked
Approval Job	Public Read/Write	Public Read Only	Checked
Approval Matrix	Public Read/Write	Public Read Only	Checked
Approval Process	Public Read/Write	Public Read Only	Checked
Approval Request	Public Read/Write	Public Read Only	Checked
Approval Request History	Public Read/Write	Public Read Only	Checked
Approval Rule	Public Read/Write	Public Read Only	Checked
Approval Rule Dimension	Public Read/Write	Public Read Only	Checked
Asset Line Item	Public Read/Write	Public Read Only	Checked
Asset Line Item History	Public Read/Write	Public Read Only	Checked
Asset Transaction History	Public Read/Write	Public Read Only	Checked
Attribute Value Matrix	Public Read/Write	Public Read Only	Checked
Bundle Component View	Public Read/Write	Public Read/Write	Checked

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Bundle Pricing Mapping	Public Read/Write	Public Read/Write	Checked
Business Category	Public Read Only	Public Read Only	Checked
Business Unit	Public Read/Write	Public Read/Write	Checked
Category	Public Read/Write	Public Read Only	Checked
CategoryTranslation	Public Read/Write	Public Read Only	Checked
Collaboration Request	Public Read/Write	Public Read Only	Checked
Config Settings	Public Read/Write	Public Read Only	Checked
Configuration	Public Read/Write	Public Read/Write	Checked
Constraint Rule	Public Read/Write	Public Read Only	Checked
Custom Data Mapping	Public Read/Write	Public Read/Write	Checked
Customer Success Program	Public Read/Write	Public Read/Write	Checked
Discount Approval Matrix	Public Read/Write	Public Read Only	Checked
DocGen Package	Public Read/Write	Public Read Only	Checked
Favorite Configuration	Public Read/Write	Public Read Only	Checked
Features	Public Read/Write	Public Read Only	Checked
Feature Set	Public Read/Write	Public Read Only	Checked
Feature Setting	Public Read/Write	Public Read Only	Checked

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Field Expression	Public Read/Write	Public Read Only	Checked
Formula Field (Approvals)	Public Read/Write	Public Read Only	Checked
Formula Field (Comply)	Public Read/Write	Public Read Only	Checked
Formula Field (CPQ)	Public Read/Write	Public Read Only	Checked
Group Mapping	Public Read/Write	Public Read/Write	Checked
Guided Question	Public Read/Write	Public Read Only	Checked
Guided Selling Rule	Public Read/Write	Public Read Only	Checked
Legal Entity	Public Read/Write	Public Read Only	Checked
Line Item Rollup	Public Read/Write	Public Read Only	Checked
Payment Term	Public Read/Write	Public Read Only	Checked
Price Dimension	Public Read/Write	Public Read Only	Checked
Price List	Public Read/Write	Public Read Only	Checked
Price Ruleset	Public Read/Write	Public Read Only	Checked
Product Attribute Group	Public Read/Write	Public Read Only	Checked
Product Attribute Matrix View	Public Read/Write	Public Read Only	Checked
Product Attribute Rule	Public Read/Write	Public Read Only	Checked
Product Attribute Rule View	Public Read/Write	Public Read Only	Checked

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Product Configuration	Public Read/Write	Public Read/Write	Checked
Product Constraint	Public Read/Write	Public Read Only	Checked
Product Constraint View	Public Read/Write	Public Read Only	Checked
Product Option Component	Public Read/Write	Public Read Only	Checked
Search Filter (Approvals)	Public Read/Write	Public Read Only	Checked
Search Filter (Comply)	Public Read/Write	Public Read Only	Checked
Search Filter (CPQ)	Public Read/Write	Public Read Only	Checked
Template	Public Read/Write	Public Read Only	Checked
Template Locale	Public Read/Write	Public Read Only	Checked
Temp Object (Comply)	Public Read/Write	Public Read/Write	Checked
Temp Object (CPQ)	Public Read/Write	Public Read/Write	Checked
Translation	Public Read/Write	Public Read Only	Checked

Configuring a Partner Community User

1. Click the **All Tabs** icon ( ). The All Tabs page is displayed.
2. Search and click Accounts. The Accounts page is displayed.
3. Click **New**. The New Account page is displayed.
4. In the **Account Name** filed, enter a name for the account and click **Save**. For example, AutoQuotePartnerAccount.
5. On the **Manage External Account** button, click **Enable As Partner**. A confirmation pop-up is displayed.

6. Click **OK**. The AutoQuotePartnerAccount partner account is created. You must complete the following configuration.

To create a user

1. Go to the Contacts related list on the account (for example, AutoQuotePartnerAccount) and click **New Contact**.
2. From the **Salutation** drop-down, select a salutation for the contact. For example, Mr.
3. In the **First Name** field, enter the first name of the contact. For example, AutoQuote.
4. In the **Last Name** field, enter the last name of the contact. For example, PartnerContact.
5. In the **Email** field, enter the email ID of the contact.
6. Click **Save**.
7. On the **Manage External User** button, click **Enable Partner User**. The New User page is displayed where the First Name, Last Name, and Alias fields are auto-populated.
8. From the **User License** drop-down, select Partner Community.
9. From the **Profile** drop-down, select a profile. For example, Partner Community User Cloned.
10. In the **Email** field, enter the email ID.
11. In the **Username** field, enter partner@conga.com.xxxxxx (here xxxxxx is your sandbox name).
12. Select the **Active** checkbox.
13. Select the **Salesforce CRM Content User** checkbox.
14. Click **Save**. A confirmation pop-up is displayed.
15. Click **OK**.

To assign permission set to the user

1. Go to the Permission Set Assignments related list on the user (for example, AutoQuote PartnerContact) and click **Edit Assignments**.
2. From the **Available Permission Sets** list, select the required permission set. For example, CPQ Sales User – Partner.
3. Click **Add** to move it to the **Enabled Permission Sets** list.
4. Click **Save**.

Configuring a Partner Community Site

1. Go to **Setup** > **Customize** > **Digital Experiences** > **All Sites**. The Digital Experiences page is displayed.
2. Click **New**. The Choose the Experience You Love page is displayed.

3. On the **All** tab, select a required theme. The Customer Service page is displayed.
4. Click **Get Started**. The Enter a Name page is displayed.
5. In the **Name** field, enter a name for your community.
6. In the **URL** field, enter a site prefix.
7. Click **Create**. It may take time to load the new community page.
8. Click **Administration**. The Administration page is displayed.
9. Click **Settings**. The Setting section is displayed.
10. Click **Activate**. A confirmation pop-up is displayed.
11. Click **OK**. You must complete the following configuration for the site.

Members

1. On the Administration page, click **Members** on the left panel. The Members section is displayed.
2. Under Select Profiles, in the **Search** drop-down, select All.
3. From the **Available Profiles** list, select the required profile. For example, Partner Community User Cloned.
4. Click **Add** to move it to the **Selected Profiles** list.
5. Click **Save**.

Menu Items

1. Click **Administration** on the top-left and click **Builder**.
2. Click the  icon on the toolbar that is displaying the HOME tab. The Navigation Menu is displayed on the right side.
3. Click **Edit Default Navigation**. The Edit Default Navigation pop-up is displayed.
4. Click **Add Menu Item**. The Menu Items section is displayed on the pop-up.
5. Enter the following details:
    a. In the **Name** field, enter Quotes.
    b. From the **Type** drop-down, select Salesforce Object.
    c. From the **Object Type** drop-down, select Quote/Proposal.
    d. From the **Default List View** drop-down, select Default.
    e. Select the **Publicly available** checkbox.
6. Click **Add Menu Item**. The Menu Items section is displayed on the pop-up.
7. Enter the following details:
    a. In the **Name** field, enter Agreements.
    b. From the **Type** drop-down, select Salesforce Object.
    c. From the **Object Type** drop-down, select Agreement.
    d. From the **Default List View** drop-down, select Default.

    e. Select the **Publicly available** checkbox.
8. Click **Save Menu**. The menus you added are displayed on the toolbar that is displaying the HOME tab.
9. Click **Publish** from top right. The Publish your site? pop-up is displayed.
10. Click **Publish** and then click **Got It**.

Configuring a Custom Button on the Quote Object for Partners

1. Go to **Setup > Create > Objects** and search for **Quote/Proposal** object.
2. In the **Custom Fields & Relationships** related list, click **New**.
3. From **Step 1. Choose the field type**, choose **Formula** as Data Type, and click **Next**.
4. From **Step 2. Choose the output type**, enter the following details and click Next:
    a. In the **Field Label** field, enter Configure Products Partners.
    b. In the **Field Name** field, enter Configure_Products_Partners.
    c. Choose **Text** as **Formula Return Type**.
5. In **Step 3. Enter formula**, paste the following URL under the **Simple Formula** tab and click **Next**:

```
IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/
Apttus_Config2__Cart businessObjectId="&Id&"&flow=PurchaseFlow", IMAGE("/
resource/Apttus_QPConfig__Button_Configure", "Configure Products Partners"),"_s
elf"), NULL)
```

6. In **Step 4. Establish field-level security**, select Field Level Security for the required profile and click **Next**. For example, Partner Community User Cloned.
7. In **Step 5. Add to page layouts**, choose the page layouts on which you want to display this field.
8. Click **Save**.
9. Provide access of this field to the required profile. For example, Partner Community User Cloned.

Validating the Partner Community User

1. Go to the contact that you created (for example, AutoQuote PartnerContact).
2. Click **Login to Experience as User**.
    - If you have multiple communities, CPQ prompts you to select a community.
    - If you have only one community, CPQ navigates you to the community home page.

3. On the Quotes tab, click **New**.
4. Select a record type as Proposal and click **Next**.
5. Verify if all required fields are visible. If not, you must verify Field-Level Security configuration.
6. Select the quote you created and assign a price list.
7. Click **Configure Products Partners**. The cart configuration page is displayed.
8. Select the required products and finalize the cart.

> If CPQ did not launch the cart, verify sharing settings or field-level access again. You must enable subscriber logs to get more detail.

Managing Products

Products are the primary component of your CPQ system. Using CPQ, you can create individual products and product bundles. Once you've created products, you can associate attributes and options to them. Options are the additional accessory products to sell together with a product. Attributes allow you to associate product specifications such as color, size, or model.

You must associate the products with Categories to classify products into logical groups. You can define hierarchies in a category to further divide the products into the smaller groups. On the Catalog page, CPQ displays the categories and hierarchy containing the products you associate with them.

Using feature sets, you can also configure CPQ to allow your users to compare the features of multiple products. By implementing feature sets, you allow your users to compare products and choose the product that best meets their requirements.

- About Products
- Managing Categories and Hierarchies
- Managing Bundles
- Managing Product Groups
- Creating and Maintaining Attributes
- Configuring Product Visibility
- Configuring Product Comparison
- Configuring Numeric Expressions

About Products

Products are standard products or services you sell to your customers and are defined in the standard Salesforce products table. A product is classified as one of the following types: Standalone, Bundle, or Options. This section shows you how to create products and then group the products to make them easier for customers to find.

A standalone product can be sold on its own under the offering category; whereas options can be grouped together as an option group and then associated with a bundle product to be categorized as an offering. You can also associate a bundle with another bundle.

The following sections describe various product operations about creating products.

- Navigating the Product Page
- Creating Products
- Cloning Products
- Configuring Product Families

Navigating the Product Page

When you launch the CPQ Admin console, Manage Product page is displayed. All the existing products are listed on this page along with their details. You can configure the columns displayed on the Manage Product Page. There are various functionalities available for you to manage the products.

Product Search

Search for the product to narrow down the product list using the Search Products box. Enter a keyword related to your product details like Product Name, Product Code, Configuration type, description. Search Product box returns all the products that contain that keyword in the product details.
There are two types of search available on the products homepage. Saved Search and Product search query.

> ⓘ Either of the search filters is applicable at a time.

Saved Search provides you with various options to search, sort, and filter products. You can either define search criteria to search for a range of products or use the product search option. The search query uses options provided in the Field, Operator, and value and displays a search result that matches the criteria. Saved Search functionality lets you save

a set of filters you define that narrows down the list of products according to your requirement.
Perform the following steps to view and search configured products in the Saved Search:

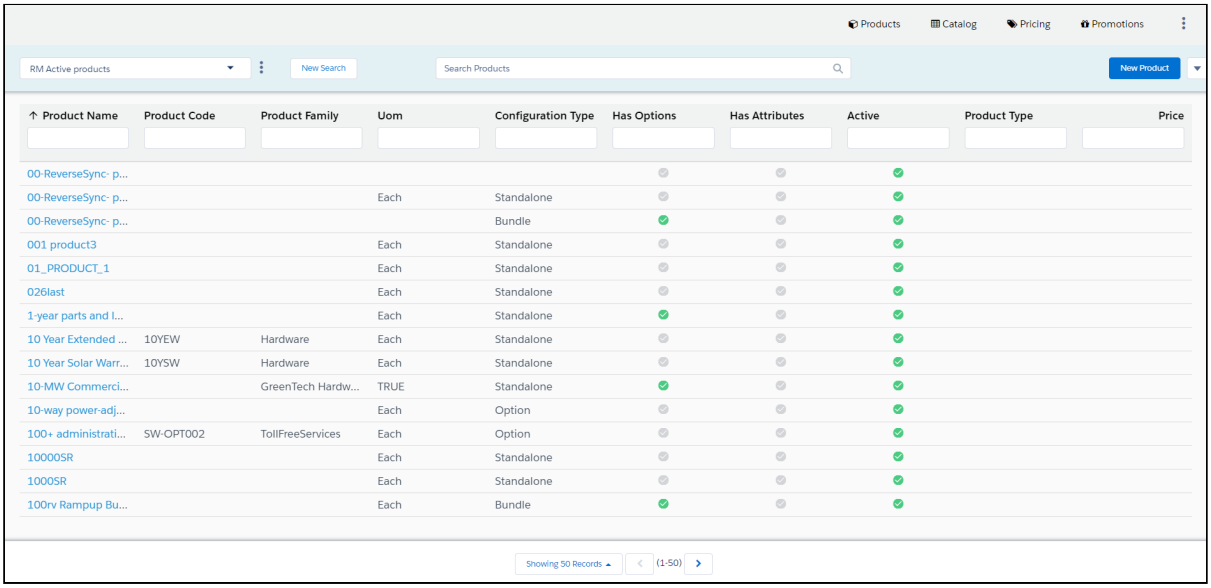
1. Click **View All** to view all products that are configured in CPQ on the dashboard. If you have queried a filter or a product, then a select **View All** to disable the existing search filter.
2. Click **New Search**. A window that consists of various fields, operators, and values are displayed.
    a. **Field**: All product Fields are available in the drop-down list such as Text, Checkbox, Date, Date/Time, Number, Percent, Picklist, Multi-Select Picklist, Currency.
    b. **Operator**: Operators that are available for selection helps create the filter expression. Operators are displayed based on the field types. For example, a picklist value will display equal to and not equal operators and a text field will display equal to, not equal to, starts with, contain, does not contain operators. If it is a multi-value picklist, the application displays include and does not include operators.
    c. **Value**: Enter a value.
    d. Click one of the following options to execute the search:
        - Click **Save As** to save the search filter and use the filter criteria.
        - Click **Apply** to enable the search filter.
        - Click **Cancel** to cancel the search criteria.
3. To search a product in the Product Search, enter the name of the product that you want to search in the search field. Products that match the search are displayed.
4. Click field headers in the product's dashboard to sort dashboard based on the field.

You can simply click the saved search record you created and the system filters the search results. You can see the saved searches dropdown list in alphabetical order for better accessibility of your list of saved searches. If you have configured a customized view, click the menu icon (⋮) next to the New Search button. The following options are displayed.

- **Edit View**: Displays a new search window.
- **Save View As**: Displays options to Save the edit view.
- **Remove View**: Removes the filter.

You can also narrow down the list using the column search (inline search). Enter a keyword related to the column in space under the column name. Unlike the Search Product field, column search matches the keyword in that particular column. Use the keywords True and False for the checkbox type columns. You can enter keywords in multiple columns at the

same time, the result displayed contains the products with details matching to all keywords you entered in different columns.



Refresh the page to clear all the filters.

Sorting and Adjusting the Columns

You can sort the product by clicking the column names. The products are sorted alphabetically, every time you click the column name the sorting switched between ascending and descending order. You can adjust the column width by dragging the columns. Hover near the column boundaries till two-way arrow appear. Click when the arrow appears and drag left or right to adjust the column width. At the bottom of the page, you can control the number of products to be displayed on each page. You can select the number of products from a pre-defined list. You cannot configure this list. By default, you see 50 records on each page.

Product Details

Product details hold the information about the product. You can view them on the Manage Product page or the product details page. When you click the product name displayed in blue color, the **DETAILS** tab is displayed. All the primary details of the products are displayed on this page. Other tabs on the secondary toolbar hold additional information like options information, price list associated with the product, rules associated with the products. You can edit the details on this page.

You can use the Mini Display functionality to view the details without leaving the Manage Product page. Click the information in any column other than the product name for the respective product. A panel on the right-hand side displays the details of the product. You

cannot edit any details on the mini display. Click the edit icon to go to the **DETAILS** tab, to edit the details.

You can see a list of all the existing products displayed in a panel on the left. When you hover the cursor over the product name, product details are displayed in a pop-up. You can configure the pop-up to display other fields. For more information, refer to Configuring the Tooltips.

Creating Products

A Product in CPQ terminology is a product or service that can be set up to be sold on its own as a standalone product or options of other products. Products or services are set up as standalone, bundle, or options of other products. You can associate a product to a category or a price list. If you created an option product, you must first associate it with an option group.

- Standalone - This is an individual product that can be sold on its own.
- Bundle - Any product that has other products (options or bundles) associated with it.
- Options - This is a product that can be sold with a bundle product only through an option group.

To create a product

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**. Click **New Product**. The product details page is displayed. **Details** tab opens by default.
3. Enter the following product information, as required.

Option	Description
Product Name	Enter a name for the product.
Product Code	The Product Code can be an internal code or a product number to identify the product within your organization.
Product Family	Enter a Product Family to categorize the products. If the required Product Family is not displayed in the list, you must add a picklist value in the Product Family field on the Product object. For more information refer to Configuring Product Families.

Option	Description
Uom	Select a unit of measurement.
Product Description	Enter a description of the product. This description is displayed when the user clicks the product name on the catalog page.
Disable Bundle Expansion	Select this checkbox to disable the expansion of the bundle or a sub-bundle on the Cart page. This does not allow the Sales Rep to expand the bundle to view the options and secondary line items. If you only disable the expansion of a sub-bundle, the Sales Rep can still expand the parent bundle. You must create this field in the Product object. The checkbox is only considered if **Configuration Type** is *Bundle* and **Has Options** is selected. Otherwise, CPQ does not disable expansion. You must create this field on the Product object. Refer to Configuring Bundle Expansion on Cart Page.
Configuration Type	Select one of the following types of product: • **Standalone** • **Bundle** • **Option** This field is mandatory.
Must Configure	Indicates that the user has to configure a product to add options or attributes or both. The Configure button is the only option available on the Catalog page. When you select Must Configure, you must also select either **Has Options**, **Has Attributes**, or both. Clearing the checkbox allows an end user both the options; either add the product to cart directly or configure the product with attributes or options, or both.
Has Options	Indicates a product as a bundle that has options (bundles/options). You can associated options and option groups to the bundle on the Structure tab. Options and Options groups are not displayed on the Configuration page, is this disabled.
Has Attributes	Indicates that the product has attributes associated to it. During the quote creation process, the end user can enter values for those attributes. Using the product attributes administration tool, you can associate product attributes to products.

Option	Description
Active	Indicates the product is active. The Product is not displayed on the Catalog page if this setting is not selected.
Product Type	Select a type of product. For example, Equipment, Service, and so on.
Has Search Attributes	Indicates that the product has search attributes and is useful for searching related products based on product attribute values.
Renewal Lead Time	Enter a value when the renewal quote must be created for this product after an order is activated and asset is generated. By default, the value is 0, which indicates that the renewal quote is created immediately after the order is activated, with the same number of Line Items in the order. If you specify 30, the renewal quote is created 30 days before the Asset End Date.

The fields on the details page are customizable. For more information refer to the Navigating the CPQ Admin User Interface topic.

4. Click **Save**.

A product is created and saved.

Cloning Products

You can also create new products by cloning existing products. You can clone the product with partial or entire configuration details. This helps you to save time if you want to create a large number of products with a similar configuration.

To clone a Product

Follow the steps below.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. From the menu icon (⋮) next to the **New Products** button, click **New from Clone**. A pop-up is displayed.
4. Fill in the required details.

Option	Description
New Product Name	Enter a name for the new product.
Product To Clone	Type the name of the product you want to clone. Or open the details of the product you want to clone and then click **New From Clone**. In this case, the field is populated automatically.
External ID	Select an external ID.
Entire Product	Select this to clone the entire configuration of the product. When you select this all the checkboxes are automatically selected.
Product Details	This indicates that the products details of the products are cloned. This option is selected by default.
Categories	Select this to include the categories associated with the product.
Attributes	Select this to include the attribute groups and attributes associated with the product.
Pricing	Select this to include all the price list and price list item associated with the product.
Options	Select this to include the option groups and options associated with the product.
Product Collateral	Select this to include the media associated with the product.
Product Group	Select this to include the product group associated with the product. If you select this, the new product is added in the product group.
Rules	Select this to include the rules associated with the product. You can select which rule to include.

5. Click **Save.**

A new product is created.

Configuring Product Families

Product Family is a standard Salesforce.com feature. You can use the Product Family picklist to categorize your products. For example, if your company sells both hardware and software, you can create two Product Families: Hardware and Software.

To begin using Product Families, you must:

- Customize a Product Family picklist
- Edit a Product and select an appropriate Product Family value

Customizing the Product Family picklist

You can customize the Product Family picklist to include the different categories of products you sell.

To customize the Product Family picklist

1. Go to **Setup** > **App Setup** > **Customize** > **Products** > **Fields**.
2. From Product Standard Fields, click **Product Family**.
3. From Product Family Picklist Values, click **New**. Add one or more picklist values with each value on its own line.
4. Click **Save**.

The Product Family picklist displays the values you entered.

For each product in your price list, edit the product and select the appropriate Product Family value.

Editing a Product to include Product Family

For each product in your price list, you need to edit the product and select the appropriate Product Family value.

You must have an existing product.

To edit a product to include Product Family

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the Products menu, click **Manage Products**.
3. Search and click the product to open the product details page.

4. On the **Details** tab, from Product Family, select an appropriate value.
5. Click **Save**.

Based on your selection, your product becomes part of a Product Family.

Managing Categories and Hierarchies

Categories are high-level logical groupings of products, affecting the way the Sales rep sees them for selection on the Product Catalog. A category can be created for browsing products in the Catalog page or they can be created for creating options groups for a Bundle product. Products are associated to a category through a category hierarchy. Category hierarchies are maintained using the Hierarchy Manager. They are also used to group the total price of the products belonging in a category on the Cart Summary.

There are three types of category:

- Offering: An Offering is a collection of products that an organization makes available to a customer. A Bundle or a Standalone product can be associated with an Offering category. If you are creating a category hierarchy for the first time, you need to create a new category through this method only. Most implementations will have an existing hierarchy.
- Offering and Option Group: This category type groups products that can be used as a category and option group. For example, a keyboard can be sold as a standalone product and also as an option along with a bundle product (such as Desktop PC).
- Option Group: Option Groups categorize multiple option products. Option products are associated to option groups, and option groups are further associated to bundle products, thus associating option to a bundle product. You can manage option groups in **Manage Option Groups** menu.

The following topics provide instruction to manage Categories and Category Hierarchy.

- Creating Categories
  Categories are created to group products for the end user's product selection. Categories are also used to group options within bundles.
- Creating Category Hierarchies
  Categories are created when a hierarchy needs to be created for the first time. Hierarchy defines the structure of the product catalog.
- Running a Category Maintenance Job
  After making any change to category definition, category to product association, or hierarchy change, you must run a Category Maintenance job. This de-normalizes the hierarchy into a custom object for reporting and totaling purposes. If category maintenance is not run, the end user may see incorrect hierarchy or totals on the cart

page. Making changes to existing categories or category hierarchies requires you to run the Category Maintenance batch job to maintain the hierarchy and associations.

Creating Categories

The purpose of categories is to group products for the end user's product selection. Categories are also used to group options within bundles. Follow the steps below to create and modify a category.

To create a Category

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Catalog**.
3. Click **New Category**. **Create New Category** pop-up is displayed.
4. Enter a mandatory **Name** and **Label**.
5. Select **Active** to activate the category. By default, the **Active** check box is already selected.
6. Select one of the following from the **Type** picklist.
    - *Offering*
    - *Offering and Option Group*

   Only these two types are allowed for category creation. Option Group type is only available for option group creation on the **Manage Option Group** menu.
7. Click **Save**.

Based on your category type selection; an *Offering*, or an *Offering and Option Group* type category is created and added.

You can create category hierarchies based on your business requirement. Select a Parent Category when you create a new category.

To modify an existing Category

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Catalog**.
3. Search and click the category to open the category details page.
4. Modify desired fields as required.

Field Name	Description
Name	Enter a name for the category.
Label	The label name is populated automatically.
Description	Enter a description of the category.
Default Search Category	When selected, **Default Search Category** displays the product by default under the category or sub-category in the catalog page. If not, the user has to first select the category from the category panel or the browse section to see the product.
Expanded By Default	Indicates that the hierarchy is expanded at all times in the catalog page
Include In Totals View	Indicates that the mode is displayed in the totals view on the shopping cart
Hide All Search Filters	Indicates that all the search filters in **Narrow Your Search** and **Refine Your Search** on the Catalog page are hidden.
Search Filter Fields	Selected filter fields are displayed in the **Refine Your Search** section. Select fields from the dropdown menu, click anywhere on the screen once selected all the desired fields.
Long Description	The Long Description is displayed for categories that have bundle products on the Options page.
Catalog's Image(s)	You can add an image like a symbol or an icon for the sub-category. Click **Browse** to select an image from your local machine.

5. Click **Save**.

The category is modified and saved. After creating or modifying the categories, you must run the Category Maintenance batch job.

Creating Category Hierarchies

Category hierarchy defines the structure of the product catalog. You can build and maintain a category hierarchy on the Category page. You can add sub-categories (also

known as nodes) to a category, add an image, reorder categories, and associate products, thus building a hierarchy. Most implementations will have an existing hierarchy.

You can create multi-level categories. For example: **Home > Laptops > Model**.

Prerequisite

You must have an existing Offering category.

To build a Hierarchy

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Catalog**.
3. Search and click the category to open the category details page.
4. The Category details page is displayed, which allows you to perform the following actions for the selected category.
    - Add or Edit a Sub-category
    - Remove a category
    - Associate Products
    - Reorder Categories
5. Click **Save**

A category hierarchy is built. The hierarchy is visible on the left-hand side of the Category Manager page. After building the hierarchy, you must run the Category Maintenance batch job.

In the following section:

- Adding Sub-Categories
- Removing Sub-Categories

Adding Sub-Categories

Sub categories (also known as nodes) categorize products within a category. You can add multiple sub-categories/nodes to categorize products within a category.

Before You Begin

You must have an existing top-level category.

To create a new sub-category within a Category

> **Note**
>
> You cannot add a sub-category to a category that has products associated to it.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Catalog**.
3. Search and click the category you desire to be the parent category. The category details page is displayed.
4. Click **ASSOCIATED CATEGORY**. Click **New Category**. Or click **New Category** on the category details page. **Create New Category** pop-up is displayed.
5. On the **Create New Category**, check that the desired parent category is selected. Search or use arrow icons at the bottom to find the desired category.
6. Fill in the information as explained in the table below.

Field Name	Description
Name	Enter a name for the category.
Label	The label name is populated automatically.
Default Search Category	When selected, **Default Search Category** displays the product by default under the category or sub-category in the catalog page. If not, the user has to first select the category from the category panel or the browse section to see the product.
Expanded By Default	Indicates that the hierarchy is expanded at all times in the catalog page
Hide All Search Filters	Indicates that all the search filters on the catalog page in Narrow Your Search and Refine Your Search sections are hidden.
Include In Totals View	Indicates that the mode is displayed in the totals view on the shopping cart

Field Name	Description
Description	Enter a description of the category.
Long Description	The Long Description is displayed for categories that have bundle products on the Options page.
Upload Image	You can add a small image like a symbol or icon for the sub-category. Click **Browse** to select an image from your local machine.

7. Click **Save.**

The sub-category is created and added as a node to the hierarchy.

Removing Sub-Categories

You can remove a sub-category and its nested categories from the Associated Category page.

To remove a sub-category

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Catalog**.
3. Search and click the category you desire to be the parent category. The category details page is displayed.
4. Click **ASSOCIATED CATEGORY**. All the associated sub-categories are listed on this page.
5. Click the remove icon(×) under the **Action** column for the sub-category you want to remove.
6. Click **Save**.

The sub-category is removed from the hierarchy.

Associating Products to a Category

Following the steps below you can associate products to a category at a category-level. You can also associate a product to a category from the product-level.

> ⓘ You cannot associate products to a category with sub-categories associated to it.

To associate a Product to a Category

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Catalog**.
3. Search and click the category to open the category details page. **DETAILS** tab opens by default.
4. Click **ASSOCIATED PRODUCTS**. If the category has sub-categories, **ASSOCIATED PRODUCTS** tab is not visible. In this Case, click **ASSOCIATED CATEGORY** and then select a sub-category. If you nested sub-categories, navigate to the last sub-category.
5. Drag and drop all the products you want to add in the category from the **All Product** list. You can search for the products using the search bar.
6. Click **Save**.

Your desired product is associated with a category.

Associating Categories to a Product

A product (standalone or bundle) must be associated to a category to be displayed on the catalog page.

This can be achieved at a category level or a product level. At the product level you can associate categories to a product and at the category level, you can associate products to a category. On the Products page and Category page, you can update and modify the categories and products respectively.

To associate Category to a Product

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click the product to open the product details page.
4. Click **ASSOCIATIONS**. Click the category hierarchy icon (). **Category Hierarchy** pane is displayed.
5. Drag and drop the desired Categories from **Category Hierarchy** pane. You can search for the categories using the search bar.
6. Click **Save**.

The product is now associated with a category.

> For performance implications, it is recommended that you limit the number of products in a single category to 1000.

Sequencing Products within a Category

You can use the orderByClause parameter to sequence the Products on the Catalog page. You can add a new field in the Product2 object (for example: as Category Rank) and set the orderByClause in this manner: productOrderByClause=Category_Rank__C NULLS LAST, Name while configuring a formula to launch the Product Catalog.

By default, the product list shown on the Catalog page is sorted by **Product Name**.

To sequence the Products on the Catalog page

1. Create a new custom field on Product2 object. For example: Category Rank.
2. From the Product Detail screen, assign the appropriate numeric value for this field.
3. Use the following formula to launch the ngCart.

```
IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 ,HYPERLINK("/apex/
Apttus_QPConfig__ProposalConfiguration?id=" &Id&"&flow=ngFlow&productOrderByClause=Ca
tegory_Rank__c NULLS LAST, Name" ,IMAGE("/resource/Apttus_QPConfig__Button_Configure",
 "Configure Products"),"_self"), NULL)
```

On the Catalog page, Products are displayed based on the sort order defined in the custom field.

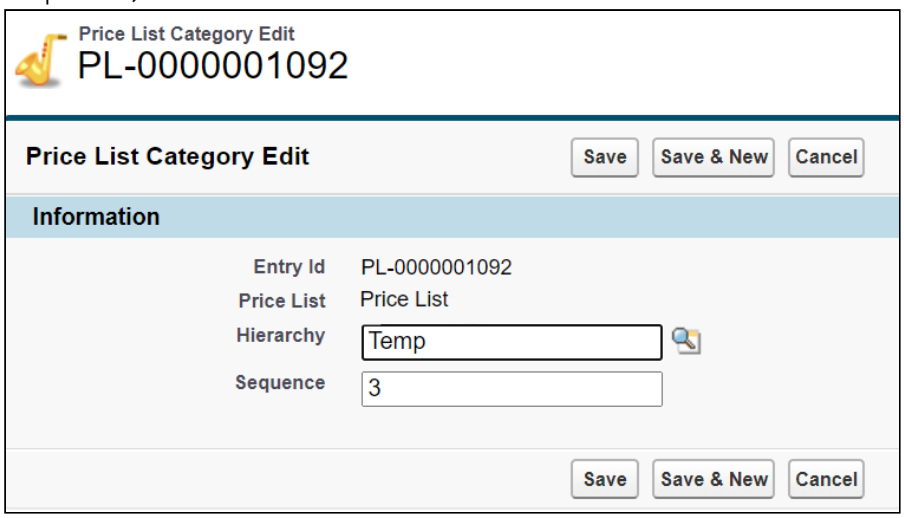
Reordering Root Categories on the Catalog Page

Product managers can correctly sequence the root or top-level catalog categories to ensure that the right product lines are prioritized or that the most common categories are positioned first. The ordering can be different for different Price Lists. Price Lists drive the hierarchy that shows up in the catalog for a given quote.

To reorder or sequence the root categories (top-level categories on the catalog page)

The categories are associated with the category hierarchy and the category hierarchy is associated with a Price List for it to show up in the catalog.

1. In Salesforce Classic, go to the **Price Lists** tab.
2. Select a price list in which you want to reorder categories.
3. Under the related list for Categories, click **Edit** for the category that is related to your target root (top-level) category.
4. Update the **Sequence** fields with the desired number. Set the sequence number relative to other categories to drive the order (Object: Price List Category, Field: Sequence).



5. Click **Save**.

> You must edit the layout to include the field **Sequence** on the related list for Category and on the Price List Category Details page.

The order set by the sequence number for a given Price List should be the order in which the categories show up on the catalog page. You do not need to run the maintenance job after updating the sequence.

You also can rearrange the order of your categories or sub-categories within a hierarchy. Reordering of categories can be done from the leaf as well as root nodes. Leaf nodes are the level of categories between the top level category and the lowest node.

To re-order a sub-category

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Catalog**.
3. Search and click the category you desire to be the parent category. The category details page is displayed.
4. Click **ASSOCIATED CATEGORY**. All the associated sub-categories are listed on this page.
5. Drag and drop the sub-categories list in the section in the middle in the desired order.
6. Click **Save**.

The system updates the order and displays the hierarchy.

Running a Category Maintenance Job

When you create a category and associate products to the category or change the category definition, you must run a Category Maintenance job. If you do not run category maintenance, the Sales rep may see incorrect hierarchy or totals on the cart page.

After making changes to the existing category definition you can run the Category Maintenance batch job only for changes rather than running for the entire definition. For example, if your category has 10 products and you added 5 more, you can run the Category Maintenance batch job only for the 5 additional products.

For more information, see Running Maintenance Jobs.

> ⓘ You must run the Category Maintenance batch job after every CPQ version upgrade.

Managing Bundles

Bundled products represent a combination of standalone products that offer added values. An option group is used to combine product options and then associate that group to a bundled product.

In this section:

- Creating Option Groups
- Associating Options to an Option Group
- Associating Options to a Bundle

- Configuring Multi-Level Bundles
- Cloning a Bundle within the same Quote
- Configuring Bundle Expansion on Cart Page
- Running a Bundle Maintenance Batch Job

Creating Option Groups

Option groups are a type of category that is used to group option products together to be associated with bundles. Options group can also be used as categories.

There are two types of option groups:

- *Option Group:* Option Groups categorize multiple option products. Option products are associated to option groups, and option groups are further associated to bundle products, thus associating options to a bundle product.
- *Offering and Option Group:* This category type groups products that are of type Standalone and Option. For example: A keyboard can be sold as a standalone product and also as an option along with a bundle product (such as Desktop PC).

Option groups are further categorized into two types Shared Option Group and Standalone Option Group. You can create these option groups on the **STRUCTURE** tab on the products details page in CPQ Admin. All the option Group you create in the **Manage Options Group** menu are by default shared.

Shared Option Groups can be associated with multiple bundles. Whereas, Standalone Option Groups are bundle specific. Though, you can associate standalone options groups with other bundles. However, only the option group structure that you saved for the first time while creating the standalone option groups is associated with another bundle. The changes you make to the option group structure after saving the first time are not reflected whenever you associated the same option group with other bundles.

When you click **New Shared Option Groups** or **New Standalone Option Groups** a pop-up opens in which you create a new option group. Follow the steps in the *To create an Option Group* below to after you click **New Shared Option Groups** or **New Standalone Option Groups**. When you create an option group in the structure page of a bundle, that option group is automatically associated with the bundle. Shared option groups are also associated with the bundles. Any change structural changes you make are reflected on all the bundles the shared option group is associated with. For example, if you change the options in the option group, the changes are reflected in all the bundles the shared option group is associated with.

To create an Option Group

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Option Groups**.
3. Click **New Option Group**. **Create New Option Group** pop-up is displayed.
4. Enter a mandatory **Name** and **Label**.
5. Select **Active** to activate the category. By default, the **Active** check box is already selected.
6. Select one of the following from the **Type** picklist.
    - *Option Group*
    - *Offering and Option Group*
7. Click **Save**.

Based on your **Type** selection; an *Option Group* or *Offering* and Option *Group* type options group is created and added.

You can create option group hierarchies based on your business requirement. Select a Parent Option Group when you create a new option group.

To modify an existing Option Group

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Option Groups**.
3. Search and click the option group to open the option group details page.
4. Modify desired fields as required.

Field Name	Description
Name	Enter a name for the options group.
Label	The label name is populated automatically.
Description	Enter a description of the options group.
Default Search Category	When selected, **Default Search Category** displays the product by default under the category or sub-category in the catalog page. If not, the user has to first select the category from the category panel or the browse section to see the product.

Field Name	Description
Expanded By Default	Indicates that the hierarchy is expanded at all times on the catalog page
Include In Totals View	Indicates that the mode is displayed in the totals view on the shopping cart
Hide All Search Filters	Indicates that all the search filters on the catalog page in Narrow Your Search and Refine Your Search sections are hidden.
Search Filter Fields	Selected filter fields are displayed in the **Refine Your Search** section. Select fields from the dropdown menu, click anywhere on the screen once selected all the desired fields.
Long Description	The Long Description is displayed for categories that have bundle products on the Options page.
Catalog's Image(s)	You can add an image like a symbol or an icon for the sub-category. Click **Browse** to select an image from your local machine.

5. Click **Save**.

The option group is modified and saved. After creating or modifying the options, you must run the Category Maintenance batch job.

> In the Salesforce Classic user interface select the **Cascade Group Changes** field on the Product Option Group details page to enable a warning message when you modify any shared option group.

Associating Options to an Option Group

You need to associate options with options groups to use them in a bundle.

Prerequisite

You must have an existing Option product and an Option Group category.

To Associate Options to Option Groups

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Option Groups**.
3. Search and click the options group to open the option group details page.
4. Click **ASSOCIATED PRODUCTS**.
5. Drag and drop all the options you want to add in the options group from the **All Product** list. You can search for the options using the search bar.
6. Click **Save**.

Option products are associated with an option group. Next, you must associate option group to a bundle product through an option group.

To Associate Options to Option Groups within a Bundle

1. On the **Manage Products** page, open the products details page of the bundle.
2. Click **Structure**.
3. Click the shortcut icon (‹) under **New Shared Option Group** button. **Option Hierarchy** panel is displayed.
4. On the **Option Hierarchy** panel select a category from the dropdown menu.
5. Drag and drop desired option group and options from the **Option Hierarchy** pane.
6. Click **Save**.

> When you modify an option group that is associated with multiple bundles, all the associated bundles are affected.
> Select the **Cascade Group Changes** field on the Product Option Group details page to enable a warning message when you modify any shared option group.

Associating Options to a Bundle

An Option is associated with a bundle product only through an Option Group to give the end user an additional choice for a product. To create an Option product, create a new product and from configuration type select Option. You must associate an Option Group that contains Options to a Bundle. After associating Options to an Option Group, you must associate the option group to a Bundle.

Prerequisite

Has Options in the **Details** tab must be selected. Otherwise, the **Structure** tab is not displayed. For more information refer to Creating Products.

To associate option groups to a bundle

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click the bundle you want to associate options with.
4. Click **Structure**. The page displays all the options and option group associated with the bundle.
5. Click the shortcut icon ( ) under **New Shared Option Group** button. **Option Hierarchy** panel is displayed.
6. On the **Option Hierarchy** panel, select Options Groups from the dropdown menu.
7. Search the option group using the search bar and then drag and drop the options group in the middle.
8. Click **Save**.

Options are now associated to the bundled product through an option group. You can associate the bundle product to a category or a price list.

You can choose to lock the option quantity value, which is populated using numeric expressions.

To modify an option group within a bundle

1. On the **Manage Products** page, open the products details page of the bundle the option group is associated to.
2. Click **Structure** and select the option group you want to modify.
3. Fill in the required details as explained in the table below:

> Option Groups can be of *Shared* or *Standalone* type. When you make any structural changes to the *Shared* option group such as removing an option, such changes are reflected in all the bundles where you have associated this option group. The changes you make to the *Standalone* type option groups are limited within the bundle.

Field Name	Description
Content Type	Options - Default value for Content Type is Options. Attributes - When content type is selected as Attributes, attribute group lookup filed is displayed. Detail page - When content type is selected as Detail page, the system will display the text field in which you can specify the detail custom page URL. Note: When you migrate data from one org to another using Data loader, ensure that you select Options value for the Content Type.
Min Options	Specify the minimum product options which a user should select on the options page.
Max Options	Specify the maximum product options which a user should select on the options page.
Min Total Quantity	Specify the minimum required value, inclusive of all the options inside an option group. For example, if you enter 10 in this field, the minimum quantity of all the options should reach 10. Thus, you can have 5 options, each with quantity=2. You can also use Numeric Expressions to populate this field. Ensure that the return value of your numeric expression is of type *Number*.
Max Total Quantity	Specify the maximum required value, inclusive of all the options inside an option group. For example, if you enter 10 in this field, the maximum quantity of all the options should reach 10. Thus, you can have 5 options, each with quantity=2. You can also use Numeric Expressions to populate this field. Ensure that the return value of your numeric expression is of type *Number*.
Is Hidden	Select this check box if you want to hide this option group on the catalog page.

Field Name	Description
Is Picklist	Select this check box when you want to display your options in the form of a picklist, from which your user can select only one option. When you select this check box, ensure that Min Options and Max Options are set to 1.
Modifiable Type	Variable – Select this option if you want to allow the user to change the quantity of the option products during the configuration. Fixed – Select this option if you want to restrict the user from changing quantity of the option products inside this option group.

Note that you should cautiously use the Min/Max option field in conjunction with hiding an option group. If no option products are visible due to Visibility rules configuration or different price lists, and you have set the minimum options as zero, the system considers this configuration as invalid.

To modify existing options inside a bundle

1. On the **Manage Products** page, open the products details page of the bundle the option is associated to.
2. Click **Structure** and select the option group which is associated with the options you want to modify. Options are displayed in the middle section.
3. Fill in the required details for an option as explained in the table below:

Field Name	Description
Min Quantity	Specify the minimum quantity for the specific option. You can also use Numeric Expressions to populate this field.
Max Quantity	Specify the maximum quantity for the specific option. You can also use Numeric Expressions to populate this field.
Default Quantity	Enter a specific number of options to the bundle during configuration or you can also use Numeric Expressions to populate this field. If you define a numeric expression, a lock icon is displayed on the Configuration page next to the option quantity.

Field Name	Description
Default	Select the check box to mark this option as Default.
Quantity Modifiable	Select this check box to make the option quantity modifiable by the user on the Options and Cart page.
Allow Cloning	Selecting this checkbox for specific product option, bundle or sub-bundle will show the clone icon next to the option on the configuration page while configuring the product. When you click this icon, the system will clone all associated attributes as is. ⓘ Cloning is only allowed for option, bundle or sub-bundle with **Max Quantity** more than 1.
Required	Select the check box to mark this option as mandatory.
Auto Update Quantity	Select this check box to enable an option quantity to automatically update, when you have the option quantity being derived from a numeric expression. Selecting this checkbox, CPQ recalculates the new quantity of an option when something else has changed that would re-evaluate the expression itself. When you disable the checkbox, the quantity is updated only has the first time of option selection.
Config Type	Choose if you want to display the options as inline or on a new Visualforce page. This is to ease the sales rep to configure the options on the same page or on a different configuration page. By default, *Inline* option is chosen for the new products, for both, options and attributes. *Inline:* The attributes and options can be configured on the configuration page. *New Page:* A wrench icon is displayed near the bundle or sub-bundle product which has to be configured. When you click this icon, a new page is opened for configuration of options and attributes.

Field Name	Description
Inclusion Criteria	Specify the minimum product options that a user should select on the catalog page.

4. You can define the fields mentioned in the table above for options individually or define for all the options at once using the edit icon ( ) next to **Action**.
5. Click **Save**.

Whenever you modify a bundle structure by adding or removing options or option groups, a message to execute the bundle maintenance batch job is displayed. If you are using Salesforce Classic, you must execute the bundle maintenance job. The bundle maintenance batch job is automatically executed in CPQ Admin. However, you must execute bundle maintenance batch job after mass changes. For more information, refer to Running a Bundle Maintenance Batch Job.

To lock an Option value

1. Define a numeric expression in the **Default Quantity** field. You can use the expression builder to create the expression.
2. When the Sales rep configures a bundle, on the Configuration page, a lock icon appears next to the option's quantity. The Sales rep can click the lock icon next to an option, to ensure that any updates to the field from which its value is derived does not affect the value set the first time. You can lock the quantity field for an option product, whose quantity is populated based on multiple attribute values using the field expression builder. Once the user enters the attribute value on the **AttributeDetail3** page and clicks Next, the value for quantity is auto-populated.
3. Select the option and click the **Lock** icon.
4. Navigate back to the Attribute detail page and edit the attribute value, then click **Next**. Notice that the value of the Option does not change even though the attribute value has changed.

Excluding Optional Product in Min/Max Criteria

You can exclude options, bundles, and sub-bundles that are marked optional from the min/max criteria validation of the option group. When marked as options using the **Is Optional** checkbox, options or bundles as optional are not considered in the **Min Options**, **Max Options**, **Min Total Quantity**, and **Max Total Quantity** criteria if this setting is enabled. This feature is useful for excluding the price and quantity of optional products from the total. To exclude optional products from min/max criteria, you need to enable the **Exclude Options Products** custom setting.

To enable optional product exclusion

1. In CPQ Admin, go to **Application Management** > **Config Page** > **Display Settings**.
2. Select **Exclude Optional Products**.
3. Click **Save**.
4. Execute the Custom Settings Maintenance batch job.

For more information on marking products are optional, refer to Configuring Products from the Catalog.

Configuring Multi-Level Bundles

Your business may require you to configure a bundle in a bundle. This offers increased control over product selection, as well as making it easier to select products because selecting one bundle full of products is easier than selecting those products separately.

Depending on your business needs, you can configure products that may contain multi-level bundles. In addition to the ability to handle many levels of sub-bundles (also known as inner bundles), some of the bundles and/or options may be ramps or tiered offerings. This feature takes into account multi-level bundles with the ability to also show ramps or tiers contained within the sub-bundles. You can also apply constraint rules to multi-level bundles.

This feature enables you to configure a bundle and also view what products are being added (along with its hierarchy) to your cart on the configuration page. The selected products appear in the **My Options** section on the left hand side of the configuration page. You can also view and understand what bundle and option group you are currently in and navigate to the parent or child/lower level bundle during the configuration. The catalog/product hierarchy navigation is similar to the catalog and the configuration page. The product hierarchy is displayed in tree view and list view on the configuration page. You can minimize/hide the tree view.

To configure sub-bundles on the configuration page, you must set up the **Select Bundle Options** Visualforce page. It is recommended that after making changes to any bundle product association, you must run a bundle maintenance job that synchronizes all the bundles and options.

> ⓘ Changes done in option properties like MinQuantity, IsDefault do not require you to run bundle maintenance.

To set up select bundle options page

1. Go to **Setup** > **App Setup** > **Develop** > **Custom Settings**.
2. Click **Manage** for *Config System Properties*.
3. Click **Edit** for *System Properties*.
4. In **Option Page Default**, type one of the following based on your requirement.

Apttus_Config2__SelectConfigOptionsDetailView	Use this when there are usage tiers, ramps, and attributes for options products. This page does not support a large number of options on a bundle.
Apttus_Config2__ConfigureBundle	Use this when there are no usage tiers, ramps, and attributes for option products. This page supports more than 300 options on a bundle.

5. Click **Save**.

The bundle options page is set up to display sub-bundles allowing you to configure a bundle in a bundle.

The next step is to run a bundle maintenance job.

Hyperlinks for Options Groups in the Summary Section

- A summary section displays Option Group and child Option Groups with hyperlinks that takes you to the corresponding section wherein the entire configuration is shown.
- You must enable a custom setting Enable Hyperlink navigation in Bundle Summary to activate the hyperlinks. This setting is enabled by default.
- In New Admin UI, this setting is available under Display settings of Bundle Page.

Cloning a Bundle within the same Quote

You can clone a bundle within the same quote, make modifications, and use it as another bundle.

This feature enables you to clone a bundle with attributes, options, and pricing, make desired changes and save it as another bundle.

To clone a bundle within the same quote

You must have an existing bundle.

1. Select a bundle and click **Validate**. You can only clone a bundle.
2. Click  (Copy icon).
3. You can configure the bundle or make the desired changes and click **Validate**.

You have created another bundle from an existing bundle.

You can proceed to Managing Pricing.

Configuring Bundle Expansion on Cart Page

The Sales Rep can expand the bundles to view the options, sub-bundles, and secondary line items on the Cart page using the expand icon. CPQ provides a feature to prevent the Sales Rep from expanding the bundles. This is helpful if you do not want to expose the detailed prices and discounts applied at the option level of a bundle or sub-bundle to the Sales Rep.

You must create the **Disable Bundle Expansion** toggle button on the Product object. You can use this button to control Sales Rep's ability to expand the individual bundle or sub-bundle on the Cart page. When you enable this field for a bundle, the Sales Rep is not allowed to expand that bundle or sub-bundle on the Cart page as the expand icon is disabled. However, the Sales Rep can expand other bundles where this field is not enabled.

You can enable the **Disable Bundle Expansion** button on the Creating Product page in CPQ Admin UI. For more information on how to enable **Disable Bundle Expansion**, refer to Creating Products. You also need to add the field in **View Cart Custom Fields** in Config System Properties.

Perform the following tasks to define Bundle Expansion functionality.

To create the field **Disable Bundle Expansion**

1. Go to **Setup** > **App Setup** > **Customize** > **Products** > **Fields**.
2. In **Product Custom Fields & Relationships** section and click **New**.
3. Select **Checkbox** data type and click **Next**.
4. Enter the details:

Option	Description
Field Label	Enter *Disable Bundle Expansion* as the value of the field. You can change the label if required.
Default Value	Select **Unchecked**.

Option	Description
Field Name	Enter *APTS_DisableExpandInCart* as the value of the field. You must only use the *APTS_DisableExpandInCart* API name in the **Field Name**. The feature is only enabled if you enter the exact name.

5. Click **Next**.
6. Select the profiles to which you want to give access to the field and click **Next**.
7. Ensure the field is added to the available layout.
8. Click **Save**.

To add field values in View Cart Custom Fields

1. Go to **Setup** > **Develop** > **Custom Settings** > **Config System Properties** > **Manage**.
2. Find **System Properties** and click **Edit**.
3. In the **View Cart Custom Fields** field, enter the following values separated by commas:
   - *Apttus_Config2__ProductId__r.APTS_DisableExpandInCart__c*
   - *Apttus_Config2__OptionId__r.APTS_DisableExpandInCart__c*
4. Click **Save**.
5. Execute Custom Setting Maintenance. Refer to Running Maintenance Jobs.

To execute Query Field Metadata

1. Go to All Tab ( ) > **CPQ Console** > **Maintenance Console**.
2. In the **Query Field Metadata** section, click **Refresh Field Metadata**.

To add Disable Bundle Expansion in Create Products page

You must add the **Disable Bundle Expansion** button to Create Product page in CPQ Admin. The button is not displayed otherwise.

Follow the instructions in Configuring the Product Creation Screen Fields to add the button.

Running a Bundle Maintenance Batch Job

After making changes to any bundle product association, you must run a bundle maintenance batch that synchronizes all the bundles and options.

You can also use the **Update View** button from the Bundle and Option Manager page to run maintenance of view record for individual bundles. It is necessary to run maintenance when a new bundle or bundle options are added or removed. To make sure that all bundles have view records, use the **Bundle Maintenance** tab to run a bundle maintenance batch to update view records for all bundles. The bundle maintenance tab uses the Update Bundle Components page to run the batch.

> ⓘ Changes made in option properties such as **MinQuanity** or **IsDefault** do not require you to run bundle maintenance.

For more information, see Running a Bundle Maintenance Batch Job.

Managing Product Groups

You can create a logical grouping of one or more product records using Product Group. This construct allows you to create combinations of products with similar characteristics/ qualities for use in a Price Rule. Product Group (as opposed to Product Family) can consist of products from different product families.

In order to configure a product group, you must create a product group and add products as members of the group. You can add products as members of a group. You can also add products of different product families as a member of a product group.

In this section:

- Creating Product Groups
- Associating Products to a Product Group

Creating Product Groups

To create a product group

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.

2. On the **Products** menu, click **Manage Product Groups**. Details first Product Group is displayed by default.
3. Click **New Product Group.**
4. In **Group Name**, enter a mandatory group name.
5. In **Description**, enter a brief description to describe the purpose of the group.
6. Click **Save**.

A product group is created and saved. After you have created a product group, you can add products as members of the group. You can view the details on the product group in the **Details** tab.

Associating Products to a Product Group

To Associate Products to Product Group

You must have an existing product group.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Product Groups**. Details of the first Product Group is displayed by default.
3. Search and click the product group to open the product group details page.
4. Click **Associated Products**.
5. Drag and drop all the products you want to add in the Product Group from the **All Product** list. You can search for the products using the search bar.
6. Click **Save**.

All the products added to the product group are displayed.

To Associate Products to Product Group Within a Product

You must have an existing product group.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Product**.
3. Search and click the product to open the product details page.
4. Click **Associations**. Click the Product Group icon(). **Product Group** pane is displayed.
5. Drag and drop the desired product groups from the **Product Group** pane. You can search for the product groups using the search bar.
6. Click **Save**.

The product is now associated to the product group.

You can view all the products that you associated with the product group listed in the center pane under the **Associated Products** tab on the Manage Product Groups page. Select a product from the list on the left-hand side pane and the list of associated products is displayed in the center. You can also search for a product from the list of associated products using the search box at the top of the center pane.

Creating and Maintaining Attributes

Attributes are used to associate certain features to a product. Examples of common attributes are color, size, weight, and more. Attributes are associated to a product either to drive pricing or simply stored for informational purposes against the line item for other purposes. Attributes are associated with products through Attribute Groups.

The Attributes administration page allows you to create and delete attribute groups that can be associated or disassociated with products. You can manage attribute groups and attributes associated to a product. You can also manage the sequence for associated attribute groups.

The subsequent chapters on this page explain the following functionalities:

- Creating Product Attribute Fields
- Creating Product Attribute Groups
- Associating Attribute Groups to a Product
- Setting the Default Price from Attributes Using a Formula Field
- Cascading Shared Attribute Values
- Classifying Attributes
- Enabling Optimized Attribute Retrieval

Creating Product Attribute Fields

Product Attributes are used to associate certain features to a product. You must create product attribute fields in the Product Attribute Value object to represent an attribute. These attribute fields are then associated to a product through attribute groups.

To Create a Product Attribute Field

1. Go to **Setup** > **App Setup** > **Create** > **Objects**.

2. On the Custom Objects page, scroll down and select the **Product Attribute Value** object.
3. Scroll down to the **Custom Fields & Relationships** related list, and click **New**. You will be directed to the standard Salesforce field creation wizard.
4. Use the wizard to create a product attribute field, set up field dependencies, define workflow rules, line item validation rules, defaults, and more.

The Product Attribute field is created.

The next step is to create an Attribute Group and associate the product attribute field you created to the attribute group.

Creating Product Attribute Groups

Attribute groups are groupings of Product Attributes. Product Attribute fields must be associated to a Product Attribute group, which is to be associated to a product. Multiple product attributes can be grouped in a single Product Attribute group and a product can have multiple Product attribute groups associated to it.

Prerequisite

Select the **Has Attributes** checkbox. Clearing the **Has Attributes** checkbox disables the Configuration page.

To Create a Product Attribute Group

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click the product you want to associate attributes with.
4. Click **ATTRIBUTES**. The page displays all the attribute and attribute groups associated with the product.
5. Click **New Attribute Group** button. A pop-up is displayed.
6. Fill in the required details.

Field Name	Description
Group Name	Type a name for the Attribute Group
Two Column Attribute Display	Indicates that the configured attributes are displayed in two column format.

Field Name	Description
Three Column Attribute Display	Indicates that the configured attributes are displayed in three column format.
Description	Describe the attribute group.

7. Click **Save**.

Attribute Group are created and automatically associated with the product.

Associating Attribute Groups to a Product

You must associate an attribute group to a product for displaying the attributes (features) on the attributes page.

Prerequisite

1. Select the **Has Attributes** check box. Clearing the **Has Attributes** check box disables the attributes page.
2. You must have an existing product attribute group with product attribute field associated to it.

To Associate an Attribute Group to a Product

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click the product you want to associate attributes with.
4. Click **ATTRIBUTES**. The page displays all the attribute and attribute groups associated with the product.
5. Click the shortcut icon (‹) under **New Attribute Group** button. **Attribute Groups** panel is displayed.
6. On the Attribute Groups panel, select *Attributes Group* from the dropdown menu.
7. Search the Attribute Group using the search bar and then drag and drop the desired Attribute Group in the middle.
8. Click **Save**.

An attribute group is associated with a product.

To Associate Attribute to Attribute Group

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click the product you want to associate attributes with.
4. Click **ATTRIBUTES**. The page displays all the attribute and attribute groups associated with the product.
5. Select an attribute group you want to associate attributes with. Details of that attribute group are displayed in the middle.
6. Click the shortcut icon (‹) under **New Attribute Group** button. Attribute Groups panel is displayed.
7. On the Attribute Groups panel, select *Attributes* from the dropdown menu.
8. Search the Attribute using the search bar and then drag and drop the desired Attributes in the Product Attributes section.
9. Fill in the required details.

Field Name	Description
Attribute	This is fields populated automatically and is not editable.
Expression	Define an expression to defined criteria.
Is Read Only	Indicates that the value of the attribute is not editable. The user cannot edit the value.
Is Hidden	Indicates that the attribute is not displayed on the configuration page. However, the attribute is associated with the attribute group.
Is Primary	Indicates that the attribute is critical and the attribute will be displayed on the Installed Products page as a critical attribute.

10. Click **Save**.

Setting the Default Price from Attributes Using a Formula Field

You can add a formula field in the Product Attribute Value object and use it in a Price List Item to derive the Default Price from the attribute. On completing this, you no longer have to add the attribute to the attribute group and associate it with the product.

This way, you can make sure that this attribute doesn't show up in the configuration pages when the end user is going through the configuration process.

Cascading Shared Attribute Values

You can enable cascading of attribute values from bundles to the associated options when you have defined shared attributes. Initially, the bundle attributes will be cascaded to options in the bundle, but subsequent changes in the bundle attribute will not be cascaded to option attributes. You can enable the cascading of attribute values at the global level or individual attribute level.

You can define the **Cascade Shared Attributes Updates** custom setting to enable cascading of attribute values globally. At the attribute level, you can enable cascading for a particular attribute after associating them to a Product Attribute Group. You can also enable cascading selectively for different actions of Asset-Based Ordering. When you define cascading at the attribute level, the **Cascade Shared Attributes Updates** custom setting is ignored for that attribute.

To enable cascading globally

1. Go to **Setup > Develop > Custom Settings > Config System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Enable **Cascade Shared Attributes Updates**.
4. Click **Save**.

To enable cascading at the attribute level

> ⓘ This functionality is only available in Salesforce Classic.

1. Go to All Tabs > **Product Attribute Groups** tab.

2. Click on **Attribute ID** from the Attributes related list. Product Attribute Details page is opened.
3. Click **Edit**.
4. Update the following fields:

Field	Description
Cascade Shared Attribute Updates	Select one of the following values: • *None:* Cascading is executed based on the global setting • *Yes*: Cascading is executed regardless of the global setting for the line item status selected in **Cascade Shared Attribute Status** • *No*: Cascading is not executed regardless of the global setting for the line item status selected in **Cascade Shared Attribute Status** ⓘ If you select the values *Yes* or *No*, CPQ overrides the value of the global setting for that attribute. The global setting is only considered when **Cascade Shared Attribute Updates** is *None*.

Field	Description
Cascade Shared Attribute Status	Select one or more line item status from the following and add them to the right-hand side box labeled as **Chosen**. CPQ only cascades attribute values for the line item with line status that matches a value selected in this field. • *All* • *Renewed* • *Incremented* • *Incremented and Merged* • *Amended* • *Changed Configuration* • *Upgraded* • *Split* • *New* • *Existing* If you leave the **Chosen** box blank, based on the value of **Cascade Shared Attribute Updates**, cascading is executed for all line items regardless of line status.

5. Click **Save**.

Cascading is executed based on the combination of the **Cascade Shared Attribute Updates** and **Cascade Shared Attribute Status** fields. The following table explains the various combination of these fields:

Cascade shared attribute updates	Cascade shared attribute status	Description
Yes	Renewed, Upgraded	Cascading is only executed for renewed and upgraded line items.
Yes	Blank	Cascading is executed for all the line items with exception of line items with *Cancelled* and *Merged* line statuses.
No	Blank	Cascading is not executed for any line item.

Cascade shared attribute updates	Cascade shared attribute status	Description
Blank		Regardless of the values selected in **Cascade Shared Attribute Status**, cascading is executed based on the **Cascade Shared Attributes Updates** custom setting.
No	Renewed, Upgraded	Cascading is executed for all the line items with exception of line items with *Renewed* and *Upgraded* line statuses.

Classifying Attributes

CPQ classifies attributes as configuration attributes and price driving attributes when you define them. Attributes are marked as price driving when you define attribute-based pricing for any product. Otherwise, the attributes are considered as configuration attributes. Classifying the attributes enables CPQ to identify the attributes used in pricing computation and fetch only the required attributes while calculating prices. The classification of attributes also reduces the possibility of encountering governor limits while calculating asset pricing by enabling CPQ to only fetch the attribute relevant for calculation.

When you execute the Criteria Maintenance Batch job, the field **Attribute Type** in Product Attribute Member Type object of these attributes is automatically set to *Pricing*. You must manually update the **Attribute Type** field for attributes that are being used in in-flight proposals.

To manually update attribute type as pricing

1. Go to All Tabs > **Product Attribute** tab.
2. Click on **Attribute ID** from the Attribute Types related list. Click **Edit**
   Or click **New Product Attribute Type Member** to create a new entry. Product Attribute Type Member detail page is opened.
3. Update the following fields:

Fields	Description
Attribute	Search and select an attribute. If you are editing, the attribute is automatically selected.
Product	Select the product associated with the attribute for pricing
Attribute Type	Select *Pricing* to indicate that the attribute drives the price.

4. Click **Save**.

Enabling Optimized Attribute Retrieval

You can enable optimized retrieval of attributes on the cart page. When you enable optimized retrieval, CPQ only retrieves the attributes in the cart that are associated with the product added to the Cart. By default, all the attributes from Product Attribute Value, Product Attribute Value Ext, Product Attribute Value Ext 2, Product Attribute Value Ext 3, and custom Product Attribute Value Ext objects are loaded on the Cart page.

To enable optimized attribute retrieval

You must define the Admin Setting *APTS_LoadProductRelatedAttributesOnly* by following the steps below:

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record.
3. Fill the following details

Field	Value
Name	*APTS_LoadProductRelatedAttributesOnly*
Value	*true* or *false*
Code	Leave the field blank

4. Click **Save**.

Configuring Product Visibility

When you go to configure products for a Quote/Proposal, you can control which products from the price list are visible on the catalog page.

The following table describes some scenarios through which you can control the product visibility on the catalog page.

Scenario	Functionality	Configuration Required
You want to ensure that the products having Red color are visible on the Catalog page.	Refine Your Search	Select **Red** check box in the color section in Refine your Search dialog box.
	Search Filters (CPQ)	Select the appropriate price list in the Inclusion Criteria; select *Color* from Field, *Equal to* from Operator, and *Red* from Value.
	Visibility Rules through Custom Classes. Refer to Product Filter Callback Class	Modify the custom callback class code to list the catalog products as `global String getProductFilterExpr(CustomClass.ActionParams params) { return 'Color__c = \'Red\'';`

Scenario	Functionality	Configuration Required

You want to ensure that the Bundle products having Red color and relevant Option products having Green color are visible on the Catalog page.	Search Filters (CPQ) and Visibility Rules through Custom Classes. Refer to Product Filter Callback Class	Select the appropriate *Price List* in the Inclusion Criteria; select *Configuration Type* from Field, *Equal to* from Operator, and *Bundle* as Value.
		Modify the custom callback class code to list the catalog products as
		`global String getProductFilterExpr(CustomClass.ActionParams params) { return 'Color__c = \'Red\'';`
		Modify the custom callback class code to list the option products as
		`global String getOptionFilterExpr(CustomClass.ActionParams params) { return 'ComponentProductId__r.Color__c =\'Green\'';`

Product Visibility can be controlled in the following functionality.

- About Refine Your Search
- About Search Filters (CPQ)

About Refine Your Search

When you are selecting items from the Product Catalog, *Refine Your Search* enables you to narrow the list of products to those that match your search filter questions.

Search filters are associated with categories, so that the *Refine Your Search* questions may change when you select different product categories. The questions are also found on the product record, which is how the Product list gets refined. When you select an answer to a question and that answer matches the answer on the product record, that product is included in the filtered Product list.

You can select multiple values from the picklist for refining your search. The search is restrictive and helps narrow down the result to show only the relevant data. The Product field value should contain the filter value. The search filter provides all possible values.

Use Case:

For example, if you select Asia and Europe from Regions drop-down menu, then products available in both these regions are shortlisted.

> For a product to be displayed it must match all of the selected *Refine Your Search* criteria.

To configure your system to use Refine Your Search you must complete the following:

Configuring the Custom Fields	Two custom fields need to be created. The field for Search Attribute Values is used to provide the search question and answers, which will be associated with a category. The field for Products is used to have the same search question and answer values available on the product record.
Configuring Search Filter Fields	This makes the custom field available from the *Search Filter Fields* list in the Hierarchy Manager . This must be completed so that the custom field can be associated with a specific category.
Configuring Refine Your Search Options	This task associates the custom field with a specific category and adds it to the Refine Your Search window in the Product Selection page.

Selecting Search Field Values for a Product	So that using *Refine Your Search* provides useful and relevant filtering, you should select a value for each search field on a product record.

When you are selecting items from the Catalog page, *Refine Your Search* enables you to narrow the list of products to those that match your search filters. The system can dynamically display only those filter values that are based on the products associated with the selected category.

Configuring the Custom Fields

To configure the custom fields

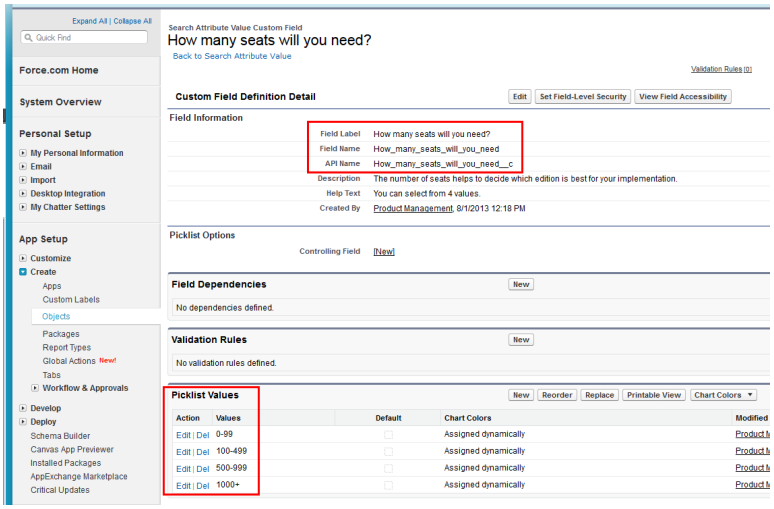
1. Go to App **Setup** > **Create** > **Objects** and select **Search Attribute Value**.
2. Go to **Custom Fields & Relationships** and click **New**.
3. Select Picklist data type and click **Next**.
4. Enter the details:

Option	Description
Field Label	This is the question. This is the text that will be displayed in the *Refine Your Search* box when you are selecting your products. It is also displayed in the product's *Product Elements* section.
List of values	These are the answers. From the Product Catalog these are the options you can choose from to refine your search on the *Products* page. From the product record you can select one of these values for the product.
Field Name	This is the basis for the Salesforce API name, which is referenced in *Search Filter Fields*. The API name takes the field name and appends__cto it.
Description & Help Text	Standard Salesforce description and help text fields.

5. Click **Next**.
6. Select which profiles will have access to the field and click **Next**.

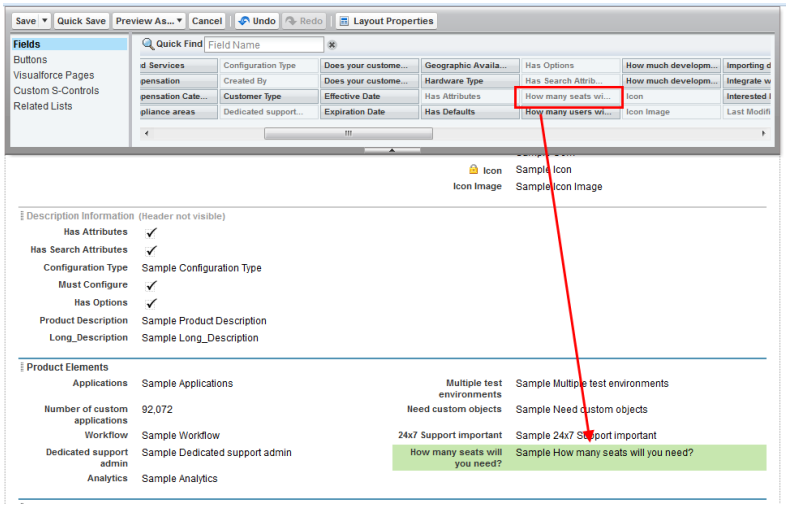
7. Ensure the field is added to available layout and click **Save**.

The new custom field is created and can now be associated with the Category Hierarchy search fields.



You must now complete the same task again, creating the same field label, values, and field name, but for Products. Go to App **Setup** > **Customize** > **Products** > **Fields** and then begin from Step 2.

If you want to change the default placement of the fields, you can reorganize them, by editing the page layout. To help group them together, typically the products custom field would be placed in a *Product Elements* section for the *Products* page layout.



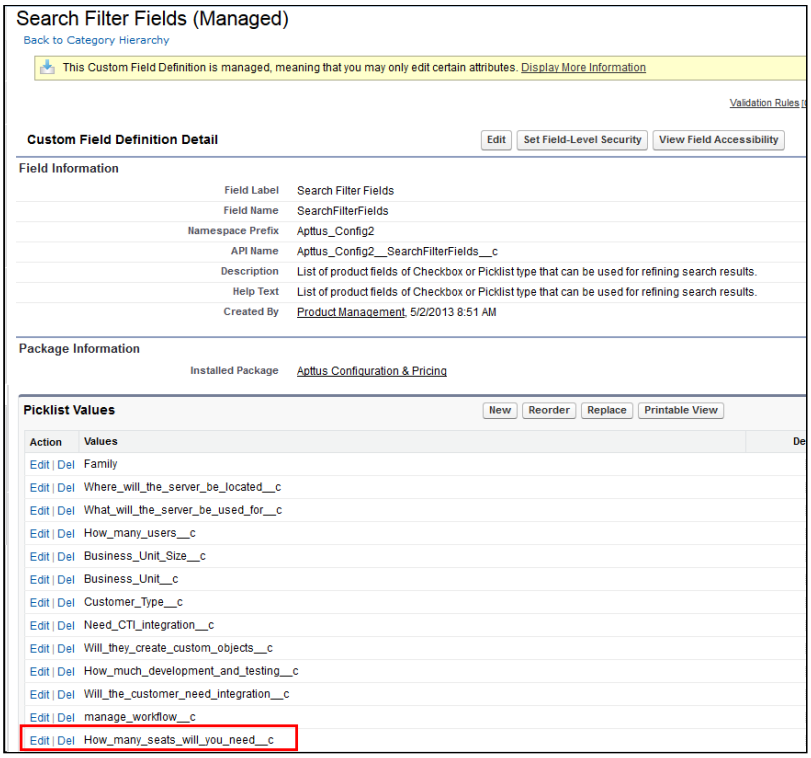
Configuring Search Filter Fields

To configure search filter fields

The custom field must have already been created.

1. Go to App **Setup** > **Create** > **Objects** and select **Category Hierarchy**.
2. Scroll to **Custom Fields & Relationships** and select **Search Filter Fields**.
3. Scroll to Picklist Values and click **New**.
4. Enter the *API Name* for the custom field created in the *Search Attribute Value* custom object and click **Save**.

That field is added to the list and is now available in the Hierarchy Manager.



You can go to the Hierarchy Manager and add that field as a *Refine Your Search* question for specific categories.

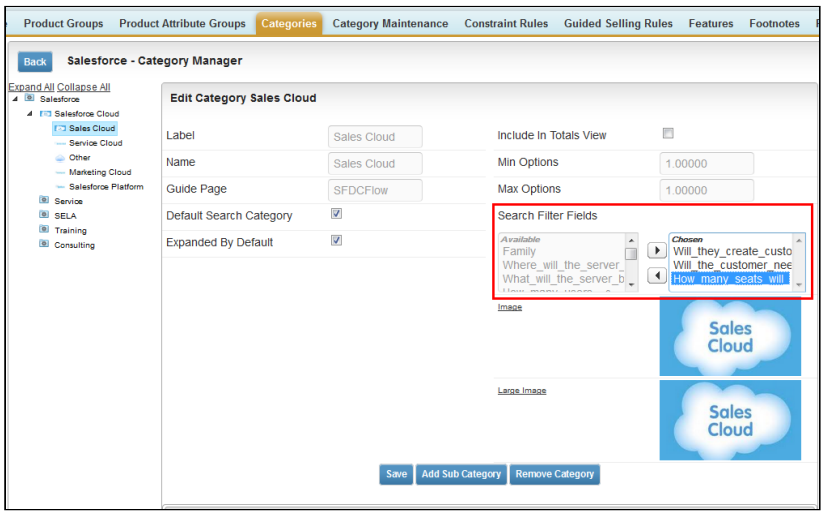
Configuring Refine Your Search Options

To configure refine search options

You must have configured Search Attribute Value custom fields and Search Filter fields.

1. Go to **CPQ Console** > **Manage Categories** and select the category you want to have *Refine Your Search* questions for.
2. In the **Category Detail** section, click **Hierarchy Manager**.

3. From the **Search Filter Fields** list, select the refine search filters you want to use and click .



4. Click **Save**.

The next time someone goes to select products for that category, the selected search filter fields will be displayed.
When desired, you can return to the Hierarchy Manager to remove or add additional search filters.

Selecting Search Field Values for a Product

To select a search field value for a product

The custom search field must be available on the page layout used to display the products.

1. Select a product.
2. Click the search field question and select the appropriate value for the product.
3. Click **Save**.

The value is displayed in the product record.

When that value is selected as a *Refine Your Search* answer, the product will be displayed in the refreshed Product list.

Configuring Refine Your Search with Dynamic Filter Values

When you are selecting items from the Catalog page, *Refine Your Search* enables you to narrow the list of products to those that match your search filters. The system now dynamically displays only those filter values that are based on the products associated with the selected category.

After you upgrade CPQ, you must run the Product Filter Maintenance task that collects all product field values related to a category. The product filter maintenance should be run when filter field association to the category changes or when the filter field values change in the product record. Also when new products are added or products are removed from a category you need to run the maintenance.

To run the Product Filter Maintenance task

You must have CPQ Summer13 SP5.5 installed.

1. Click ⊞ and click **Product Filter Maintenance**.
2. Click Update **Product Filters**.

The batch job is completed collecting all the product field values related to a category.

About Search Filters (CPQ)

When you configure products for a Quote/Proposal, you can control which products from the price list are visible on the catalog page based on search filters, where you set up the criteria of your choice.

For example, your company has 2 types of accounts in a quote, *Personal* and *Business*, and the products are of types *Personal*, *Business* or *Both*. You can define filters to display only products that are valid for the type of account on the quote. If the account type is *Business*, only the products that are of type *Business* or *Both* are displayed. In this scenario, you need to create 2 Search Filters.

Within a search filter, you can select the header level fields of object records as Entry Criteria for deciding whether a price list's products will be filtered to a certain subset of the full list. This subset also uses the header level fields of object records to determine which products are displayed.

The system checks whether a search filter is applicable to the price list you are using whenever you go to the product selection page. When you click **Configure Products** in the Quote/Proposal, the search filter is executed.

You can also apply the visibility rules for the categories displayed on the Catalog page. This functionality allows you to display only the relevant categories to a Sales rep who sells only a specific set of products from the large catalog. By hiding only the relevant category, you can still allow the Sales rep to search for the products and add them to the Cart.

> Filters defined in Search Filter (CPQ) are not applied on the Service Catalog, when the Sales rep uses **Add/Remove** and **Add/Remove at all levels** buttons available on the Service Cart page to navigate to the Service Catalog.

If you are hiding the product and its parent category, you need to configure two Search Filters, one for the product and another for the category.

Each visibility rule type runs in sequence. The rule stops processing after the criteria for the first rule is satisfied. If multiple visibility rules on the product are satisfied, only the first rule gets executed. If one visibility rule on the product and one on the category is satisfied, both rules get executed, since they are of different types.

> For the Sales rep to see the Refine Your Search filters on the catalog page, ensure that you do not hide corresponding leaf categories. Refine Your Search that is applicable for the child category can only be used when such child categories are not made hidden through the Search Filter.

In this section,

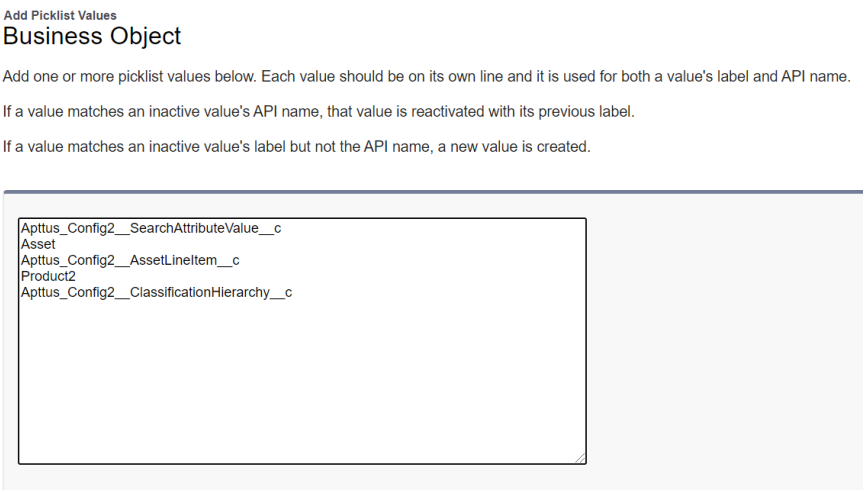
Configuring Search Filter (CPQ)

You must follow the steps below in Salesforce Classic.

To Configure Business Object Field

1. Go to **Setup** > **App Setup** > **Create** > **Objects** and select **Search Filter (CPQ)**.
2. Select **Business Object** under the **Custom Fields & Relationships** section.
3. From **Values** section, Remove *SearchAttributeValue__c* by clicking **Del**. This value must be deleted if it has not already been deleted.

4. Click **New** to add the following values in the picklist. Enter the API names of the objects you want to be able to use as a filter in separate lines. For example,
    a. To define visibility for products add the following values.
        - *Apttus_Config2__SearchAttributeValue__c*
        - *Asset*
        - *Apttus_Config2__AssetLineItem__c*
        - *Product2*
    b. To define visibility for categories add the following value.
        - *Apttus_Config2__ClassificationHierarchy__c*



5. Click **Save**.

To Configure Filter Type Field

1. Go to **Setup** > **App Setup** > **Create** > **Objects** and select **Search Filter (CPQ)**.
2. Select **Filter Type** under the **Custom Fields & Relationships** section.
3. In the **Values** section, click **New** and enter the names you want to use to group the **Business Object** picklist values.
    a. For Products
        - *Asset*
        - *Product*
    b. For Category
        - *Category*
4. Click **Save**.
5. From **Field Dependencies** section, click **New**.
6. Select **Filter Type** from **Controlling Field** drop-down and select **Business Object** from **Dependent Field** drop-down.
7. Click **Continue**.

8. Select which business object fields should be associated with which filter. To select the Business Object value, double-click the value.

Filter Type:	Product	Asset
Business Object:	*Asset*	**Asset**
	Apttus_Config2__AssetLineItem__c	**Apttus_Config2__AssetLineItem__c**
	Product2	*Product2*
	Apttus_Config2__ProductConfiguration__c	*Apttus_Config2__ProductConfiguration__c*
	Apttus_Config2__ProductAttributeValue__c	*Apttus_Config2__ProductAttributeValue__c*
	Apttus_Config2__RollupData__c	*Apttus_Config2__RollupData__c*
	Apttus_Config2__OrderLineItem__c	*Apttus_Config2__OrderLineItem__c*
	Apttus_Config2__ExternalOrderSummaryItem__c	*Apttus_Config2__ExternalOrderSummaryItem__c*
	Apttus_Config2__Incentive__c	*Apttus_Config2__Incentive__c*
	Apttus_Config2__Milestone2__c	*Apttus_Config2__Milestone2__c*
	Apttus_Config2__MilestoneTemplate__c	*Apttus_Config2__MilestoneTemplate__c*
	Apttus_Config2__SearchAttributeValue__c	*Apttus_Config2__SearchAttributeValue__c*
	Apttus_Config2__ClassificationHierarchy__c	*Apttus_Config2__ClassificationHierarchy__c*

9. Click **Save**.

The settings have now been configured so that you can create Search Filters (CPQ).

Creating Visibility Rules

Follow the steps below to create a visibility rule.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Visibility Rules**.
3. Click **New Visibility Rule** button. **DETAILS** page is displayed.
4. Fill in the information as explained in the table below.

Field Name	Description
Name	Enter a name for the visibility rule.
Filter Type	Select the type of filter you want to define. You can filter the products or categories. This field determines the values in the **Business Object** dropdown. ⓘ You must use Salesforce Classic UI to define Product Attributes Value or Product Attributes Value Extension objects in Product Visibility Rules. For more information about Product Attributes Value Extension objects, refer to About Product Attribute Value Extensions.

Field Name	Description
Business Object	Select which object you to apply the filter on. This field controls the values available in the **Filter Criteria** field.
Value Object	Select which object you want to use to control the fields available for **Inclusion Criteria**.
Sequence	Enter a number to define a sequence in which the visibility rules are evaluated.
Description	Describe the visibility rule.
Active	Select the toggle button to activate the visibility rule.
Inclusion Criteria	This is fields is visible after you select **Value Object**. Select the fields and expressions you want to use as entry criteria for the filter. You can use a maximum of three fields. If you use multiple fields, the relationship between them is an *AND* relationship, meaning each expression must evaluate as true. You can use Advanced Options to define a more complex formula. Click expression icon( ). Click **Add New Criteria**. Fill in the following information to define an inclusion rule. a. Row Num b. Field c. Operator d. Value Click **Save**.

Field Name	Description
Filter Criteria	Select the fields and expressions you want to use to control which products will be visible when the **Inclusion Criteria** is met. You can have a standard *AND* relationship between the expressions or use Advanced Options to the fields together in a more complex formula. Click expression icon( ). Click **Add New Criteria**. Fill in the following information to define an inclusion rule. a. Row Num b. Field c. Operator d. Map To e. Value Click **Save**.

5. Click **Save**.

When someone goes to configure products, the products and the category that are visible to them will be limited to those defined by the Filter Criteria when the Inclusion Criteria is met.

Use Case: Configuring Category Visibility

To display the category whose name starts with Hardware

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Catalog** menu, click **Manage Visibility Rules**.
3. Click **New Visibility Rule** button. **DETAILS** page is displayed.
4. Fill in the information as explained below.

Field Name	Description
Name	Enter a name for the visibility rule.
Filter Type	Select *Category*.
Business Object	Select *Category Hierarchy*.

Field Name	Description
Value Object	Select *Product Configuration* or *User* if you want to define inclusion criteria.
Sequence	Enter *1* as the sequence number.
Description	Describe the visibility rule.
Active	Select the toggle button to activate the visibility rule.
Filter Criteria	Click expression icon( ). Click **Add New Criteria**. Fill in the following information to define an inclusion rule. i. Enter *1* in the **Row Num**. ii. Select *Name* as a **Field.** iii. Select *starts with* as an **Operator.** iv. Specify desired category name as a **Value.** For example *Hardware*. Click **Save**.

5. Click **Save**.

CPQ displays only the categories that satisfy the criteria in the visibility rule on the Catalog page.

Configuring Product Comparison

You can enable a Sales rep to compare features of multiple products on the catalog page.

In order to configure product comparison on the catalog page, you must set up feature sets with one or more features. You must associate these feature sets to one or more products and then define feature values for each product. The configuration allows you to enable or disable a product feature to be displayed during product comparison.

If you do not want to use the Compare Products Feature, you can hide the button from the Catalog page.

The following topics describe the instruction to configure the Product Comparison:

- Setting Up Feature Sets
- Associating Feature Sets to a Product
- Defining Feature Values for a Product
- Hiding Products Comparison

Setting Up Feature Sets

Feature sets are a collection of features that can be associated to a product.

You must first define a feature set and then define features for each set.

To create feature sets

1. Click ✚ and click **CPQ Console**.
2. From the Product Management section, click **Add Feature Set**.
3. Type a mandatory **Feature Set Name**, and type a Description.
4. In the Feature Information section, type a **Feature Name** and a Description.
5. To add more than one feature, click **New**. A new row is added. The sequence number is auto-populated. You can rearrange the sequence by clicking and dragging the row.
6. Click **Save**.

A feature set is set up with features.

You must associate a feature set to a product for product comparison on the catalog page.

Associating Feature Sets to a Product

Using Product Console, you must associate a feature set to a standalone and a bundle product.

To associate a feature set to a product

You must have an existing product.

1. From the CPQ Console, click **Manage Products** and select an existing product to associate a feature set.
2. Click the **Product Console** button and scroll down to the Additional Data section, and click **Features**.

3. Click **Manage Associations** and from the Available Groups list, select a feature set for the association and click **Add**.

The feature set is associated with a product.

Define values for the features you just created.

Defining Feature Values for a Product

You must define the feature values for a given product.

To define feature values for a product

You must have an existing feature set and must be associated to a product.

1. On the Product Feature Values page, select the feature set listed in the left pane and type values next to the each feature.
2. To display the feature on the product comparison page, select **Is Included** and click **Save**.

Features Values that you just defined are saved.

Hiding Products Comparison

There is a custom setting that enables you to show or hide the *Compare Products* button and the product compare checkbox in the Catalog page.

After applying the new setting, you will no longer see the *Compare Products* button and the item level product compare check boxes.

To hide the compare products button on the catalog page

1. Click ✚ and click **Config Settings**.
2. Click **Catalog Page Settings** and from the Product List Settings section, select **Hide Compare Products** to hide the compare products feature on the catalog page.
3. Click **Save**.

The compare products feature is not displayed on the catalog page.

Configuring Numeric Expressions

Numeric expression is the extended custom implementation of SFDC formula builder, which is fine tuned by Conga. Numeric expression follows the syntax of SFDC formula and it provides all logical, text, date & time, math functions along with standard arithmetic operations. Note that numeric expression works for number fields.

Where you can use numeric expressions?

- To set the attribute value
- To set the minimum, maximum, and total quantity at the option group level
- To set the minimum, maximum, and default quantity at the option group level
- To define numeric rollups at product, product group, and cart level
- To set line item fields of auto-included products through constraint rules
- To set the criteria of product attribute rule (PAR)

> The Apttus Expression builder feature is disabled by default. To enable this feature, click **Custom Settings** > **Config System Properties** > **System Properties** > **Enable Field Expressions**.

Use Case:

Consider a requirement to configure a Home Solar Power starter bundle for SunHome. This product converts sun light on the panels into electricity for a small household. The bundle auto-includes two key components:

- 80-100 watt Solar Panels
- 12v @ 105 AH batteries

To find the right quantities of the panels and the batteries required, you must note the electricity usage to calculate the Total daily WattHours and Battery Capacity you require. The bundle also requires the calculated number of wires to be selected where the wiring length can vary based on the site study. The derivation is:

To meet this requirement, you must create:

- *Summary of Usage* attribute group, which contains attributes such as Refrigerator Run Time, Television Run Time, and Microwave Run Time, and receives inputs from the user.
- *Summary of WattHours required* attribute group, which contains attributes that derive the values from the attributes above, using the numeric expression builder.
- SunHome included component option group, which contains options that derive the default quantity from the attributes of the *Summary of WattHours required*, using numeric expression builder.

Using the expression builder, you can create numeric expressions to complete the following tasks.

- Populate product attribute values
- Set the min/max quantity for any option and option group
- Set the default quantity or the min/max quantity for any bundle option

To configure numeric expressions, you must complete the following processes.

- Populating Product Attribute values Using Numeric Expression
- Populating Option Group Min/Max Quantities Using Numeric Expression
- Populating Option Default and Min/Max Quantity Using Numeric Expression
- About Manage Field Expressions Page
- Running the Criteria Maintenance Job for Expression Fields
- Creating Numeric Roll-Up Summary Fields for Objects
- Examples and Related Syntax for Numeric Expressions

Populating Product Attribute values Using Numeric Expression

Create a product with attributes. You can use these attribute values to create a numeric expressions.

Example

SunHome is a product with the following attributes:

- Refrigerator Run Time
- Microwave Run Time
- Television Run Time
- House Lights

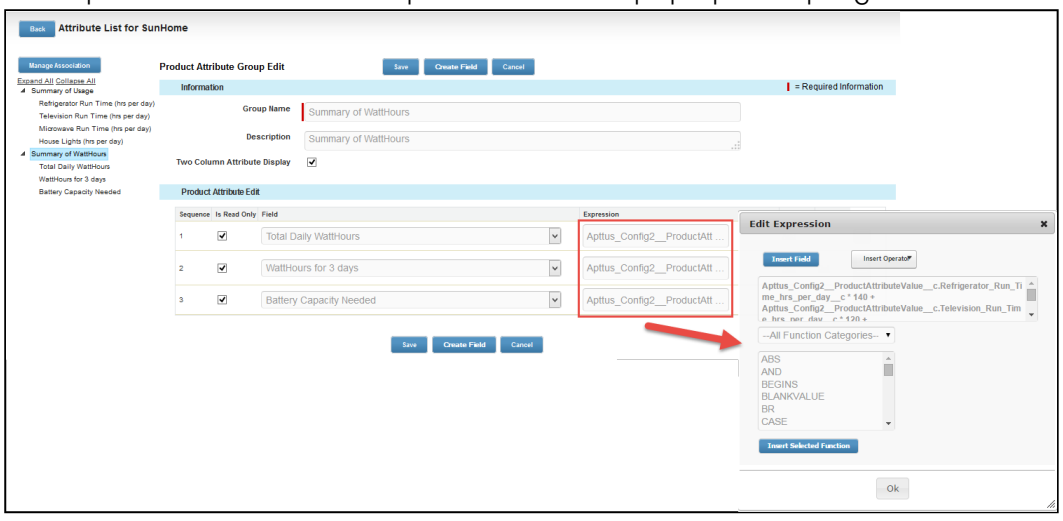
Each attribute has the datatype defined as number. You must also create an attribute group named *Summary of WattHours required* and assign it to the product SunHome. The attribute group contains the following attributes.

- Total daily WattHours required
- WattHours required for 3 days
- Battery capacity

Each attribute has the datatype defined as number. You can derive the values for these attributes using numeric expressions.

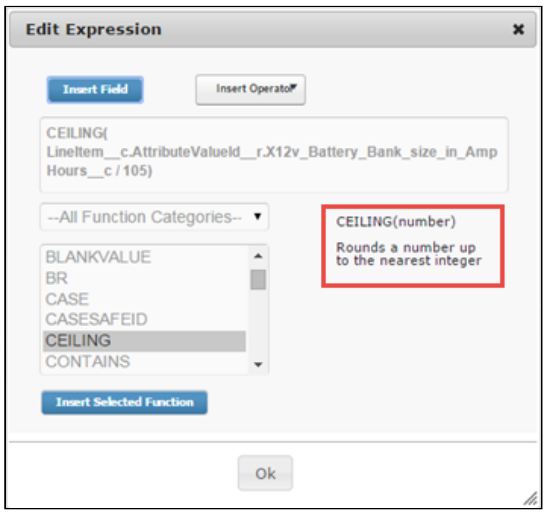
To populate product attribute values using numeric expressions

1. From the Products detail page, select SunHome and click **Product Console**.
2. Scroll down to the Additional Data section, and click **Attributes**.
3. Click **Summary of Watthours** required Attribute Group. All the attributes listed under the attribute group are displayed.
4. To associate an expression to an attribute, click the text field next to the attribute in the Expression Column. An expression builder pop up is displayed.

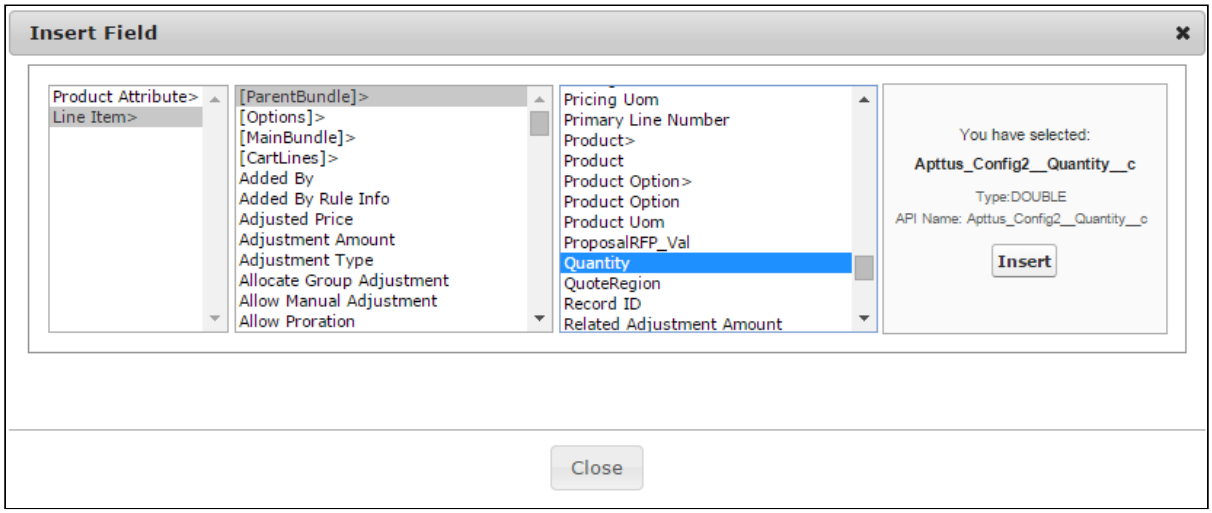


Selecting any function displays the example syntax on the right on how you can use

it.



5. Click **Insert Field**. To retrieve the fields for building the expression, click Insert Field and choose Product attributes or Line Item. Line Item allows you to get access to fields at Product and Quote objects as well.



6. In the Insert Field dialog, click **Product Attribute** and select the required attributes to derive the values for attributes of SunHome. The following expressions are built for each attribute:
    - Total daily WattHours required:
      Apttus_Config2__ProductAttributeValue__c.Refrigerator_Run_Time_hrs_per_day__c * 140 +
      Apttus_Config2__ProductAttributeValue__c.Television_Run_Time_hrs_per_day__c * 120 +
      Apttus_Config2__ProductAttributeValue__c.Microwave_Run_Time_hrs_per_day__c * 1000 +
      Apttus_Config2__ProductAttributeValue__c.House_Lights_hrs_per_day__c * 200

- WattHours Required for 3 days:
  Apttus_Config2__ProductAttributeValue__c.Total_Daily_WattHours__c * 3
- Battery Capacity Needed (50% Discharge):
  Apttus_Config2__ProductAttributeValue__c.WattHours_for_3_days__c * 0.5

> For more common syntax examples, see Examples and Related Syntax for Numeric Expressions.

7. Click **Save**.

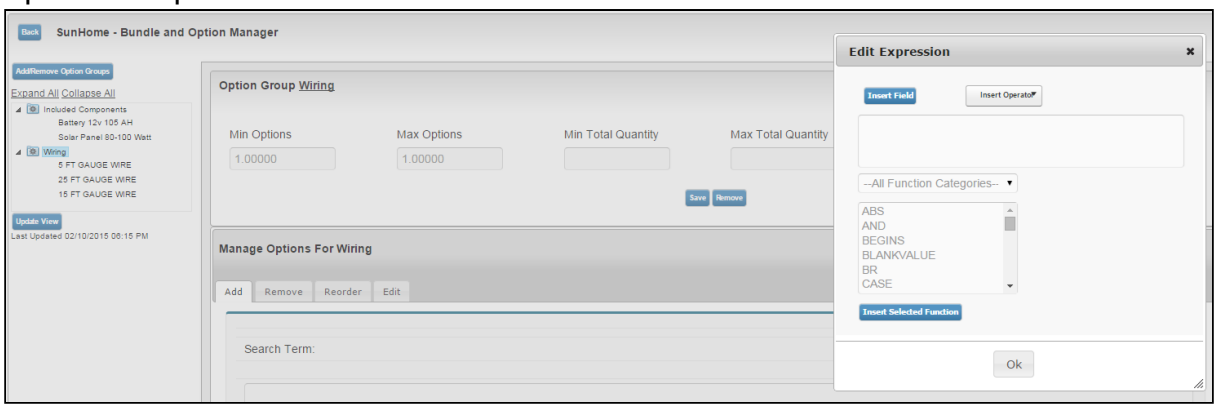
The expressions are built and the product attribute fields are populated with the value that the expression retrieves.

> After completing this process, you must run the Criteria Maintenance job for all expression fields. Refer to Running the Criteria Maintenance Job for Expression Fields.

Populating Option Group Min/Max Quantities Using Numeric Expression

Configure options for a product

1. Select the product that you want to assign numeric field expressions to.
2. Click **Product Console**, scroll down to the Catalog section, and click **Manage Bundles/Options**.
3. To associate an expression to min/max total quantity of an option group, select an **Option Group**.



4. Select one or both of the following fields to add an expression.
   - Min Total Quantity
   - Max Total Quantity

   The expression builder opens.

5. Click **Insert Field**. To retrieve the fields for building the expression, click **Insert Field** and choose Product attributes or Line Item. Line Item allows you to get access to fields at Product and Quote objects as well. Ensure that the return value of your numeric expression is of type *Number*.
   For our SunHome Product, we can use this to enforce minimum quantity selection for the option group *Wiring*.

   > For more common syntax examples, see Examples and Related Syntax for Numeric Expressions.

6. Click **Save**.

The expression is built and the option group quantity field is populated with the value that the expression retrieves.

> After completing this process, you must run the Criteria Maintenance job for all expression fields. Refer to

Populating Option Default and Min/Max Quantity Using Numeric Expression

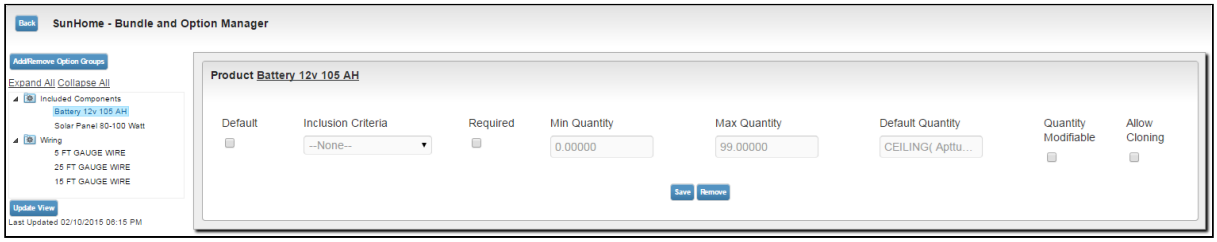
You must have an existing option product. You must also create an attribute group named *Summary of WattHours required* for SunHome. Following is a list of attributes in the group:

- Total daily WattHours required
- WattHours Required for 3 days
- Battery capacity

Each attribute has the data type defined as number. You must derive the values for options using numeric expressions. You must also create an option group with the options. Based on our scenario, this section outlines the procedure for an option group SunHome Component Quantity with options, Number of Solar Panels and Number of Batteries whose values are derived from the attributes above.

1. Navigate to the SunHome product, and click **Product Console**.
2. From the Catalog section, click **Manage Bundles/ Options**.

3. To associate an expression to an option, expand the Option Group, select an Option for which you want to add a field expression.



4. To add an expression, click one or more of the following:
   - Min Quantity
   - Max Quantity
   - Default Quantity

   The expression builder pop up appears.
5. Click **Insert Field**. To retrieve the fields for building the expression, click Insert Field and choose Product attributes or Line Item. Line Item allows you to get access to fields at Product and Quote objects as well.
6. To derive the values for the options of SunHome, in the Insert Field dialog, navigate to Product Attribute. The expressions built for each of the options are as follows:
   - Battery: CEILING(Apttus_Config2__LineItem__c=>ParentBundle.Apttus_Config2__AttributeValueId__r.Battery_Capacity_Needed__c/(12 * 105))
   - Solar Panel: CEILING(Apttus_Config2__LineItem__c=>ParentBundle.Apttus_Config2__AttributeValueId__r.Total_Daily_WattHours__c/(90 * 5)) Total_daily_WattHours_required__c, 0) / (90 * 5))

   For more common syntax examples, see Examples and Related Syntax for Numeric Expressions.
7. Click **Save**.

The expressions are built and the option quantity field is populated with the value that the expressions retrieve.

After completing this process, you must run the Criteria Maintenance job for all expression fields. Refer to Running the Criteria Maintenance Job for Expression Fields.

About Manage Field Expressions Page

The Manage Field Expressions page in the CPQ Admin user interface holds all the existing field expressions that you created. You can search, open, update, or create field expression

from the Manage Field Expressions page. In CPQ Admin, Go to the **Products** tab and from the dropdown click **Manage Field Expressions**. A list of all the existing field expressions is displayed with the fields **Name**, **Active**, **Update Field** and **Is Modifiable**. There are various functionalities available for you to manage field expression.

Field Expression Details

You can search for a field expressions on the Manage Field Expressions page using the **Search Field Expressions** bar. You must search using any keyword that the field expressions name contains. Click the name of the field expression to view the related information on the Details page. The fields like **Name**, **Update Object**, **Update Field**, **Value Expression** and, **Operator** are displayed on the Details page. You can update the field expressions on this page as well.

New Field Expressions

You can create new field expressions using the **New Field Expression** button on the Manage Field Expressions page and Details page of the Field Expressions.

To create new field expressions

Follow the steps below to create a new field expression.

1. Click **New Field Expression**. The Details page is displayed.
2. Fill in the following information in the fields displayed on the Details page. Because you are creating field expressions from the Manage Field Expressions page, only the values relevant to field expressions are displayed in the dropdown of the fields listed below.

Field	Description
Name	Enter the name of the field expression
Active	Toggle the button to the right-hand side to activate the field expression
Is Modifiable	Toggle the button to the right-hand side to enable editing of the field expression
Expression Type	This field is populated by default with the value *Field update*.

Field	Description
Expression Context	Select the value *Record Update* from the dropdown.
Source Object	Select the value *Line Item* from the dropdown.
Scope	Select one of the following values that are displayed in the dropdown: • *Header Level*: to apply field expression on fields at header level • *Product*: to apply field expression on the product line items • *Product Group*: to apply field expression on the product group level.
Filter Expression	Enter the filter expression. Use the expression builder to create a formula.
Update Object	Select one of the following objects that are displayed in the dropdown: • *Line Item* • *Product Attribute*
Update Field	Select the field on which you want to apply the field expression.
Operator	Select an operator from the dropdown.
Value Expression	Enter the value expression. Use the expression builder to create a formula.

3. Click **Save**.
4. Click the menu icon () next to the **New Field Expression** button. Click **Update Field Expression Criteria Fields** to execute the batch job.

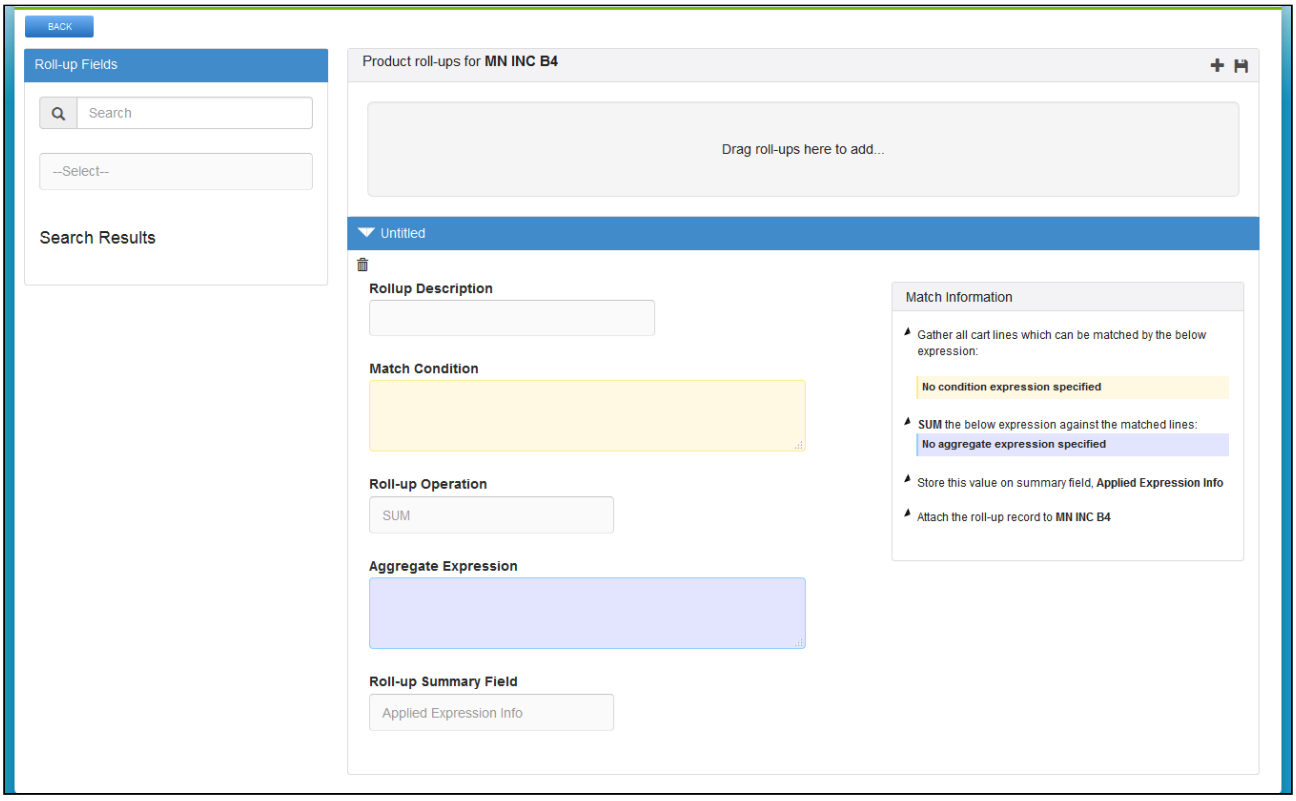
Running the Criteria Maintenance Job for Expression Fields

You must have existing field expressions.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. Click the menu icon( ) on the main menu bar at the top.
3. Click **Criteria Maintenance**. The Criteria Maintenance popup is displayed.
4. Click the **Update Expression Fields** button. This might take some time to update all expression criteria fields.
5. Close the Criteria Maintenance popup after the update is complete.

Creating Numeric Roll-Up Summary Fields for Objects

A Roll-up Summary Field can be defined at the Cart level or at a Bundle Product Level to aggregate value from each line item in the given context and with a match condition to filter out unwanted lines.



The **Match Condition** area enables you to enter an Expression created using the Expression Builder that establishes which line items will be filtered and have a Roll-Up.

The **Operator** defines the operation performed on all the aggregated line item values.

The **Aggregate Expression** is an Expression created using the Expression Builder that specifies the value to be rolled-up.

The Roll-up is by default created at the Cart level unless you add the following to the Match Condition:

ParentLineNumber = $scope.LineNumber

This retrieves all the options for the bundle line in the aggregation. You can use this expression to aggregate values at an individual bundle level in order to drive a bundle configuration rule.

The Expressions can be built for any line item field, product field, and attribute value fields. These Roll-Up Fields can be further used for multiple functions such as building other Expressions, Assign default quantities, attributes values.

Roll-up Field Name	Return Type	Roll-Up Operation	Match Condition	Aggregate Expression	Object
Total Quantity	Number	SUM	IsPrimaryLine = TRUE	Quantity	Line Item
Total Weight	Number	SUM	Product.Type = Hardware	Quantity * Product.UnitWeight	Line Item
Bundle Heat Dissipation	Number	SUM	ParentLineNumber = $scope.LineNumber && IsPrimaryLine = TRUE	Quantity * Product.UnitHeatDiss	Line Item
Lowest Setup Fee	Currency	MIN	ChargeType = 'Setup Fee'	ListPrice	Line Item
Max Line Item Discount	Percentage	MAX	ALL	NetAdjustment	Line Item
Avg Cost	Currency	AVG	ExtendedCost>0	ExtendedCost	Line Item

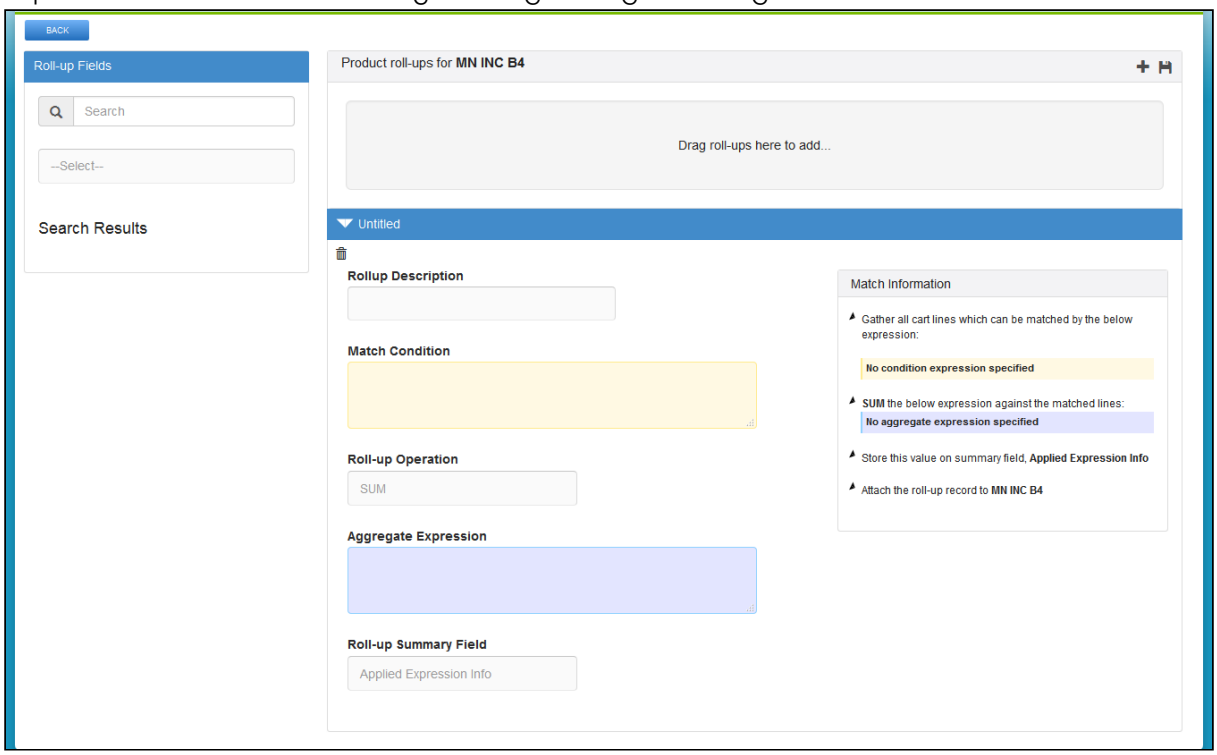
⚠ Extensive use of rollups at quote level may have performance implications for a cart with more than 20 product line items. It is also recommended to limit numbers of quote level rollups to below 20.

> ⓘ You can model your numeric expressions at the configuration level so that can rollup custom fields at the deal level. Rollups on configuration can be set through Apttus_Config2_*LineItem*_c.$Config.$Rollups (but does not return any value in client mode).

To create roll-up summary field

Create a custom Salesforce field in Roll-up Summary Object. Ensure that the field is unique and multiple expressions are not used to populate the same field. This field comprises or holds the value derived from the aggregate expression in the Field Expression builder.

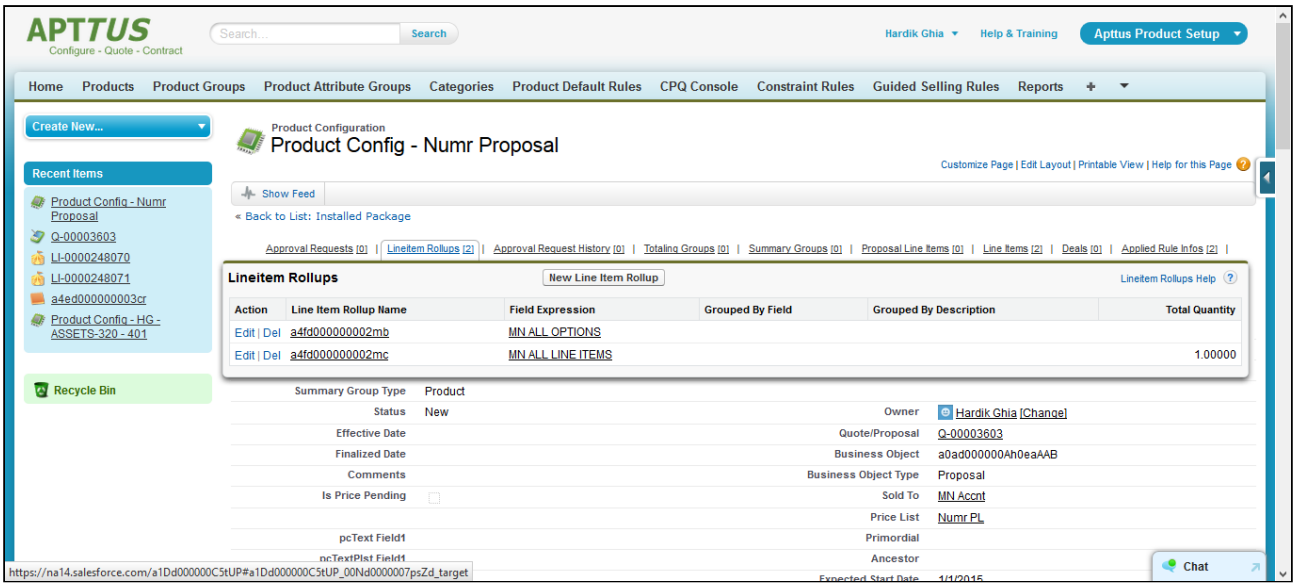
1. You can invoke the field expression builder for:
    - Bundles: To define a rollup field for a particular bundle, navigate to the Products tab, search and select a product and click Product Console.
    - Quote or Product Groups: To define a rollup field for a product group or at header level, navigate to the CPQ Console tab.
2. Click **Rollups** from the Additional Details section.
3. Click ✚ to add a new rollup field for a product.
4. Expand the Untitled section by clicking the right-facing arrow.



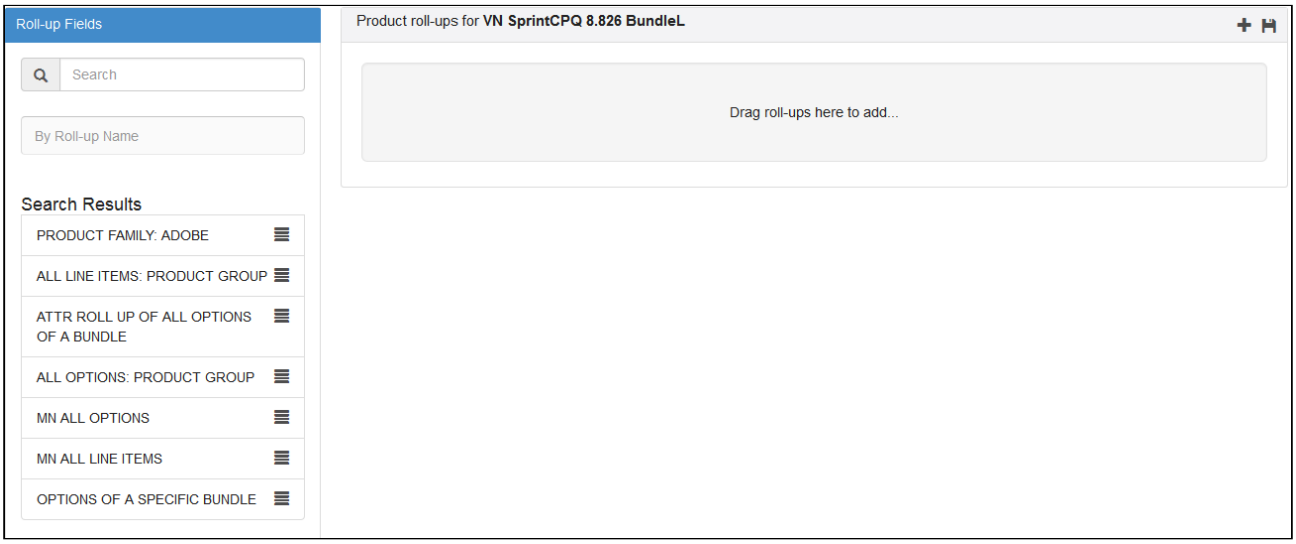
5. Select the Rollup Summary field created on the Rollup Summary object.

6. In the Match Conditions area, specify the expression based on which the results are filtered. The numeric expression rollup is applicable only for those line items that satisfy the criteria in the Match Conditions area.
7. From the Roll-up Operation area, select an operation based on which the rollup is created.
8. In the Aggregate expression area, specify the expression based on which the value is populated in the Rollup Summary Field. For example, (Quantity__c * ProductId_r.Weight__c) *2 Based on the Aggregate expression values, the rollup summary field is populated. Data is populated in The Match Information area based on the information you enter in all the above steps.
9. Click the **Save** icon in the top right corner of your screen.

The rollup for a product is created successfully. You can navigate to the cart page and check whether the rollup is applied successfully.



After you create rollup fields for a product, you can re-use the rollups across products or within the same product by searching for the rollup, product, or product group name in the Search field. Drag and drop the requisite roll-up conditions that you want to use to the right greyed out pane.



About Manage Rollups Page

The Manage Rollups page in the CPQ Admin user interface holds all the existing rollup type numeric expressions that you created. You can search, open, update, or create rollups from the Manage Rollups page. In CPQ Admin, Go to the **Products** tab and from the dropdown click **Manage Rollups**. A list of all the existing rollups is displayed with the fields **Rollup Name**, **Active**, **Update Field** and **Is Modifiable**. There are various functionalities available for you to manage rollups.

Rollup Details

You can search for a rollup on the Manage Rollups page using the **Search Rollups** bar. You must search using any keyword that the rollup name contains. Click the rollup name to view the related information on the Details page. The fields like **Name**, **Update Object**, **Update Field**, **Value Expression** and, **Operator** are displayed on the Details page. You can update the rollups on this page as well.

New Rollups

You can create new rollups using the **New Rollup** button on the Manage Rollups page and Details page of the rollups.

To create new rollups

Follow the steps below to create new rollups.

1. Click **New Rollup**. The Details page is displayed.

2. Fill in the following information in the fields displayed on the Details page. Because you are creating rollups from the Manage Rollups page, only the values relevant to rollups are displayed in the dropdown of the fields listed below.

Field	Description
Name	Enter the name of the rollup
Active	Toggle the button to the right-hand side to activate the rollup
Is Modifiable	Toggle the button to the right-hand side to enable editing of the rollup
Expression Type	This field is populated by default with the value *Rollup*.
Expression Context	Select the value *Rollup* from the dropdown.
Source Object	Select the value *Line Item* from the dropdown.
Scope	Select one of the following values that are displayed in the dropdown: • *Header Level*: to apply rollup on fields at header level • *Product*: to apply rollup on the product line items • *Product Group*: to apply rollup on the product group level.
Filter Expression	Enter the filter expression. Use the expression builder to create a formula.
Update Object	This field is populated by default with the value *Line Item Rollup*
Update Field	Select the field on which you want to apply the rollup.
Operator	Select an operator from the dropdown.
Value Expression	Enter the value expression. Use the expression builder to create a formula.

3. Click **Save**.

4. Click the menu icon (⋮) next to the **New Rollup** button. Click **Update Field Expression Criteria Fields** to execute the batch job.

Examples and Related Syntax for Numeric Expressions

This section lists out the common examples and related syntax that you can refer to for your specific scenario.

- Syntax to fetch quantity of any product line selected for the quote:

  ```
  Apttus_Config2__LineItem__c.$CartLines['Apttus_Config2__ProductId__r.Name','Pro
  duct A'][0].Apttus_Config2__Quantity__c
  ```

  Insert Field > **Line item** > **CartLines** > Select the field and provide the filter and index.
- Syntax to fetch sibling option quantity within the same parent bundle:

  ```
  Apttus_Config2__LineItem__c.$ParentBundle.$Options
  ['Apttus_Config2__OptionId__r.Name','Solar Panel 80-100 Watt'][0].Apttus_Config
  2__Quantity__c
  ```

  Insert Field > **Line item** > **ParentBundle** > **Options** > Select the field and provide the filter and index.
- Syntax to retrieve product field value as part of an expression:
  - Retrieving a bundle product field

    ```
    Apttus_Config2__LineItem__c.Apttus_Config2__ProductId__r.Numr_Numb__c
    ```

    Insert Field > **Line item** > **Product** > Select the field
  - Retrieving an Option product field when the expression is created for a bundle product.

    ```
    Apttus_Config2__LineItem__c.$Options[field,value]
    [index].Apttus_Config2__OptionId__r.Numr_Numb__c
    ```

    Insert Field > **Line item** > **ParentBundle** > **Options** > Select the field and provide the filter and index.
  - In Bundle-In-Bundle setup, if you want to update an option product's attribute, min quantity, and max quantity based on a field value from its bundle product which is an option of another bundle.

```
Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__OptionId__r.Numr_Numb__c
```

In Bundle-In-Bundle setup, if you want to update an option product's attribute, min quantity, and max quantity based on a field value from the top bundle product.

```
Apttus_Config2__LineItem__c.
$MainBundle.Apttus_Config2__ProductId__r.Numr_Numb__c
```

- Syntax to retrieve quote header field as part of an expression:
  - Displaying the Quote header field from a bundle product.

```
Apttus_Config2__LineItem__c.Apttus_Config2__ConfigurationId__r.Apttus_QPCo
nfig__Proposald__r.Numr_Field1__c
```

  - Displaying the Quote header field from an Option product.

```
Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__ConfigurationId__r.Apttus_QPConfig__Proposal
d__r.Numr_Field1__c
```

- Syntax to retrieve product attribute value as part of an expression:

```
Apttus_Config2__ProductAttributeValue__c.Total_Daily_WattHours__c
```

Insert Field > **Product Attributes** > Select the field

- Syntax to define the value of an attribute using another attribute of the same product or option

```
Apttus_Config2__ProductAttributeValue__c.Length__c * .25
```

- Syntax to retrieve parent bundle product attribute value as part of an expression:

```
Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__AttributeValueId__r.Battery_Capacity_Needed__c
```

Insert Field > **Line item** > **ParentBundle** > **AttributeValue** > Select the field

- Syntax to define rollups on header
  Specify the Object to be updated and the field of the object to be updated. In the

value expressions field, specify the fields from which you want to fetch the values from.

```
BLANKVALUE($.AttributeValueId__r.Total_daily_WattHours_required__c, 0) * 3
```

- Syntax to define rollups for Products
  Specify the scope as Product, and enter Product name in the Product field. Ensure that you define a parent field as LineItemId_c in the Rollup Criteria area.
- Syntax to define rollups for Options
  Specify the scope as Product, and in the Match Expression area, enter ParentBundleNumber__c == $condition.PrimaryLineNumber__c
- Syntax to define rollups Product Group
  Specify the scope as Product Group, and enter Product Group name in the Product Group field. Ensure that you define a parent field as LineItemId_c in the Rollup Criteria area.
- Syntax to define rollups for specific Quote Numbers or Account Numbers:
  Specify the scope as Header, and in the Match Expression area, enter

```
BLANKVALUE(ConfigurationId__r.Proposald__r.Apttus_Proposal__Account__r.Name ==
'Account Name',0)
```

- Syntax to define option rollups for specific bundles or multiple bundles
  Specify the scope as Product, specify the product name in the Product field and in the Match Expression area, enter

```
ParentBundleNumber__c == $condition.PrimaryLineNumber__c
```

  Ensure that you define a parent field as LineItemId_c in the Rollup Criteria area.
- Syntax to define a rollup based on the sum of attribute values
  Specify the scope as Product, specify the product name in the Product field and in the Expression Criteria area, specify the Value Expression as

```
BLANKVALUE($.AttributeValueId__r.PerfAttr1__c,0) +
BLANKVALUE($.AttributeValueId__r.PerfAttr2__c,0) +
BLANKVALUE($.AttributeValueId__r.PerfAttr3__c,0)
```

  Ensure that you define a parent field as LineItemId_c in the Rollup Criteria area.
- Syntax to define numeric roll upsummary record in a numeric expression
  In order to roll up the specific product attribute (which is numeric) based on the options product quantity roll up, do the following.

In the Quantity roll up for options,
Under Match Condition, specify
Apttus_Config2__LineItem__c.Apttus_Config2__ParentBundleNumber__c != NULL()
Select SUM as roll up operation.
Under Aggregate expression, specify
Apttus_Config2__LineItem__c.Apttus_Config2__Quantity__c
Then use the roll upsummary in the expression in the following way:

```
Apttus_Config2__LineItem__c.$Rollups['fieldName',value].Apttus_Config2__TotalQu
antity__c
```

- Syntax to set attribute value of a product from the attribute value of another product in the cart:

```
BLANKVALUE(VALUE(Apttus_Config2__LineItem__c.
$CartLines['Apttus_Config2__ProductId__r.Family','Laptop'
].Apttus_Config2__AttributeValueId__r.Size__c),17)
BLANKVALUE(VALUE(Apttus_Config2__LineItem__c.
$CartLines['Apttus_Config2__ProductId__r.Name','Toshiba Satellite S875-
S737617.3 Inch'].Apttus_Config2__AttributeValueId__r.Size__c),14)
BLANKVALUE(VALUE(Apttus_Config2__LineItem__c.
$CartLines['Apttus_Config2__ProductId__r.ProductCode','LP104'
].Apttus_Config2__AttributeValueId__r.Size__c),14)
```

- Syntax to set attribute value of an option item with the attribute value of another option item inside the same bundle.

```
 Apttus_Config2__LineItem__c.$ParentBundle.
$Options['Apttus_Config2__OptionId__r.Name','Option
1.1'].Apttus_Config2__AttributeValueId__r.Length__c
```

- Syntax to set Option's Default/Min/Max Quantity based on attribute value of the parent bundle:

```
BLANKVALUE(VALUE(Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__AttributeValueId__r.Data_Transfer_Limit_Month_in_
TB__c),0)
BLANKVALUE(Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__AttributeValueId__r.Number_of_Users__c+5, 0)
```

- Syntax to conditionally set the value of attribute or min/max/default quantities of an option item, depending on the value of an attribute from the parent bundle.

```
IF(Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__AttributeValueId__r.Edition__c=='Enterprise',
BLANKVALUE(Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__AttributeValueId__r.Number_of_Users__c+5, 0),
BLANKVALUE(Apttus_Config2__LineItem__c.
$ParentBundle.Apttus_Config2__AttributeValueId__r.Number_of_Users__c+10, 0))
```

- Syntax to set the quantity of an option using quantity of another option in the same bundle.

```
 BLANKVALUE(Apttus_Config2__LineItem__c.$ParentBundle.
$Options['Apttus_Config2__OptionId__r.Name','Option
1.1'].Apttus_Config2__Quantity__c * 2, 0)
```

- Syntax to set the value of an attribute value or min/max/default quantities using a numeric rollup:

```
 BLANKVALUE(Apttus_Config2__LineItem__c.$Config.$Rollups['Name','Cart Weight'].
Total_Weight__c,0)
```

- Syntax to set quantity of a product to the sum of quantities of other 2 products in the cart.

```
BLANKVALUE(Apttus_Config2__LineItem__c.
$CartLines['Apttus_Config2__ProductId__r.Name','Logitech Keyboard
K120'].Apttus_Config2__Quantity__c,0) +
BLANKVALUE(Apttus_Config2__LineItem__c.
$CartLines['Apttus_Config2__ProductId__r.Name','Logitech Wireless
Keyboard'].Apttus_Config2__Quantity__c,0)
```

Managing Product Rules

You can use Product Rules to regulate the selection of the products by the Sales rep on the Catalog page. When the Catalog exceeds a simple number of products, it can be difficult for the Sales rep to find their desired product based on the specific requirements of the company. Product Rules can guide the Sales rep to follow what is best and compliant for the company.

For example, your company sells a range of products and there are certain regulations you must follow to sell the products, for example, there are products that cannot be sold together, products that cannot be sold in certain regions, or products that must be sold along with another product. You can define rules for such scenarios to help the Sales rep avoid selling a partial product or sell products they are not supposed to.

Product Rules consist of a set of conditions and actions. The rules are invoked when the conditions are satisfied and based on the definitions of the rules, actions are executed on the Catalog page. You can manage them from the Manage Rules tab in CPQ Admin.

You can configure the following types of rules:

- Constraint Rules: Configuration rules used to conditionally include, exclude, recommend, replace, or validate a product or set of products based on other products or set of products added to the cart.
- Attribute-based Configuration: Helps you modify the product configuration on your cart based on the selection of product attributes using Product Attribute Rules and Attribute Value Matix.

About Manage Rules Page

The Manage Rules page in the CPQ Admin user interface holds all the details of the existing constraint rules, product attribute rules, and attribute values matrices that you created. You can see them listed in the left-hand side pane. You can search, open, update, or create different rules from the Manage Rules page. In CPQ Admin, Go to the **Products** tab and from the dropdown click **Manage Rules**. There are various functionalities available for you to manage rules.

Rule Details

You can search for a rule on the Manage Rules page using the **Search** bar. You must search using any keyword that the rule name contains. You can filter out the different types of rules listed on the page. Click different icons that represent different rules to filter them.

- Constraint Rule icon ()
- Product Attribute Rule icon()
- Attribute Value Matrix icon()

Click the rule name to view the related information in the center pane. The fields **Rule Name**, **Sequence, Effective Date**, **Bundle Context**, **Active** and, **Description** are displayed on the center pane. You can update the rules on this page as well.

New Rules

You can create new rules using the **New Constraint Rules**, **New Product Attribute Rule** and **New Attribute Value Matrix** buttons on the Manage Rules page. The below table mentions the list of topics you can refer to create different rules.

Rules	Topic
Constraint Rules	**Creating Constraint Rules**
Product Attribute Rules	**Creating Product Attribute Rules**
Attribute Value Matrix	**Creating Attribute Value Matrices**

ⓘ You can start following the instructions from Step 5 in the abovementioned topics to create the rules.

Managing Constraint Rules

Constraint Rules are configuration rules used to conditionally include, exclude, recommend, replace, or validate a product or set of products based on other products or set of products added to the cart. For example, when Product A is added to the cart, Product B can be automatically added based on an auto-inclusion type rule. Similarly, when Product B is added to the cart, the sales rep can be prevented from adding Product C. Constraint rules are applied to a product during the product configuration process, either on the client-side or the server-side.

The constraint rule consists of three associated Objects:

- Constraint Rule Object: Header level object that links the condition with the action. There are only three significant properties in this object, the active flag, action association, and condition association. To configure a constraint rule, you must first create a constraint rule header.
- Constraint Rule Condition: Captures the condition that triggers the rule.
- Constraint Rule Action: Captures the rule action that is applied to the cart when the rule condition is satisfied.

CPQ supports five types of constraint rules. Each of the rules has support for their action intents on either server-side, client-side, or both.

Rule Type	Definition	Example	Server-Side Supported Functionalities	Client-Side Supported Functionalities
Inclusion	Used to add a product to the cart.	If the Sales rep has selected a laptop, then automatically include a laptop charger.	• Auto-inclusion • Prompt • Message	• Auto-inclusion • Prompt • Message
Exclusion	Used to exclude or prevent the addition of a product to the cart.	If the Sales rep has selected a region where a certain model of laptop is not available, then disable that laptop model.	• Prompt • Hide • Disable • Message	• Prompt • Hide • Disable • Message
Replacement	Used to replace and add a product to the cart.	If the Sales rep has selected a product that is deprecated or not available then replace that product with another product that is available in the catalog.	• Prompt • Message	
Recommendation	Used to prompt product suggestions to the user for addition to the cart.	If the Sales rep has selected a laptop and has not selected a mouse, then recommend the user to select a mouse.	• Prompt • Message	• Prompt • Message
Validation	Used to prevent the user from finalizing the cart without resolving validation errors.	If the Sales rep chose a laptop and their limit was to buy only 5 units at a time, then validate the quantity.	• Message	• Message (Except for Product Groups)

Constraint rules are optimized with the client-side caching mechanism to improve performance and user experience. You can choose to process all constraint rules at the

client-side or server-side, by defining the setting **Constraint Rule Execution Mode**, in *Config System Properties*.

> The constraint rules are processed at the server-side by default. However, always use Client-Side Constraint Rules in a Service CPQ flow.

Note the following limitations when enabling Client-Side Constraint Rules (CSCR).

- Replacement type constraint rules are not supported.
- Product Scope, **Option Groups**, is not supported. If such a constraint rule is defined an error is displayed on the Catalog page.
- When you configure a constraint rule with **Action Type** = **Exclusion** and **Min/Max Match Rule** = **Exclude After One**, if the sales representative adds the product for the first time, CPQ displays a constraint rule message on the cart in the "Client" mode.
- **Match in Assets** in Match Conditions for Constraint Rule Actions is only supported for Inclusion type Client-Side Constraint Rule for Prompt, Auto-include, and Show Messages actions. For Exclusion type Client-Side Constraint Rule for Prompt, and Show Message actions.

> Constraint rules are not executed on line items with the status as *Cancelled*. To avoid errors, add a criterion to exclude line items with line status as *Cancelled* in all such constraint rules.

You can also execute a constraint rule across the following bundle structures:

- From parent bundle structure to nested bundle structure
- From nested bundle structure to parent bundle structure
- Across nested bundles

You must follow the best practices while defining Constraint Rules. For more information, see CPQ Best Practices: Governor Limits.

Subsequent sections explain the following topics:

- Configuring Product Option Group in Product Scope
- Scenarios for Using Constraint Rules
- Disabling Constraint Rule Execution upon Cart Finalization

Creating Constraint Rules

There are two different methods you can use to create Constraint Rules. You can create new constraint rules from the beginning or clone an existing constraint rule. Follow the steps below to create constraint rules.

To create constraint rules

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click any product for which you want to define a constraint rule.
4. Click **Rules**. The page displays different rules tab.
5. Click Constraint Rule icon () next to the title, **RULES**, to launch the constraint rules listed for the product.
6. Click **New Constraint Rule** button.
7. Fill in the required details.

Field Name	Description
Rule Name	Enter a name for the constraint rule.
Is Active	Flags whether the Attribute Value Matrix is active.
Bundle Context	Indicated whether the constraint is evaluated within a bundle.
Sequence	Enter the number in which you want to sequence the constraint rule.
Effective Date	Select the date from which the constraint rule is in effect.
Expiration Date	Select the date until which the constraint rule is valid.
Short Description	Describe the constraint rule.

8. Click **Save**.

The constraint rule is created.

Also, you can clone an existing constraint rule.

To clone a constraint rule

1. Open an existing constraint rule.
2. Click the menu icon ( ) under the **Rule Name** column.
3. Click **Clone Rule**.
4. Enter a name for the new constraint rule.
5. Click **Save**.

The constraint rule is cloned.

> If a constraint rule is deleted or deactivated, then, if it was used in an existing quote, the changes made by the rules are reverted and the user can finalize and proceed with their quote without being blocked. CPQ does not show any error while you modify the constraint rules and re-configure the products for the existing quotes.

Creating Constraint Rule Conditions

Constraint rule conditions should be met in order to trigger a rule. There may be more than one condition for a rule. A constraint rule condition is met when the Sale rep adds a product that is part of the product scope you defined in constraint rule condition, to the cart. When the constraint rule condition is satisfied, the constraint rule action is executed on the cart. Different conditions are associated with an expression. When the condition association expression is not specified, all the conditions should be met in order to trigger the rule.

The rule condition is defined by product scope. Product scopes are groups in which the product belongs or a set of fields you can use as conditions. The product scope could be specific to a Product, Product Family, Product Group, Product Option Group, and a Product Field Set. Product scops are explained in detail later in this section.

You can define additional condition criteria like Country Codes or Color. The additional criteria on the condition is based on the line item object. Condition Criteria filter conditions that are applied to the line item created from the selected products that are within the product scope definition.

When criteria is defined, the condition is met when the product scope specification and criteria are both met. For example, based on your company's requirement, you cannot sell Product B if you are selling more than 3 units of Product A. In this case, you can define condition criteria to validate the quantity of Product A.

When using a constraint rule with condition criteria for a product with multiple line items, the system only validates the primary line item to evaluate the constraint rule.

You must run a criteria maintenance job whenever you add, remove, or make any changes to criteria. Even if you remove a constraint rule that has criteria defined, you must run a criteria maintenance.

> ⓘ You can create a Constraint Rule Condition or Constraint Rule Action criteria that is based on a field from the Product Attribute Value Extension object.

To Create Constraint Rule Conditions

You must have an existing constraint rule header.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click any product for which you want to define a constraint rule.
4. Click **Rules**. The page displays different rules tab.
5. Click Constraint Rule icon () next to the title, **RULES**. All the constraint rules associated with that product are displayed.
6. Search and click the constraint rule.
7. Fill in the required details under the **IF(scope)...** section. Enter a number under **Condition** column.
8. Select one of the following options from **Product Scope** drop down. The last column changes based on the product scope you select. Enter the name in the last column accordingly.

Option	Description
Product	After selecting, enter the name or product code of the product in the Product field. This implies that the product should be present in the shopping cart in order to meet the condition.

Option	Description
Product Family	After selecting, enter a Product Family name in the Product Family field. When specified, one or more products from the specified Product Family should be present in the shopping cart in order to meet the condition. The number of products that should be present depends on the Match Rule.
Product Group	After selecting, enter a product group in the Product Group field. This implies that one or more products from the specified product group should be present in the shopping cart. The number of products that should be present depends on the Match Rule.
Product Field Set	After selecting Product Field Sets in the product scope, an add icon (+) is displayed, add product fields. This implies that when the value of the field specified in the product scope matches the value of the field on the cart, the constraint rule condition is satisfied. You must add atleast one Product Field. a. Click the add icon (+). b. Select a product field from the picklist. c. Select an operator. d. Define the value of the product field in **Value**. Each of the Product Field has onepick list value from the product object. The Product Field should be set up to have only one choice. If the choices are changed at a later time, the constraint rule conditions with Product Field Set specification should also be updated to have the correct values. Match Rule is not applied on Product Field Set based constraint rules. CPQ considers **Match Rule** as **Match Any in Group** by default in such case.

Option	Description
Product Option Group	After selecting this option, enter an **Option Group ID** associated with the bundle in the **Product Option Group** field. This implies that one or more products from the specified product option group must be present in the shopping cart. The number of products that must be present depends on the Match Rule. ⚠ If you define the constraint rule in CPQ Admin, enter the option group name associated with the bundle in the **Product Option Group** field.
Header Level	This functionality is deprecated.

9. Click the setting icon ( ). A pop-up is displayed. Click **CONDITIONS** tab and fill in the details as explained below.

Option	Description
Condition Name	The name is populated automatically. You can edit the name as you desire.
Match In Primary Lines	Selecting this means the condition is met only when the condition products are among the primary lines (bundle or standalone) of the shopping cart.
Match In Options	Selecting this means the condition is met only when the condition products are in the option line items of the shopping cart.
Match in Cart Options	Selecting this means the condition is met only when the condition products are among the cart line items of the shopping cart.
Match in Location	Selecting this means that constraint rules are executed within a location selected on the catalog. When the Sales rep selects a location and adds the condition product to the cart, constraint rule actions are executed. Upon changing the location, constraint rules are invoked again if conditions are met for the newly selected location.

Option	Description
Match in Related Lines	Selecting this means the condition is met only when the condition products are among the related line items (bundle or standalone) on the shopping cart.
Match in Service Assets	Selecting this means the condition is met only when the condition product is a selected asset. An asset remains a selected asset while the Sales Rep is still on the Service Catalog. The selected asset considered a related line item if the Sales rep navigates to the Service Configuration, Service Cart after relating a service, or Installed Products page and returns to the Service Catalog. Follow the below practices while defining **Match in Service Assets:** • to exclude or include options from a bundle on the Configuration page, use **Match in Related Lines**. • to exclude services on the Service Catalog, use **Match in Service Assets**. • to exclude and include both services from Service Catalog and option in service bundle from the Configuration page, use **Match in Service Assets** and **Match in Related Lines** both. • Conga recommends that for the Inclusion type constraint rule, you must select **Match in Service Assets** and **Match in Related Lines** both for the functionality to work as expected. ⓘ CPQ creates related line items for service product added through auto-inclusion, which are deleted if you remove the auto-included service product

Option	Description
Match Rule	Select one of the following choices. The value is None by default. Match count is done before applying the criteria. • *None* means that the rule condition is met if any one of the products in the product scope is selected. It is treated as Min/Max with Match Min Products as 1. • *Match Any In Group* means that the rule condition is met if any one or more of the products from the group is selected. *Match Any In Group* is the same as Include Min/Max with Match Min Products as 1. It is also applicable to the Product Family and Product Field Set. • *Match All In Group* means that all of the products from the group must be present in the shopping cart to meet the condition. This is applicable only to the product group and Product Family. • *Min/Max* also requires you to specify a value in the Match Min Products field.
Match Min Products	This field is applicable only when you select Min/Max from the Match Rule picklist. The condition is met, when the minimum number of products specified is selected from the Product Group, Product Family, or as specified by the Product Field Set specification.
Match Max Products	This field is applicable only when you select Min/Max from the Match Rule picklist. The condition is met, when the max number of products specified is selected from the Product Group, Product Family, or as specified by the Product Field set specification.

Match Rule is applicable to product scope of type Product Group, Product Family, and Product Field Set only.

10. Click **Save**.
11. Click the flare icon ( ) to define condition criteria. A pop-up is displayed.
12. Click **New Filter**.
13. Type a **Field**, an **Operator**, and type a **Value**. You can enter more than one criteria.
14. Click **Save**.

You can add multiple conditions in a constraint rule. Click add icon (+) and follow the steps again.

You can also define condition association to define additional criteria to further filter the product scope condition. Next to **IF(scope)...**, select *Custom* from the drop-down menu. When there are more than one condition for a constraint rule, this field defines when to consider one or more of the conditions using AND/OR association. The sequence number defined in the **Condition** column is used as an identifier of the condition.

Few examples, you defined conditions with sequence numbers as *1*, *2*, *3*, *4*, *5*, and *6*. The following examples describe various condition association

Condition Association	Description
(1 AND (2 OR 3))	Either Conditions 1 and 2 or Conditions 1 and 3 must be satisfied on the cart
1 AND NOT 2	Condition 1 is in the cart while Condition 2 is not
1 AND NOT(2)	Condition 1 is in the cart while Condition 2 is not
1 AND 1	Only Condition 1 is evaluated on the cart. If you have defined multiple conditions, CPQ does not evaluate the remaining conditions
NOT(5 OR 6)	Neither Condition 5 or 6 are in the cart
NOT 5 AND 6	Condition 5 is not in the cart, but Condition 6 is in the Cart.

It is recommended that you use logical operator - AND - to display the resultant record if all the conditions separated by AND is TRUE. For instance, (1 OR 2) AND (3 OR 4) AND 5. If you enter an invalid condition association, an error is displayed when you save the constraint rule.

> ⓘ You must set the Sequence for each condition or the Condition Association will be unreliable. Also, a constraint rule with only NOT in condition will not work. Example, NOT (1 AND 2).

The constraint rule conditions are created. You can now define a constraint rule action.

Creating Constraint Rule Actions

Constraint rule actions are performed, when certain constraint rule conditions are met. A rule action can be of inclusion, exclusion, validation, recommendation, or replacement type.

- An inclusion rule action enforces selection of certain products as specified by the product scope specification or criteria specification in the inclusion rule.
- Similarly an exclusion rule action enforces exclusion of certain products from the shopping cart.
- The validation rule displays an error message when the criteria conditions are met.
- The recommendation rule displays products based on the condition. The recommendation rules support Product, Product Family, and Product Group.
- The replacement rule helps you swap a product with another product by displaying a message with products that can be replaced. This is useful when an organization decides to discontinue a product and prompts you to go for a replacement.

  > CPQ supports replacement rules on ABO carts only for Swap and Split-Swap.

While creating a constraint rule action, when you select product scope as Product, you must enter details only in the Product field. Similarly, when you select product scope as Product Family or Product Group, you must enter details in their corresponding fields only.

> You should be careful not to put too many actions into a single constraint rule. You must follow the best practices while defining Constraint Rules. Refer to *CPQ on Salesforce Best Practices: Governor Limits*. You should test the constraint rules thoroughly before pushing them to a live environment.

For Validation type constraint rules, you can set up action criteria by defining product scope as Action Criteria. Action criteria validate the condition against the criteria you defined and executes the action intent. For example, based on your company's requirement, you cannot sell more than 3 units of Product A at a time. In this case, for condition as Product A, you can define action criteria to validate if the quantity of Product A is more than 3. If the quantity is more than 3, CPQ executes the action intent which could be an error or a warning.

When the criteria conditions are met, based on the action intent the rule action message is displayed either instantly or upon finalization. You can use the line item fields and related fields of the line item object, such as the attribute value field, product configuration fields, and product fields can also be used. When you use a constraint rule with action criteria for a product with multiple line items, the system only validates the primary line item to evaluate the constraint rule.

If the rule action is error type, the Sale rep cannot finalize the configuration until the criteria condition values have changed to make the validation ineffective.

You must run a criteria maintenance job whenever you add, remove or make any changes to criteria. Even if you remove a constraint rule that has criteria defined, you must run a criteria maintenance.

To create a Constraint Rule Action

You must have an existing constraint rule header with an existing rule condition. Select the constraint rule header. The Constraint Rule detail page appears.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click any product for which you want to define a constraint rule.
4. Click **Rules**. The page displays different rules tab.
5. Click Constraint Rule icon () next to the title, **RULES**. All the constraint rules associated with that product are displayed.
6. Search and click the constraint rule.
7. Fill in the required details under the **THEN(statements)...** section. Enter a number under **Action** column.
8. From **Action Type**, select one of the following:

Option	Description
Exclusion	Enforces exclusion of certain products from the shopping cart.
Inclusion	Enforces selection of certain products as specified by the product scope specification or criteria specification in the inclusion rule.
Recommendation	Product recommendations are displayed when the condition is met.
Replacement	Product replacements are suggested when the condition is met.
Validation	Displays the error message when the criteria conditions are met.

Action Type selection impacts the options available for **Action Intent**.

> Constraint rules of **Action Types** = *Recommendation* and *Replacement* do not support **Condition Criteria** and work only for the Product Scope.

9. From **Action Intent**, select one of the following:

Option	Description
Check on Finalization	Applies rule action on finalization of the shopping cart.
Disable Selection	Disables selection of excluded products.
Hide	Hides the selection of excluded products. If the value **Hide** is not present in the picklist, you can add the value in the picklist by following the steps below. a. Go to **Setup> Create> Objects> Constraint Rule Action> Action Intent** b. Click **New**. Type **Hide** and click **Save**. After adding the value, on the Action Intent details page find the field **Controlling Field** under the section Picklist Options and click **[Change]**. Double-click **Hide** below **Exclusion** (do not select **Hide** for any other Action Types). Click **Save**.
Prompt	Displays a message with choices of products that you can select or remove.
Show Message	Displays a message.
Auto Include	Adds selected products automatically. When the number of products selected by the rule action is more than the number mentioned in the Match Rule condition, auto-inclusion of products is not triggered. A message is displayed from the rule action instead. Note: Auto Include will not apply when the match rule value is Include All.

10. From **Product Scope**, select one of the following:

Option	Description
Action Criteria	The action criteria are defined using the flare icon ( ).
Product	In the Product field, click to select a product to be included or excluded. You search the product using product name or code.
Product Family	In the Product Family field, enter a Product Family. One or more of the products from the Product Family will be included or excluded based on rule type and match rule specifications.
Product Group	In the Product Group field, enter a Product Group. One or more of the products from the Product Group will be included or excluded based on rule type and match rule specifications.

11. According to the product scope you select, enter values in the last column.
12. Click the setting icon ( ). A pop-up is displayed. Click **STATEMENTS** tab and fill in the details as explained below.

Option	Description
Statement Name	The name is populated automatically. You can edit the name as you desire.
Match In Primary Lines	Select this option to include or exclude the products that are primary line items (offering) based on your Action Type selection.
Match In Options	Select this option to include or exclude the products that are options based on your Action Type selection.
Match in Cart Options	Select this option to include or exclude the product that are cart line items based on your Action Type selection.

Option	Description
Match in Asset	Select this option to exclude, auto-include, or prompt to include or exclude a product, product family, or product group when a product does not already exist as an installed product.
Repeat Inclusion	Indicates that every time you add a condition product the action product is auto-included. For example, adding Product A to the cart auto-includes Product B then, if you add Product A to the cart again, Product B is auto-included again. By default, action product is only auto-included once. You can select Repeat Inclusion for Client-Side Constraint Rules as well. However, only one condition statement with product scope is supported on client-side.

Option	Description
Min/Max Match Rule	Applies to inclusion and exclusion type rule action only, and that has product scope as Product Family, Product Group, or Action Criteria. Select any one of the following to complete the rule action: • *None*: No product selection is allowed. For inclusion type constraint rule: • *Include Any*: Allows selection of up to one product. Rest of the products in the prompt are disabled. • *Include All:* Allows selection of all products within the Product Group or Product Family. Note: Only this match rule works with Action Intent Auto Include. • *Include Min/Max*: Requires a minimum number of products specified in the Min Products field. For exclusion type constraint rule: • *Exclude All*: All the products are excluded. • *Exclude After One*: Allows selection of only one product. Any other selection will trigger the rule action. ⓘ When you configure a constraint rule with **Action Type** = **Exclusion** and **Min/Max Match Rule** = **Exclude After One**, if the sales representative adds the product for the first time, CPQ displays a constraint rule message on the cart in the "Client" mode. • *Exclude After Max*: Allows selection of a maximum number of products specified in the Max Products field. Any other selection over and above the specified number will trigger the rule action.

Option	Description
	When you define Min/Max for options at Option Group level, then Option Group level configuration is honored over the **Min/Max Match Rule** for prompt type inclusion constraint rule. Hence, if the maximum value that is configured at Option Group level is satisfied then the remaining options are disabled.
Sequence	Enter a number to indicate the order in which constraint rule action messages should be displayed.
Message Type	Select from *Error* and *Warning*. If you select *Error,* you cannot finalize the cart unless you satisfy the constraint rule and if you select *Warning,* you can finalize the cart but the warning message is displayed on the cart until you satisfy the constraint rule.
Message Details	Enter text to be displayed when the rule action is effective. These messages can also be translated in multiple languages. For inclusion and exclusion type rule actions product name substitution parameter {0}, {1} can be used. For example: **{0} requires {1}**. For validation type rule actions only {0} will be substituted by triggering products.

13. Click **Save**.
14. Click the flare icon ( ) to define action criteria. A pop-up is displayed.
15. Click **New Filter**.
16. Type a **Field**, an **Operator**, and type a **Value**. You can enter more than one criteria.
17. Click **Save**.

You can add multiple actions in a constraint rule. Click add icon (+) and follow the steps again.

Constraint rule actions are created which will trigger when a constraint rule condition is met.

Translating Custom Messages for Constraint Rules

Custom messages for Constraint Rules can be translated in multiple languages based on the User's org language.

These messages can be translated from the Custom Messages tab. To set up, you require the Message, Message Tag, and the Source Object ID. The Source Object ID is the Constraint Rule Action ID.

You can add as many language as you want that are supported by Salesforce.

To translate custom messages

You must have an existing constraint rule with message.

1. Click and click **Custom Messages**.
2. To create a new custom message, click **New**.
3. Type a mandatory **Custom Message Name**.
4. In **Message**, type the same message you defined in the constraint rule action. For example: If the Constraint Rule Action Message is *{1} should be suggested when customers select {0}, the Custom Message must also be {1} should be suggested when customers select {0}*.
5. Based on the language of the User's profile, type the translated message in the relevant message box. For example: For German, type the message in German language in Message_de box.
6. In **Message Tag**, type the same message as in the Message box.
7. Type or copy and paste the **Source Object Id**.
8. Click **Save**.

If the language of the user's profile is German, on the Catalog page, when the Constraint Rule is triggered, the message is displayed in German language.

Running the Constraint Rule Maintenance Job

Constraint Rule maintenance is a batch job that must be run whenever changes are made to rule conditions and when you upgrade to a new version of CPQ.

This includes:

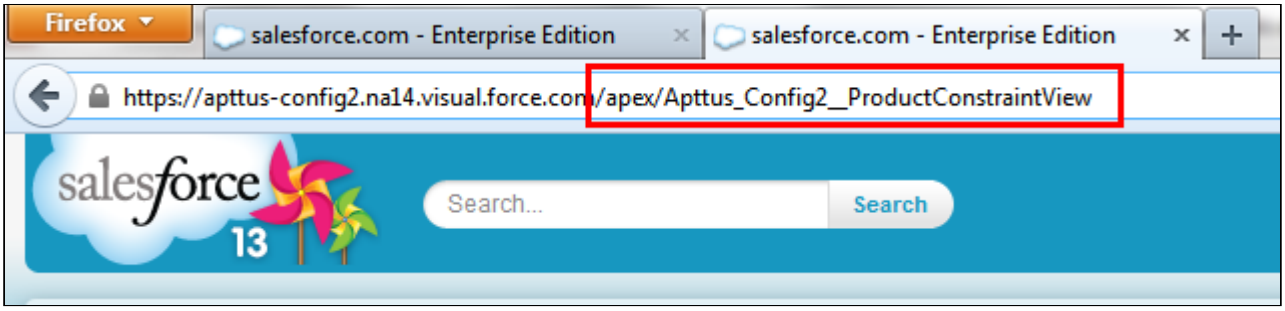
- Adding or removing a constraint rule.
- Adding or removing a constraint rule condition.
- Modifying anything in a constraint rule condition.

Users who are going to run Constraint Rule Maintenance must have read/write access to ProductConstraintView__c object . All CPQ users must have at least read access. These users should also have ProductConstraintView Visualforce page access and Apex class access enabled.

If the Constraint Rule Maintenance tab is not already available, you can add it as follows:

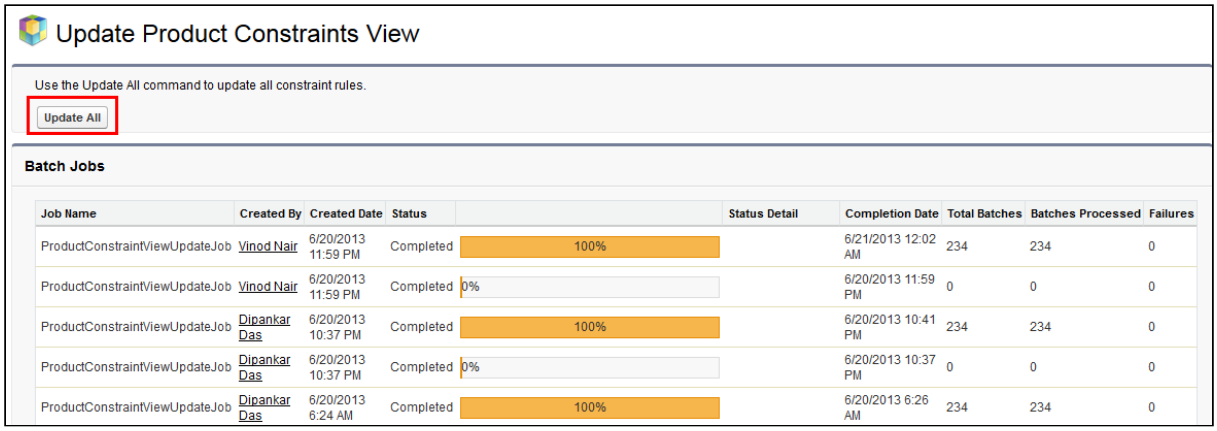
1. Go to **App Setup** > **Create** > **Apps** and click **Edit** beside **Apps Apttus Batch Updates**.
2. Add **Constraint Rule Maintenance** and click **Save**. The tab should now be available.

If the tab is still not available, as a workaround you can manually append apex/Apttus_Config2__ProductConstraint View to your URL in the address bar.



To run a constraint rule maintenance job

1. Go to the Constraint Rule Maintenance tab.
2. Click **Update All**.



The batch job is executed. The administration task is complete and an updated history for all the batch jobs is displayed. The key item to observe is Status. When Completed is displayed it means the job has run successfully, even if the percentage indicator remains at 0%.

When you run the Constraint Rule Maintenance job, CPQ inserts new rule records without affecting existing records and the quotes with existing records.

Providing Scalability in Constraint Rules

Despite you have configured large number of constraint rules for your products, CPQ executes the Constraint Rule Maintenance Job seamlessly.

You can set upto 50000 constraint rules with maximum of 10000 products and 7 conditions for each Constraint Rule.

When you have a large number of constraint rules to be processed in the maintenance, sometimes CPQ may hit governor limits. You can define a threshold to execute the Constraint Rule Maintenance Job in batches and avoid hitting governor limits. The default percentage value of the threshold, when left blank, is 100%. You can reduce the percentage based on the amount of processing your org performing. The amount of processing is based on the following factor, you can reduce the threshold when you have a number of records or a combination of them:

- Total number of SOQL queries issued
- Total number of records retrieved by SOQL queries
- Total heap size
- Maximum CPU time on the Salesforce servers

Follow the steps below:

To set a threshold for Constraint Rule Maintenance Job

1. Go to **Setup > Develop > Custom Settings > Config System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Enter a number in the **CR Maintenance Governor Limits Threshold** field. The number denotes the percentage of the governor limit.
4. Click **Save**.

Configuring Inclusion Rules

Inclusion rule use case: You have a product A and product B. You want to set up an inclusion rule such that when a user selects Product A, a warning message is displayed to include Product B.

To set up an inclusion rule

1. On the Constraint Rule Condition Detail page, in the **Criteria** section, from **Product Scope**, select *Product*.
2. In **Product**, enter *Product A* as the product and in the **Match Conditions** section, select *Match in Primary Lines* and click **Save**.
3. On the Constraint Rule Action Detail page, in **Message**, type *{0} requires {1}*. {0} denotes the condition product and {1} denotes the action product.
4. From **Action Type**, select *Inclusion* and from **Action Intent**, select *Show Message*.
5. From **Action Disposition**, select *Warning*.
6. From **Product Scope**, select *Product* and in **Product**, enter *Product B* as the product.

7. In the **Match Conditions** section, select *Match in Primary Lines* and click **Save**.

The inclusion type rule is saved and added. On the catalog page, when a user selects Product A, the system shows a warning message.

> CPQ provides support for maximum products to be added to cart using Inclusion Prompt Rule. Hence, you need to set minimum and maximum both values in order to add products and CPQ disables all remaining option products that go beyond the maximum number post selection. If you select less than the maximum value, you have to manually close the prompt as the prompt closes automatically only when the maximum number is reached.
> To enable checkboxes in the prompt, select **Show Checkbox In Rule Dialog** in Configuring Configuration Engine Settings.

Auto-Inclusion with Line Item Updates

While creating auto-inclusion constraint rules, you can specify an Action Criteria based on which products are auto-included. For every new product auto-included, line items are added to the cart. You can now update the fields of the auto-included line items, such as, quantity, term, base price, and adjustment type. You can also update the fields based on condition defined in the Update Expression field.

> **Note**
>
> Line updates are only applicable on Auto-inclusion rules with the scope as product. Add only 10 line item expressions. Please consult with Conga before adding more than 10 action expressions. There may be a more optimal way of approaching the requirement.

Scenario 1
If you purchase ten chargers or more, you can offer 2 USBs free. In this case, if the quantity of chargers in the cart is updated to >=10, then auto-include USB to the cart with the quantity set to 2, adjustment type set to % discount, and Adjustment value set to 100.

If the quantity of the chargers on the criteria line item is updated to less than 10 then auto remove Auto included USB line item from the cart.

Scenario 2

If you purchase a product with selling term greater than 2 years, you can auto-include 1 year of Premium Maintenance for free. In this case, if the Term of the product line item in the cart is updated to >=24, then auto include another line item in the cart for Premium

Maintenance with term set to 12 months, start date set to start date of the criteria line item, adjustment type set to % discount, and Adjustment value set to 100.

If the term of the product on the criteria line item is updated to less than 24 then auto remove Auto included Premium maintenance line item from the cart for each Quantity.

Scenario 3

For every 10 chargers you buy, auto-include 2 chargers. In this case, if the quantity of chargers in the cart is updated to 56, then auto include another line item in the cart for chargers with quantity set to 10 (56/10 = 5), adjustment type set to % discount, and Adjustment value set to 100.

If the quantity of the chargers on the criteria line item is updated to 60 then the quantity of the Auto included charger line item in the cart is updated to 6 from 5.

If the quantity of the chargers on the criteria line item is updated to 45 then update quantity of Auto included charger line item in the cart to 4 from 5.

If the quantity of the chargers on the criteria line item is updated from 56 to less than 10 then auto remove Auto included charger line item from the cart.

To specify line item updates to Auto-Included Products

1. Navigate to **CPQ Console**, click **Inclusion Rule**. All the Inclusion Rules are listed.
2. Click **Edit Action Expression** next the **Message** field. The Edit Action Expressions popup appears.
3. Select the field you want to update and click the **Expression** field. The expression builder appears.
4. Specify the Expression using which the Line Item Field is populated. For example, you can populate the Adjusted Price as 100 percent for an Auto-included Line Item. You can also update the Line Item field based on rollup field you have defined. Specify the roll up field name.

Adding a product that triggers an auto-inclusion rule, adds another product as a line item whose fields can be populated using the Expression Builder.

Added By and Added By Rule Info

The fields **Added By** and **Added By Rule Info** on the Line Item detail page of a product holds the information about how the product was added to the cart. The **Added By** field holds the value *Constraint Rule when* the product was added through a constraint rule or *User* when the product was added by a Sales Rep. The field **Added By Rule Info** is only populated when the product is added by auto-inclusion or prompt inclusion constraint rule. The information about the constraint rule is displayed in **Added By Rule Info** field.

Configuring Exclusion Rules

Example 1: Exclusion rule use case: You have a product A and product B. You want to set up an exclusion rule such that when a user selects Product A, an error message is displayed to exclude Product B.

To set up an exclusion rule

1. On the Constraint Rule Condition Detail page, in the **Criteria** section, from **Product Scope**, select *Product*.
2. In **Product**, enter *Product A* as the product and in the **Match Conditions** section, select *Match in Primary Lines* and click **Save**.
3. On the Constraint Rule Action Detail page, in **Message**, type *{1} is excluded on the addition of {0}*. {0} denotes the condition product and {1} denotes the action product.
4. From **Action Type**, select *Exclusion* and from **Action Intent**, select *Prompt*.
5. From **Action Disposition**, select *Error*.
6. From **Product Scope**, select *Product* and in Product, enter *Product B* as the product.
7. In the **Match Conditions** section, select *Match in Primary Lines* and click **Save**.

The exclusion type rule is saved and added. On the catalog page, when the Sales Rep selects Product A, and then selects B. The system shows an error message that Product B is excluded in the addition of Product A.

Hide excluded options at rule or product level

You can hide the disabled options at an individual product level in addition to the global level wherein it is applied on all products.
You can hide the product options that are excluded by using the Exclusion type constraint rule.
Suppose you have a product, Bundled ABC Software and Hardware with the following options:

1. 24x7 Same Day Onsite Response
2. 24x7 Support for Sev 1
3. Proactive Support Services
4. Remote Account Manager

Then you can select option 1 and disable options 2 to 4. You have the flexibility to hide disabled options at product level. This enables you to use the product configuration for the targeted product.

Example 2: Exclusion rule use case: You have Asset A, Service A. You want to set up an exclusion rule such that when a user selects Asset A on the Installed Product page, on the Service Catalog Service A must be disabled.

To set up an exclusion rule

1. On the Constraint Rule Condition Detail page, in the **Criteria** section, from **Product Scope**, select *Product*.
2. In **Product**, enter *Asset A* as the product, and in the **Match Conditions** section, select *Match in Service Assets* and click **Save**.
3. On the Constraint Rule Action Detail page, in **Message**, type *{1} is excluded on the addition of {0}*. {0} denotes the condition product and {1} denotes the action product.
4. From **Action Type**, select *Exclusion* and from **Action Intent**, select *Disable Section*.
5. From **Action Disposition**, select *Error*.
6. From **Product Scope**, select *Product,* and in Product, enter *Service A* as the product.
7. In the **Match Conditions** section, select *Match in Primary Lines* and click **Save**.

The exclusion type rule is saved and added. On the Installed page, when the Sales Rep selects Asset A, Service A is disabled on the Service Catalog.

Configuring Validation Rules

Validation rule use case: You have a product A and you want to set up a validation rule such that when a user selects Product A and should not be able to enter more than 5 quantity.

To set up a validation rule

1. On the Constraint Rule Condition Detail page, in the **Criteria** section, from **Product Scope**, select *Product*.
2. In **Product**, enter *Product A* as the product and in the **Match Conditions** section, select *Match in Primary Lines* and click **Save**.
3. On the Constraint Rule Action Detail page, in **Message**, type *{0} cannot exceed 5 quantity*. {0} denotes the condition product.
4. From **Action Type**, select *Validation* and from **Action Intent**, select *Show Message*.
5. From **Action Disposition**, select *Warning*.
6. From **Product Scope**, select *Action Criteria*.
7. In the **Match Conditions** section, select *Match in Primary Lines* and click **Save**.

8. On the Constraint Rule Detail page, scroll down to the Constraint Rule Actions related list and click **Edit Criteria**.
9. From **Field**, select *Line Item : Quantity*, from **Operator**, select *greater than*, and in **Value**, type *5*. Click **OK**.

The validation type rule is saved and added. On the catalog page, when a user selects Product A, and enters quantity as 6. The system shows an error message that Product A cannot exceed 5 quantity.

Configuring Product Option Group in Product Scope

A constraint rule supports the product option group as a scope.

While creating the constraint rule condition, you can add a Product Option Group picklist value under Product Scope and have a lookup field for the Product Option Group, which is represented by a product ID. These product IDs are different for each bundle and help trigger a constraint rule separately for each bundle. Earlier instead of product IDs, different option groups were created and shared across bundles, which in turn caused malfunctioning of the constraint rules.

Use Case: 24x7 support option must be included when all options are selected from the Additional Services option group (Option Group (Match All) to Product, Auto-include rule) Suppose you are configuring the Conga Snap Performance bundle, when you select the Telephone Support option, Email Support option, and Online Chat Support option, then you can see that the 24x7 Support option is selected by the system automatically.

Similarly, When you select all of the service options from the Additional Services Option Group. you can automatically include a support option for the following constraint rule criteria:

- Option to Option, Option Group (Match All) to Product, Auto-include
- Option to Option, Option Group (Match All) to Product, Recommendation
- Option to Option, Option Group (Match All) to Product, Validation
- Option to Option, Option Group (Match All) to Product, Replacement
- Option to Option, Option Group (Match Any) to Product, Auto-included

Additionally, when you select all of the service options from the Additional Services Option Group. you can automatically exclude a support option for the following constraint rule criteria:

- Option to Option, Option Group (Match All) to Product, Disable Selection
- Option to Option, Option Group (Match Any) to Product, Disable Selection
- Option to Option, Option Group (Match Any) to Product, Recommendation
- Option to Option, Option Group (Match Any) to Product, Replacement
- Option to Option, Option Group (Match Any) to Product, Validation

On the Server side, CPQ supports the Product Option Group as a scope for constraint and attribute rules, wherein the option groups are shared amongst the bundle products through associations.

Scenarios for Using Constraint Rules

State-based Constraint Rules

A constraint rules evaluates a state of the configuration while presenting options. For example, an inclusion rule may have an action to add option A or B to the model, but option A has already been excluded by a different rule that was associated with a different selection Hence, in this case option A is not available for selection.

Use Case:
Bundle 1 - Laptop
Processor 1TB (Option 1)
Processor 2TB (Option 2)

Bundle 2 - CISCO Switch
8 Ports (Option 1)
12 Ports (Option 2)
16 Ports (Option 3)

Cisco Server (standalone)
Office 365 (standalone) Product Group Software
Google suite(standalone) Product Group Software

Rule setup -
Rule 1 - Condition add Cisco Server. Google Suite is excluded. Setup exclusion error rule
Rule 2 - Condition add Cisco Switch, Office 365 is excluded. Setup exclusion warning rule
Rule 3 - Condition add laptop, Product Group Software is included (max 1). Setup inclusion error rule.

Scenario 1 (error) = Rule 1 and 3 are explained below:

- You add Cisco Server to cart, Google Suite is disabled and CPQ displays an error message.
- You configure a laptop, you are prompted to add a product from Product Group Software, Google Suite is disabled on prompt, and Office 365 is available to be added.

Scenario 2 (Warning) = Rule 2 is explained below:
You add Office 365 to cart, then add Cisco Switch to cart, CPQ removes Office 365 and provides a warning message.

Constraint Rules with Prompt Scenarios

On the inclusion prompt, CPQ disables or hides the products that are excluded by an exclusion type constraint rule to keep the validity of the cart intact.

When a product is disabled, CPQ shows an error message on the Cart or Catalog page. However, for the Warning type exclusion rule the product is not disabled in the prompt. When you enable **Hide Disabled Options** setting under Config System Properties, then CPQ hides the product.

Use Case:

Bundle 1 - Laptop

Processor 1TB (Option 1)

Processor 2TB (Option 2)

Bundle 2 - CISCO Switch

8 Ports (Option 1)

12 Ports (Option 2)

16 Ports (Option 3)

Cisco Server (standalone)

Office 365 (standalone) Product Group Software

Google suite(standalone) Product Group Software

Rule setup

Rule 1 - Condition add Cisco Server. Google Suite is excluded. Setup exclusion error rule

Rule 2 - Condition add Cisco Switch, Office 365 is excluded. Setup exclusion warning rule

Rule 3 - Condition add laptop, Product Group Software is included (max 1). Setup inclusion error rule.

Scenario 1 (error)

Jane adds Cisco Server to cart

Google Suite is disabled

System displays Error Message

Jane configures laptop

User is prompted to add product from Product group software

Google Suite is disabled on prompt

Office 365 is available to be added

Scenario 1 (Warning)

Jane adds Office 365 to cart

Jane adds Cisco Switch to cart

System removes Office 365 and provides Warning Message

Jane configures laptop

User is prompted to add product from Product group software

Google Suite and Office 365 is available to be added

Criteria-based Replacement Rules on the Server-side

Replacement rules on Server-side that based on the condition criteria that must be defined using fields from the following objects:

- Line Item
- Product
- Configuration
- Option
- Product Attribute Value

The condition criteria comprise a filter expression with operators and values.

> ⓘ Replacement type constraint rules are not supported on the Client-side.

Use Case for Replacement type constraint rule:

You bought 5 units of Performance Cloud -Enterprise, CPQ prompts to replace the product with Performance Cloud- Ultimate.

Rule has following constraint rule Action and Condition

Condition:

Product scope: Product

Product: Performance Cloud -Enterprise

Condition criteria: Quantity = 5

Action:

Action type: Replacement

Action Intent: Prompt

Product Scope: Product

Product: Premium Support

Disabling Constraint Rule Execution upon Cart Finalization

You can disable the execution of server-side constraint rules to optimize performance upon cart finalization. You must set the Admin Setting *APTS_DisableConstraintRulesOnFinalize* to *true* to use this feature.

> This feature is not supported with the Replacement type Constraint Rule.

To disable constraint rule execution

Perform the following steps to define the Admin Setting *APTS_DisableConstraintRulesOnFinalize*:

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record.
3. Fill in the following details

Field	Value
Name	*APTS_DisableConstraintRulesOnFinalize*
Value	*true*
Code	Leave the field blank

4. Click **Save**.

Managing Attribute-Based Configuration

Attribute-Based Configuration helps you customize the product configuration on your cart on the basis of selection of product attributes. The attributes define characteristics or the desired features for the product. The attribute configuration may or may not affect price of the product. Each attribute value selection ensures that the rest of the attributes get filtered on the basis of selected attributes and the result set is narrowed. The attributes can also leverage expressions and calculations to derive values that help identify the products.

> **Note**
>
> - Ensure that you have selected **Has Attributes** check box in your product.
> - Ensure that you have selected **Enable Field Expression** in the *Config System Properties* in Configuring Custom Settings.

With the New UI, CPQ allows you to create and configure an unlimited number of attributes, overcoming the 800 record limit of Salesforce. You can use additional attributes for Product Definitions, Product Configurations, and Attribute-based Rules.

For an example, suppose you have a product configuration behavior similar to a laptop review website where the selection of RAM and HDD restricts the Color and Screen Size, and selection of Color and Screen Size restricts any other component. The dependencies can be in any direction without having a set selection sequence. Within CPQ, you have to create a Compatibility table to define all the valid combinations for the Color, Screen Size, RAM, HDD, and Fingerprint Reader picklist values.

The attribute-based configuration provides a rules framework to drive configuration based on your Product Attributes. The rules framework allows you to set up the following dependencies or rules between the product attributes. You should perform the following steps depending on your business requirements.

1. Create Products
2. Create Product Attributes
3. Create a Product Attribute Group
4. Associate Product Attribute Group to a Product
5. Create Product Attribute Values
6. Create Attributes in Attribute Value Matrix Entry object
7. Create Attribute Value Matrices
8. Create Product Attribute Rules

Creating Product Attribute Values

You must create the attributes that you want to add in your product as custom fields in the Product Attribute Value object. You must add these attributes in the Attribute Value Matrix Entry object as well, in case you want to use Attribute Value Martix.

In our example, we have added Laptop Color, Laptop RAM, Laptop Screen Size, Laptop Fingerprint Reader, and Laptop HDD attributes to the Product Attribute Value object.

Edit | Del | Replace	Laptop Color	Laptop_Color__c	Picklist	10/5/2015 10:56 PM
Edit | Del | Replace	Laptop Fingerprint Reader	Laptop_Fingerprint_Reader__c	Picklist	10/5/2015 11:00 PM
Edit | Del | Replace	Laptop HDD	Laptop_HDD__c	Picklist	10/5/2015 10:58 PM
Edit | Del | Replace	Laptop RAM	Laptop_RAM__c	Picklist	10/5/2015 10:58 PM
Edit | Del | Replace	Laptop Screen Size	Laptop_Screen_Size__c	Picklist	10/5/2015 10:59 PM

To create product attribute value

1. Go to **Setup > App Setup > Create > Objects** and select **Product Attribute Value**.

   > If you have more than 800 attributes already create Product Attribute Value object, use Product Attribute Value Extension objects. For more details, refer to Creating Custom Product Attribute Value Extensions.

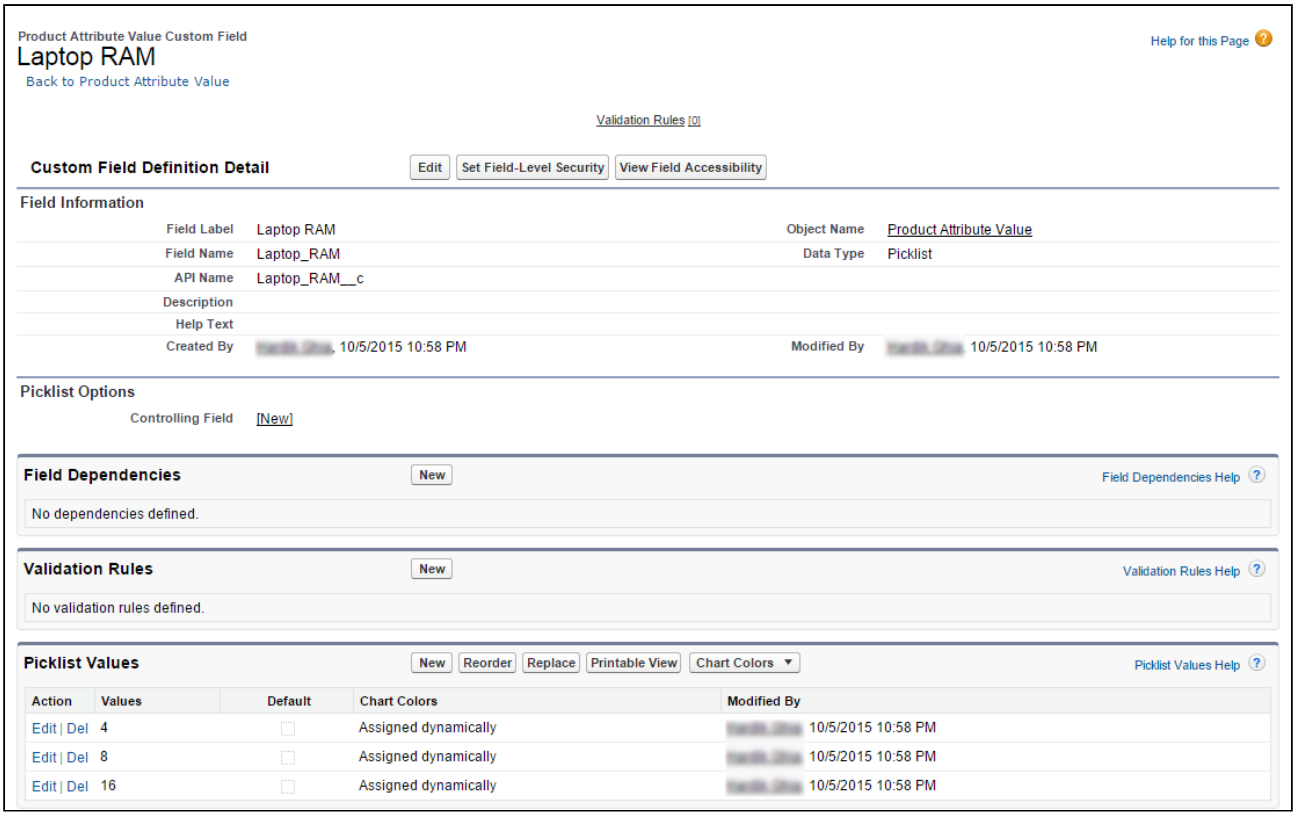
2. Under Custom Fields & Relationships section, click **New**.
3. Select **Picklist** from the Field Type.
4. Click **Next**.
5. Enter **Field Label**.

   > The attribute label must be unique across all the Product Attribute Value and Product Attribute Value Extension objects. Otherwise, CPQ does not provide correct results when executing constraint rules, product attribute rules, attribute value matrices, price rules, and price Matrices that are defined using attributes with duplicate names.

6. Enter values for picklist.
7. Click **Next**.
8. Establish field level security and click **Next**.
9. Select appropriate page layouts where you want to add this field.
10. Click **Save.**

Let's create Laptop RAM attribute in the Product Attribute Value object.

1. Go to **Setup > App Setup > Create > Objects** and select **Product Attribute Value**.
2. Under Custom Fields & Relationships section, click **New**.
3. Select **Picklist** from the Field Type.
4. Click **Next**.
5. Enter Laptop RAM as **Field Label**.
6. Enter 4, 8, 16 (separated by a line) as values for picklist.
7. Click **Next**.
8. Establish field level security and click **Next**.
9. Select appropriate page layouts where you want to add this field.
10. Click **Save.**



Similarly, create Laptop Color, Laptop HDD, Laptop Screen Size, and Laptop Fingerprint Reader attributes in the Product Attribute Value object.

You have created attributes in the Product Attribute Value object.

About Product Attribute Value Extensions

CPQ provides you the ability to create and configure an unlimited number of attributes, overcoming the 800 record limit of Salesforce. The Product Attribute Extension objects are a replica of the Product Attribute Value object. These objects are an extension to the Product Attribute Value object through a combination of master-detail and look-up relationships. The fields from the extension objects are displayed in the Product Console to be associated as attributes with the products. All of the attributes created under the extension object can be used as normal attributes for your products. For example, you can use the extension objects in Attribute Value Matrices, Product Attribute Rules, Constraint Rules, Numeric Expressions, Document Generation, Asset-Based Ordering, and Asset pricing. These attributes can be used when Sales Rep renews, change, and amends an asset.

The following extension objects that are available to you in addition to the regular Product Attribute Value object:

- Product Attribute Value EXT
- Product Attribute Value EXT 2
- Product Attribute Value EXT 3

You must use the **Product Attribute Value** and Product Attribute Value Extension objects sequentially. Each product attribute value object has a limit of 800 records per object and you must use the next object only after the limit of the object is reached. For example, you must completely use the **Product Attribute Value** object before utilizing the **Product Attribute Value Ext** object. And you must only use **Product Attribute Value EXT 2** after the **Product Attribute Value Ext** object is completely used.

Similar to the **Product Attribute Value** object, Agreement, Asset, Order, and Proposal Attribute Value objects also have respective extensions. CPQ uses the Agreement, Asset, Order, and Proposal Attribute Value object and the respective extension objects to store the values of the attributes associated with the agreement, asset, order, and proposal line items. Along with Product Attribute Value objects, you must create the same attribute fields in Agreement, Asset, Order, and Proposal Attribute Value objects as well, to retain the attribute values in agreements, assets, order, and proposal line items from configuration line items.

If you define attributes in attribute value extension objects, you must enable the **Populate Attribute Extension** custom setting. The setting allows CPQ to retain the attribute values from configuration line items to the agreement, asset, order, and proposal line items upon creation.

To enable Populate Attribute Extension

You must enable **Populate Attribute Extension** checkbox to indicate that the attribute value extension objects contain attribute fields.

1. Go to **Setup > Develop > Custom Settings > Config System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Enable **Populate Attribute Extensions**.
4. Click **Save**.

> ⓘ
> - This enhancement is available in the CPQ Admin UI only.
> - The attributes added in the Product Attribute Value Extension object are not available for Price Dimensions (Pricing engine). Only the first 800 attributes will be visible for the pricing.

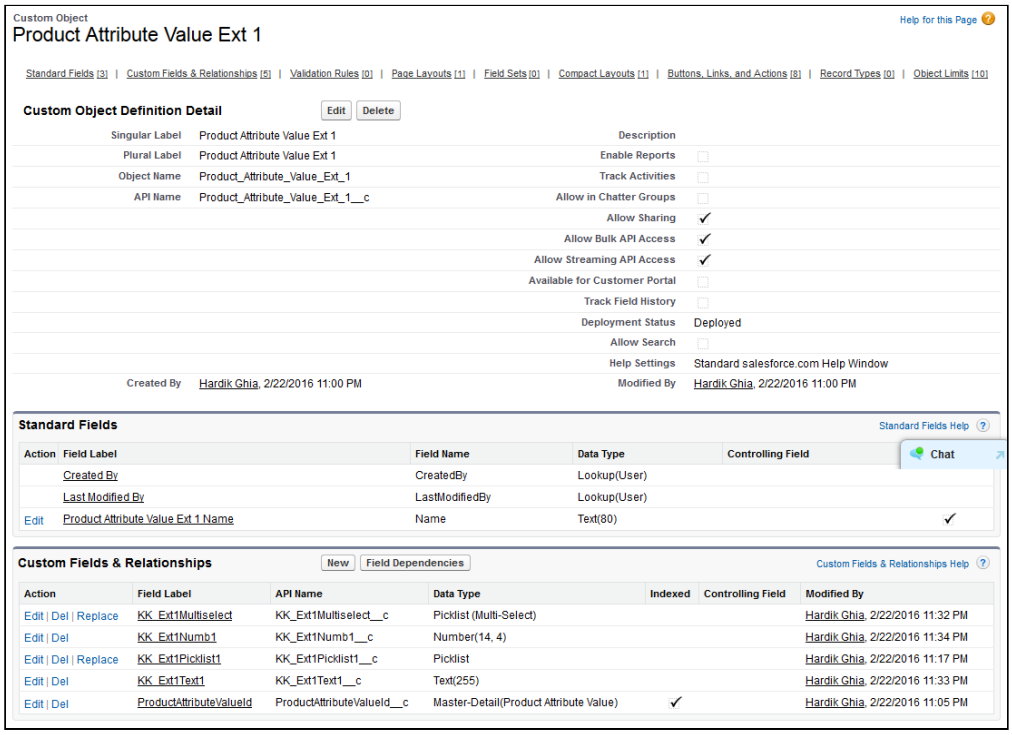
You can also create custom Product Attribute Extension objects. Follow the steps below.

Creating Custom Product Attribute Value Extensions

You can also create custom Product Attribute Extension objects. Follow the steps below.

To create a custom extension object

1. Navigate to **Setup > App Setup > Create > Objects** and click **New Custom Object** to create a new extension object.
2. Enter the desired label, such as *Product Attribute Value Ext 1* to indicate that this object is an extension to Product Attribute Value object.
3. Fill in the requisite details and click **Save**.
4. Navigate to the Custom Fields & Relationships section and click **New** to create a new custom relationship.
5. Choose *Master-Detail Relationship* as the **Field Type** and click **Next**.
6. From the **Related To** drop-down list, choose *Product Attribute Value* and click **Next**.
7. In the **Field Label**, type *ProductAttributeValueId* and click **Next**.
8. Choose necessary **Field level security** for the reference field and click **Next**.
9. Choose the page layouts where you want to reference this field and click **Next**.
10. Verify the **Related List label** for your field and click **Save**.
    Ensure that the API name for the newly created custom relationship is *ProductAttributeValueId__c*, as shown in the figure below.



After you have created the extension object, follow the same steps mentioned here to create the attributes for this object in the form of custom fields.

To create a lookup relationship with the custom Product Attribute Value extension object

For each extension object, a lookup relationship must be created so that the objects are connected.

1. Navigate to **Setup > App Setup > Create > Objects** and search for *Product Attribute Value* object.
2. Under the Custom Fields & Relationships section and click **New** to create a new custom relationship.
3. Choose *Lookup Relationship* as the **Field Type** and click **Next**.
4. From the **Related To** drop-down list, choose the object you created in the above procedure. In our example, choose *Product Attribute Value Ext 1* and click **Next**.
5. In the **Field Label**, type *ProductAttributeValueId*. In the **Field Name**, type *ProductAttributeValueId1* and click **Next**.
6. Choose necessary **Field level security** for the reference field and click **Next**.
7. Choose the page layouts where you want to reference this field and click **Next**.
8. Verify the **Related List** label for your field and click **Save**.
   Ensure that the API name for the newly created custom relationship is *ProductAttributeValueId1__c*.

Field Information

Field Label	ProductAttributeValueId	Object Name	Product Attribute Value
Field Name	ProductAttributeValueId1	Data Type	Lookup
API Name	ProductAttributeValueId1__c		
Description			
Help Text			
Created By	Hardik Ghia, 2/22/2016 11:11 PM	Modified By	Hardik Ghia, 2/22/2016 11:11 PM

Lookup Options

Related To	Product Attribute Value Ext 1	Child Relationship Name	Product_Attribute_Values
Related List Label	Product Attribute Values		
Required	☐		
What to do if the lookup record is deleted?	Clear the value of this field.		

To configure Custom Settings for the custom extension object

For each lookup relationship, you create with the custom extension object, you must make an entry in a custom setting.

> ⓘ You cannot use custom extension objects and standard CPQ extension objects simultaneously. You must only add entries in the **Product Attribute Extension Tables** custom setting if you want to use custom extension objects. Otherwise, leave the field blank.

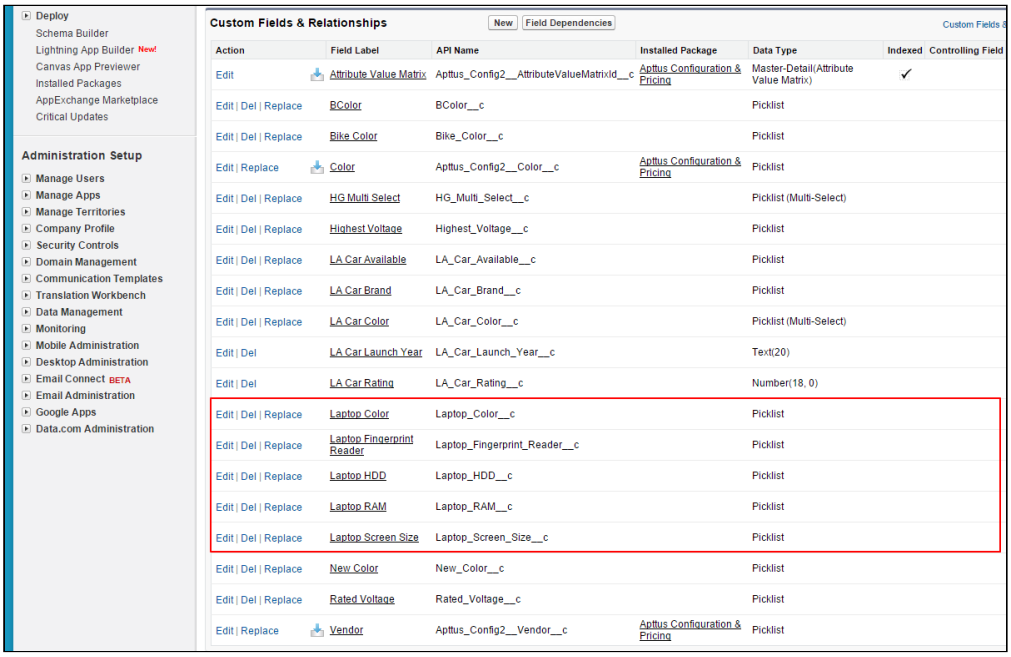
1. Navigate to **Setup > App Setup > Develop > Custom Settings** and click **Manage** next to **Config System Properties**.
2. Next to **System Properties**, click **Edit**.
3. In the **Product Attribute Extension Tables** field, type the API name of the lookup field you have created in the above procedure. In our example, type *ProductAttributeValueId1__c*.
   Ensure that you have specified the API names of all your lookup relationships here, separated by a comma. For example, if you have 4 extension objects and have their corresponding 4 lookup relationships, you must specify 4 API names in this field.
4. Click **Save**.

You can create attributes in Product Attributes Value Extension object the same way you create attributes in the Product Attribute Value object, just select the extension object that you created in the above-mentioned steps. Refer to Creating Product Attribute Values.

Creating Attributes in the Attribute Value Matrix Entry Object

The Attribute Value Matrix Entry object must comprise attributes that are the same as the entries in Product Attribute Value with the same API Name. The attributes that you add in the Attribute Value Matrix Entry object are displayed on the Attributes page on the cart.

In our example, we have added Laptop Color, Laptop RAM, Laptop Screen Size, Laptop Fingerprint Reader, and Laptop HDD attributes to the Product Attribute Value object. Hence, we must add these attributes in the Attribute Value Matrix Entry object as well.



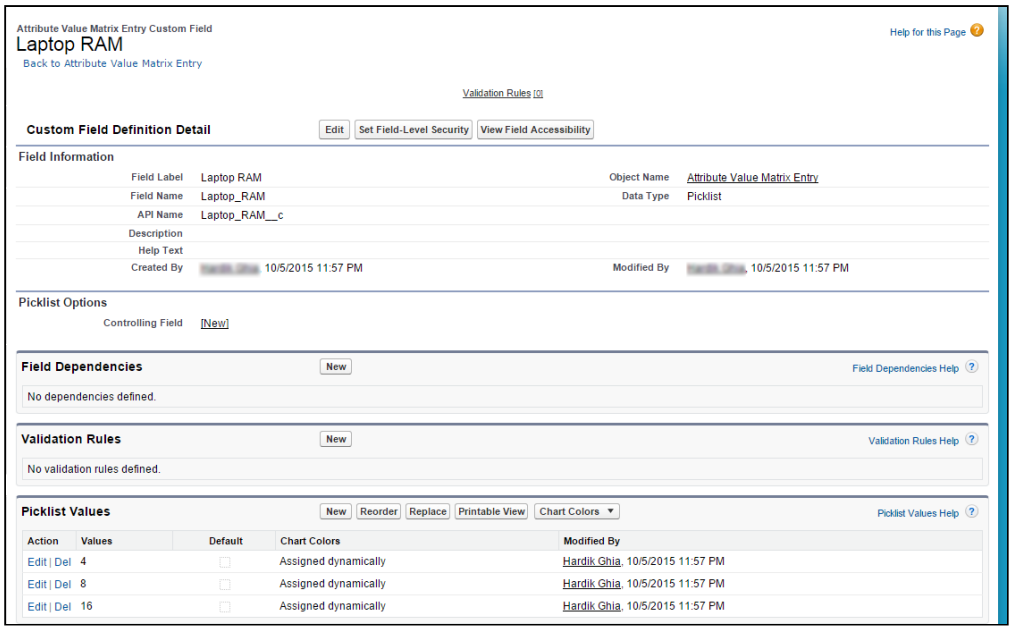
To create an attribute in the Attribute Value Matrix Entry object

1. Go to **Setup > App Setup > Create > Objects** and select **Attribute Value Matrix Entry**.
2. Under Custom Fields & Relationships section, click **New**.
3. Select **Picklist** from the Field Type.
4. Click **Next**.
5. Enter **Field Label**.
6. Enter values for picklist.
7. Click **Next**.
8. Establish field level security and click **Next**.
9. Select appropriate page layouts where you want to add this field.

10. Click **Save.**

Let's create Laptop RAM attribute in attribute value matrix entry object.

1. Go to **Setup > App Setup > Create > Objects** and select **Attribute Value Matrix Entry**.
2. Under Custom Fields & Relationships section, click **New**.
3. Select **Picklist** from the Field Type.
4. Click **Next**.
5. Enter Laptop RAM as **Field Label**.
6. Enter 4, 8, 16 (separated by a line) as values for picklist.
7. Click **Next**.
8. Establish field level security and click **Next**.
9. Select appropriate page layouts where you want to add this field.
10. Click **Save.**



Similarly, create Laptop Color, Laptop HDD, Laptop Screen Size, and Laptop Fingerprint Reader attributes in the Attribute Value Matrix Entry object.

You created attributes in the Attribute Value Matrix Entry object.

Now, you should create the Attribute Value Matrix.

Creating Attribute Value Matrices

Attribute Value Matrices are used to associate attributes to the product. You can use the attribute value matrices to drive the product selection. You can associate a maximum of

five attributes with each other. The attributes defined in the Attribute Value Matrices are bi-directional. Once you create a matrix, a compatibility table is created. This compatibility table for Attribute Values helps you define all the valid combinations for the attribute values.

Use Case

For example, Amttus, a leading laptop manufacturer, wants to use attributes to drive the product selection of their users. A laptop has five attributes: Color, HDD, RAM, Screen Size, and Fingerprint Reader. The following table describes the products and its attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB
RAM	8	16	4	8	8	16
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

After you create an attribute value matrix, when the user selects Black from the Color picklist menu, the HDD picklist menu will contain only 1 TB and 2 TB as options. RAM picklist menu will contain 8 and 16 as options. Screen Size will contain 14 and 14 HD as options. The Fingerprint Reader picklist menu will contain Yes and No as options.

To Create an Attribute Value Matrix

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click any product, preferably the one you want to associate Attribute Value Matrix with.
4. Click **Rules**. The page displays different rules tab.
5. Click Attribute Value Matrix icon( ) next to the title, **RULES**. Attribute Value Matric tab is displayed
6. Click the menu icon( ) next to **New Constraint Rule** and click **New Attribute Value Matrix**.

7. Fill in the required details.

Field Name	Description
Matrix Name	Enter name of the Attribute Value Matrix.
Active	Flags whether the Attribute Value Matrix is active.
Effective Date	Select the date from which the Attribute Value Matrix is in effect.
Expiration Date	Select the date until which the attribute value matrix is valid.
Description	Describe the attribute value matrix.
Application Type	A picklist field, which contains *Default*, *Constraint* and *Force Set* values. *Default*: Selecting this lets you choose the picklist values which you have configured as default values in the Matrix View. The rest of the values are visible in the drop-down list, but you cannot select any of them on the Cart. *Constraint*: Selecting this value lets you choose only those picklist values which you have configured in the Matrix View. All the other values are disabled in the drop-down list on the cart. This also works with the Multi-select picklist. *Force Set:* Selecting this value lets you automatically set the picklist values which you have configured in the Matrix View (considering the last user selection). As soon as you change the attribute value for one attribute, the system sets the attribute values for other attributes immediately. If there is a Matrix entry with null values and **Treat Null As Wildcard** is set to false, the system resets the attribute values to null.
Treat Null As Wildcard	This field works in conjunction with **Application Type**. If you select this check box, all the picklist values (default or constrained) configured in the Matrix View are available for selection on the Cart. Deselecting this check box disables all the picklist values on the Cart.

Field Name	Description
Matrix Fields	Click the plus icon to add attributes to the attribute value matrix. If you don't select any attribute then, **THEN(statements)...** section remains blank.

8. Select the attribute value matrix you just created from the **RULES** section.
9. Fill in the required details under the **IF(scope)...** section.

Field Name	Description
Product Family	A multi-select field. Click in the text box to view the available product families and select the appropriate ones. You can also enter a keyword and search for a Product Family.
Product Group	A multi-select field. Click in the text box to view the available product groups and select the appropriate ones. You can also enter a keyword and search for a product group. The product groups are filtered on the basis of product families that you select.
Products	A multi-select field. Click in the text box to view the available products and select the appropriate ones. You can also enter a keyword and search for a product. The products are filtered on the basis of product families and product groups that you select.
Account Location	A multi-select field. Click in the text box to view the available account locations and select the appropriate ones. You can also enter a keyword and search for a location. The locations are filtered on the basis of product families, product groups, and products that you select.

10. Define the desired matrix under the **THEN(statements)...** section. Click Add Row to add more rows to the matrix. Click **Add Column** to add more attribute in the matrix. You can add or remove the row using the action menu icon(⋮).
11. Create multiple valid combinations from the picklist menus. You can select the appropriate attributes from the picklist menu.
    Note that these attribute picklists work in a bi-directional manner. For example, if you choose **LaptopColor** = Black, **LaptopHDD** defaults to *1TB.* Similarly, if you select **Laptop HDD** as *1TB*, the system defaults **Laptop Color** to *Black*.
12. Click **Save**.

If you create or edit the criteria, you must run **Criteria Maintenance > Update Expression Fields**.

Creating Product Attribute Rules

Product Attribute Rules allow product configuration according to product attributes that are based on the actions you select. Each attribute value selection should ensure that the rest of the attributes get constrained to allow only valid selections to help narrow the result set. The attributes can also leverage expressions and calculations to derive values that help identify the product variant. This feature simplifies the configuration process, eliminates the need for component-level choices and provides guided selling at the product configuration level. You can also set multiple actions for the same product attribute rule.

The product attribute rule is applied to the criteria that you define in the Criteria section. You can define simple and advanced criteria. In simple criteria, you can select field, operator and value. In advanced criteria, you can set a condition, if it is true, then the actions in the Action section are triggered.

If you have set a default value for a field from Salesforce and you define a product attribute rule with an action of setting the default value, then the value mentioned in the Salesforce field is set. In short, you cannot override the default value that is set in the Salesforce field by using a product attribute rule.

> **Note**
>
> Attributes are constrained on the basis of selection of other attributes. The constrained values are not displayed in the drop-down menu. In order to reset the list, click  or select -None- from the picklist.

For example, Amttus, a leading laptop manufacturer, wants to drive the product selection of their users by using attributes. A laptop has five attributes - Color, HDD, RAM, Screen Size, and Fingerprint Reader. The following table describes the products and its attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB
RAM	8	16	4	8	8	16

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

Now, we want to set the default value of Fingerprint Reader as yes, if the user selects White from Color picklist menu and 500 MB from HDD picklist menu.

To Create a Product Attribute Rule

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Products** menu, click **Manage Products**.
3. Search and click any product, preferably the one you want to associate Product Attribute Rule with.
4. Click **Rules**. The page displays different rules tab.
5. Click Product Attribute Rule icon( ) next to the title, **RULES**. Product Attribute Rule tab is displayed
6. Click the menu icon( ) next to **New Constraint Rule** and click **New Product Attribute Rule**.
7. Fill in the required details.

Field Name	Description
Rule Name	Enter the Name of Product Attribute Rule.
Active	Flags whether the product attribute rule is active.
Effective Date	Select the date when the product attribute rule is in effect. Leave this field blank if you want this product attribute rule to be always effective.
Expiration Date	Select the date when the product attribute rule will become invalid. Leave this field blank if you want this product attribute rule to be always effective.

8. Select the product attribute rule you just created from the **RULES** section.
9. Fill in the required details under the **IF(scope)...** section. You can leave them blank if the fields are not required.

Field Name	Description
Product Family	A multi-select field. Click in the text box to view the available product families and select the appropriate ones. You can also enter a keyword and search for a product family.
Product Group	A multi-select field. Click in the text box to view the available product groups and select the appropriate ones. You can also enter a keyword and search for a product group. The product groups are filtered on the basis of product families that you select.
Products	A multi-select field. Click in the text box to view the available products and select the appropriate ones. You can also enter a keyword and search for a product. The products are filtered on the basis of product families and product groups that you select.
Criteria	Enter the formula to be evaluated. When this formula evaluates to true, the rule is executed.

ⓘ The Scope value for **Product**, **Product Family**, and **Product Group** can be left blank instead of the default value *All*. Leaving the field blank prevents the rule from searching across all records of that type, improving the scalability performance of product attribute rules. For example, the Product scope can be specific for product value, while leaving **Product Family** and **Product Group** blank, so that they are not queried because they are not needed.

10. Fill in the fields under the **THEN(statements)...** to define the Product Attribute Rule.

Field Name	Description
Target Attribute	Select the target attribute. Use type ahead to display a list of applicable fields or attributes.

Field Name	Description
Action Type	Select the action that you want to perform on the product attribute. Available options are: • Allow - To restrict the visibility of values on the target field. • Default - To add a default value on the target attribute. • Hidden - To hide the target attribute. • Disabled - To make the target attribute as a read-only field. • Required - To make the target attribute as a required field. • Reset - To auto-populate a default value on the target attribute if the field is left blank. ⓘ Click the relevant link to see the individual examples.
Value Expression	Enter the Value Expression. You can use the methods such as JOIN and DISCRETEINTERVAL. **DISCRETEINTERVAL(start, stop, interval)**: This method returns a discrete array of values from start to stop separated by interval. Params: • start - the first element of the specified dimension in the array • stop - the last element of the specified dimension in the array • interval - the distance between two elements in the array **JOIN(array, separator):** This method returns a concatenated text string of all the elements of an array, using the specified separator between each element. Params: • array - an array that contains the elements to concatenate. • separator - the string to use as a delimiter between values.Separatoris included in the returned string only if the value has more than one element. If no separator is specified, the default will be a comma (,). For example, if you want to limit the deductible amount (an attribute) for California state from 200 to 1000, with an interval of 100, use the expression as:*JOIN(DISCRETEINTERVAL(200,1000,100),';')*

- Click **Save**.

> Once you save your record, update the **Target Attribute** with the prefix *Apttus_Config2__ProductAttributeValue__c*. For example, if your **Target Attribute** is *ProductAttributeValueId1__r.Ext1Multiselect__c*, after the update, your **Target Attribute** should be *Apttus_Config2__ProductAttributeValue__c.ProductAttributeValueId1__r.Ext1Multiselect__c*. This up-gradation is required to work with PAV Extension Objects.

Allowing Values Conditionally for a Picklist or Multi-Select Picklist Attribute

Attribute-Based Configuration allows you to create the dependency for the target attribute. You can configure if you want to allow the attribute values based on a condition. The constrained value should be derived from a value expression, for example, can be pulled from a product field. This helps in providing only a relevant set of attributes for the user to make selections, thus providing a guided experience for configuration.

The following table displays the data types on which you can perform this action and the value of Value Expression required.

Action	Data Types supported	Value Expression
Allow	Picklist Multi-picklist	The values to be allowed. 'Red;Blue;Black;Orange' For example, '2014;2012;2015'

For example, Amttus, a leading laptop manufacturer, wants to use attributes to drive the product selection of their users. A laptop has five attributes - Color, HDD, RAM, Screen Size, and Fingerprint Reader. The following table describes the products and its attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
RAM	8	16	4	8	8	16
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

The following table describes scenarios where you can use this feature.

Scenario	Target Field	Value Expression
When the user selects **White** color, you want to allow **Yes** as a Fingerprint reader attribute.	Laptop Fingerprint Reader	'Yes'

Configuring Default Values Conditionally for Attributes

Attribute-Based Configuration allows you to create the dependency for the target attribute if that attribute should conditionally have a default value set. This feature helps provide starting point configurations for the product and reduce the number of clicks for the user.

> **Note**
>
> If you have set a default value of an attribute using numeric expression and you set the default value of the same attribute using Product Attribute Rule, then the attribute will populate the default value that is set by numeric expression. That is, the value mentioned in numeric expression has priority.
> Similarly, if you have set a default value of an attribute from Salesforce field and you set the default value of the same attribute using the Product Attribute Rule, then the attribute will populate the default value that is set using the Salesforce field. That is, the value mentioned in Salesforce field has priority.

The default values are set only when the attributes have no value. For example, you have selected *None* as picklist value. If you have set a value manually in your attribute, then the default value does not override it.

Note

When you are setting the default value for the target field, the following points must be kept in mind.

- The default value for number Data Type must be written as mentioned in the field value.
- The default value for text and picklist Data Type must be written between single quotes.
- The default value for multi-picklist Data Type must be written between single quotes and must be separated by a semicolon without any space.

The following table displays the datatypes on which you can perform this action and the value of Value Expression required.

Action	Datatypes supported	Value Expression	Example
Default	Number Text Picklist Multi-picklist	The default value. For example, 'RED'	Number: 79 Text: 'Red' Picklist: 'Blue' Multi-picklist: 'Red';'Black'

For example, Amttus, a leading laptop manufacturer, wants to use the attributes to drive the product selection of their users. A laptop has five attributes: Color, HDD, RAM, Screen Size, and, Fingerprint Reader. The following table describes the products and its attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB
RAM	8	16	4	8	8	16
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

The following table describes scenarios where you can use this feature.

Scenario	Target Attribute	Value Expression
When the user selects 2 TB from Laptop HDD picklist, Laptop RAM must set 16 as the default value.	Laptop RAM	16
When the user selects Black from Color, 1 TB from HDD, 8 from RAM, and 14 from Screen Size, Laptop Fingerprint Reader must set No as the default value.	Laptop Fingerprint Reader	'No'

To disallow the value to be changed by a user, create a read-only attribute or make it conditionally read-only. For more information on conditionally making an attribute read-only, refer to Conditionally Make an Attribute as Read-Only.

Hiding Attributes Conditionally

Attribute-Based Configuration allows you to create the dependency for the target attribute to either make the attribute conditionally visible or hide the attribute based on other selections. This helps to provide only relevant sets of attributes for the user to make selections, thus providing a guided experience for configuration.

The following table displays the data types on which you can perform this action and the value of Value Expression required.

Action	Datatypes supported	Value Expression
Make Hidden	Number Text Picklist Multi-picklist	

For example, Amttus, a leading laptop manufacturer, wants to use the attributes to drive the product selection of their users. A laptop has five attributes: Color, HDD, RAM, Screen Size, and Fingerprint Reader. The following table describes the products and attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB
RAM	8	16	4	8	8	16
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

The following table describes scenarios where you can use this feature.

Scenario	Target Attribute	Value Expression
When the user selects 4 from RAM, you want to hide the Screen Size attribute.	Screen Size	
When the user selects Black from Color and 1 TB from HDD, you want to hide the RAM attribute.	RAM	

Configuring Attributes Conditionally as Disabled

Attribute-Based Configuration allows you to create the dependency for the target attribute if that attribute should conditionally be marked as *Disabled*. This feature prevents the the user from changing a value of an attribute.

The following table displays the data types on which you can perform this action and the value of Value Expression required.

Action	Data Types supported	Value Expression
Disabled	Number Text Picklist Multi-picklist	

For example, Amttus, a leading laptop manufacturer, wants to use attributes to drive the product selection of their users. A laptop has five attributes: Color, HDD, RAM, Screen Size, and Fingerprint Reader. The following table describes the products and its attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB
RAM	8	16	4	8	8	16
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

The following table describes scenarios where you can use this feature.

Scenario	Target Attribute	Value Expression
When the user selects Black from Color, 1 TB from HDD, 8 from RAM, and 14 HD from Screen Size, the Fingerprint Reader must be disabled.	Laptop Fingerprint Reader	

Configuring Attributes Conditionally as Required

Attribute-Based Configuration allows you to create the dependency for the target attribute if an attribute must be made conditionally mandatory while selection. This helps to prevent the user from missing any important attribute that must be entered.

The system will auto-select the remaining value if only one value is remaining in a **Required** attribute after the constraints are fired through a matrix or a product attribute rule.

The following table displays the data types on which you can perform this action and the value of Value Expression required.

Action	Datatypes supported	Value Expression
Make Required	Number Text Picklist Multi-picklist	

For example, Amttus, a leading laptop manufacturer, wants to use the attributes to drive the product selection of their users. A laptop has five attributes: Color, HDD, RAM, Screen Size, and Fingerprint Reader. The following table describes the products and its attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB
RAM	8	16	4	8	8	16
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

The following table describes scenarios where you can use this feature.

Scenario	Target Attribute	Value Expression
You want to ensure that the user selects Screen Size.	Laptop Screen Size	

Resetting Attributes Conditionally

Attribute-Based Configuration allows you to create the dependency for the target attribute to conditionally reset an attribute based on other selections. This helps in ensuring a default value to be added to the target attribute if the user leaves the attribute blank or null. If the user does not select any value in the attribute, the default value be selected by default.

The following table displays the datatypes on which you can perform this action and the value of Value Expression required.

Action	Datatypes supported	Value Expression
Reset	Number Text Picklist Multi-picklist	The default value. For example, 'White' .

For example, Amttus, a leading laptop manufacturer, wants to use the attributes to drive the product selection of their users. A laptop has five attributes: Color, HDD, RAM, Screen Size, and Fingerprint Reader. The following table describes the products and its attributes.

Product	A-560	A-560m	A-440	A-440m	A-320	A-320m
Color	Black	Black	White	White	Black	White
HDD	1 TB	2 TB	500 MB	500 MB	1 TB	1 TB
RAM	8	16	4	8	8	16
Screen Size	14 HD	14 HD	15	15 HD	14	14 HD
Fingerprint Reader	Yes	Yes	Yes	Yes	No	Yes

The following table describes scenarios where you can use this feature.

Scenario	Target Attribute	Value Expression
When the user selects Black from Color, 16 from RAM, 14 HD from Screen Size, and Yes from Fingerprint Reader, you want to reset the value of HDD to 2 TB.	Laptop HDD	'2 TB'

Scenario to Use Product Attribute Rules with Criteria

Product Attribute rules (PAR) effective date based on quote dates

When the effective date of the quote on the product configuration page fulfills the PAR criteria that you defined, a Rule gets executed.

For example,

Rule Criteria: Apply rules where,
Rule Start Date <= Quote Expected Start Date AND Rule End Date >= Quote Expected End Date Date

PAR Effective Date	PAR Expiration Date	Effective Date (On Product Config)	Result
01-11-2017	30-11-2017	15-11-2017	Rule is getting applied
01-11-2017	30-11-2017	31-10-2017	Rule is not getting applied

Managing Pricing

Pricing enables you to rapidly set up pricing structures. Pricing comprises of price lists and price list items. A price list controls which products are visible to the end user. A price list contains several price list items; each linked to a product. You can associate price lists to a product and create price list items for products at the product level as well. You cannot create a price list at the product level.

> Most of the pricing features are available **Admin UI**. However, some features are currently not available in **Admin UI**. In such cases, you must use **CPQ Console**.

- Managing Price Lists and Price List Items
- Managing Price Rules
- Creating Price Matrices and Dimensions
- Managing CPQ Formula Fields
- Configuring Related Pricing
- Enabling Price Ramps
- Enabling Auto Ramp Creation
- Enabling Tiered Pricing
- Running a Criteria Maintenance Job
- Configuring Contract Pricing
- Configuring Pricing Profiles and Pricing Batch Size
- Configuring Email Notifications for Apex Governor Limit Warning
- Managing Cost and Profitability (Cost Breakdown)
- Configuring Unit of Measure (UOM) and Frequency Conversion Rate
- Configuring the Currency Rounding Mode
- Managing Currency Conversion
- Managing Price Waterfall

Managing Price Lists and Price List Items

Pricing enables you to rapidly set up pricing structures. Pricing comprises of price lists and price list items. A price list controls which products are visible to the end user. A price list contains several price list items; each linked to a product.

You can simply select options and have them priced completely accurately reflecting the latest price lists and discounting rules. You can price any products or bundles based on features or options selected. You can set up pricing rules, constraints, dependencies and extraneous variables.

In the following sections:

- Creating Price Lists
- Associate a Price List with Categories
- Viewing and Publishing Favorite Configurations for a Price List
- Creating Price List Items
- Configuring Price Escalators

Creating Price Lists

Price lists are containers of price list items. Price lists can also have a currency set if your environment is multi-currency enabled.

Price lists are associated with categories so that an end user can view and select products from the shopping cart. For example, Price list A is associated with Category B, when you select price list A on the Quote/Proposal Edit page, only category B and associated products are displayed in the shopping cart.

You can create a price list at any time; however, it is most efficient to create a price list before creating products to easily associate the products with the price list.

To create a price list

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**.
3. Click **New PriceList**.
4. Enter the following details and click **Save**.

Field	Description
PriceList Name	Enter the name of the price list. This is a mandatory field.
Short Description	Enter a description for the price list.
Is Active	Move the slider to the right to enable the price list.
Effective Date	Select a date from when the price list must be effective.
Expiration Date	Select a date in the calendar on which the price list should be expired or made inactive.

Field	Description
Based on Price List	Select the name of an existing price list depending on which the new price list is created. This field is useful in scenarios such as your organization wants to sell the same products in different currencies or in the same currency with some discount. For example, your organization sells products in the United States in the USD currency. You have created a price list called USD Price List. For all products, you have created the required price list items, which are associated with USD Price List. Now you want to sell the same products in Great Britain in the GBP currency. You can create another price list with the following details: • **PriceList Name**: GBP Price List • **Currency**: GBP - British Pound • **Based On Price List**: USD Price List If you sell a product that costs 1 USD in the Unites States, the price of it will be converted to GBP based on the currency conversion rate defined at **Setup** > **Company Profile** > **Manage Currencies**. In this way, you can create a price list based on another price list, without creating corresponding price list items.

Field	Description
Based on Adjustment Type	Select the type of adjustment. The following adjustment types are available: • % Discount: If the Sales Rep applies a % discount type of adjustment, the List Price is discounted by a percentage specified in the adjustment amount field. The new amount is stored on the cart line item. • Discount Amount: If the Sales Rep applies a discount amount of adjustment, the List Price is discounted by a percentage specified in the adjustment amount field. The new amount is stored on the cart line item. • % Markup: If the Sales Rep applies a percentage markup of adjustment, the List Price is increased by the percentage specified in the % Markup field. The new amount is stored on the cart line item. For example, if you want to increase the price of laptops during the month of September, you can apply a markup amount or a percentage markup that adds to the List price. • Markup Amount: If the Sales Rep applies a markup amount of adjustment, the List Price is increased by the amount specified in the markup amount adjustment field. The new amount is stored on the cart line item. • Price Factor: This is a multiplier. Select this and enter an amount in the **Based on Adjustment Amount** field. Using this price factor, CPQ multiplies the List Price on the cart.
Currency	Click the drop-down list and select the currency. ⓘ The currency defined in the price list overrides the currencies that you define in other objects such as price list item, product, quote, quote line item.
Based on Adjustment Amount	Enter a value based on the adjustment type that you have selected.

Field	Description
Type	Select the type of Price list from the drop-down list. The following price list types are available: • Standard: Select standard price list if the quote derives it pricing from price list and its related price list items configured in the CPQ solution. • Contract: Select contract price list if the quote derives its pricing from an agreement. You must also associate a contract number when you select this price list.
Account	Associate an account with the price list.
Cost Model	Search and select a cost model. A cost model is a container holding all cost types (parent and child cost types). You can associate one cost model with multiple price lists. However, one price list can have only a single cost model associated with it. For more information, see Defining Cost Models and Associating Cost Models with a Price List.
Owner	Enter a name of the owner for Pricing activities.
Contract Number	Enter a contract number to which you want to associate this price list. This enables contract pricing in CPQ.

A price list is created and saved. You can associate all related pricing, products, and options with them.

Associate a Price List with Categories

Price list must be associated to an Offering category for the products to be displayed on the catalog.

The ASSOCIATED CATEGORIES tab on the price list displays the category associated to it and this category is displayed to the end user in the shopping cart. In this way, you can control the display of products by category.

Prerequisites

You must have an existing offering category and a price list.

To associate price list to a category

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**. A list of existing price lists is displayed.
3. Select a price list and click the **ASSOCIATED CATEGORIES** tab.
4. From the All Categories pane, drag and drop the required categories to the associated categories pane.
5. Click **Save**.

The price list is associated with a category.

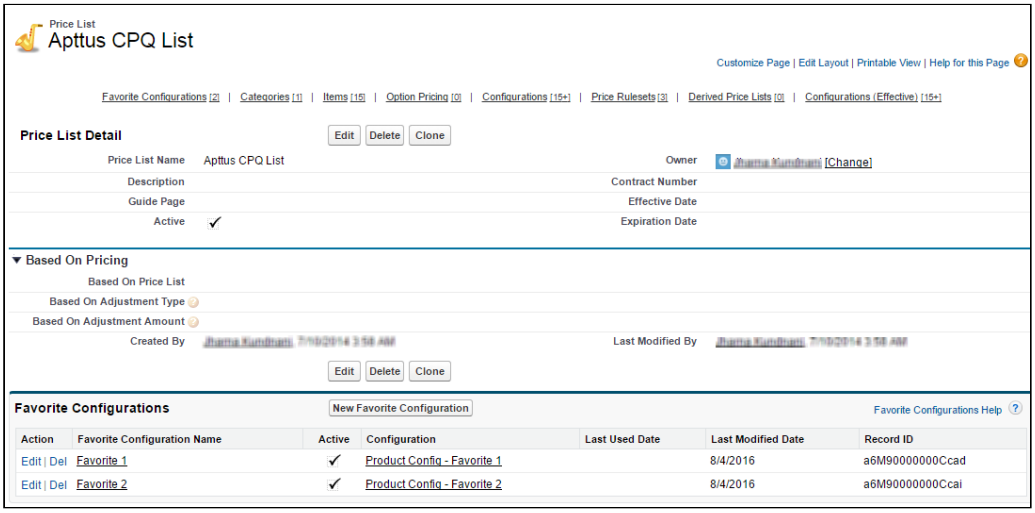
Viewing and Publishing Favorite Configurations for a Price List

After the Sales rep saves the favorite configurations, as an admin, you can view these favorite configurations collectively under a Price List. The favorite configurations are always linked to the Price List and after you associate this Price List to your Quote/Proposal, a separate category, named **Favorites**, containing your favorite configurations is displayed on the Catalog page.

You can do the following customizations on favorite configurations:

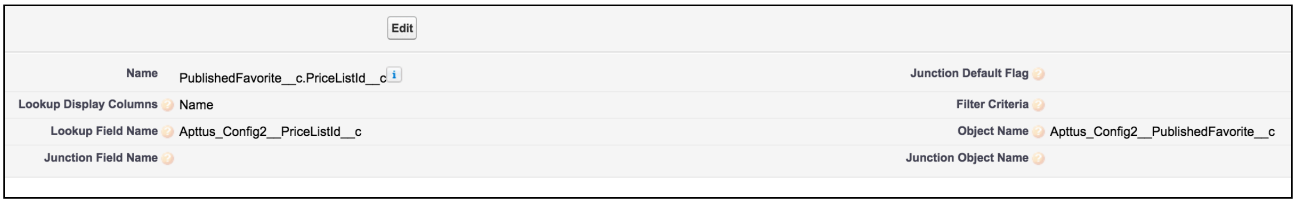
- Associate an image
- Customize the label of your favorite category
- Customize the fields that should be displayed to the sales reps on the catalog page
- Customize the columns to be displayed on the **Save as My Fav** dialog box on the cart.
- Disable the favorite configuration related UI elements.

For more details, see Configuring Favorite Settings.



The **Publish** button on your favorite configuration record allows you to publish a favorite configuration record to another price list. This is useful when you want to use the same configuration records across a number of price lists.

Ensure that you have configured Configuring Lookup Field Settings with the values as described in the image below before publishing your favorites to another price list.



Creating Price List Items

Price List Items contain detailed pricing information about a product.

A product can be set up with one or more price list items. Each price list item includes the list price for the product including details of charges such as per unit, flat price, one time, recurring, and more. All the fields on a price list item are price related and impact the final price on the shopping cart. Each price list item represents different ways, a customer is charged for a product. For example: License fee, Implementation fee, and more. Price list items are categorized into price lists.

To create a price list item

If a field is not available in the Admin UI, you must use CPQ Console to configure it.

1. Navigate to the **CPQ Console** tab.
2. Click **Manage Price Lists** and select an existing price list.

3. Scroll down to the Items related list, and click **New Item**.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**. A list of existing price lists is displayed.
3. Select a price list and click the **PRICELIST ITEMS** tab.
4. From the Products pane, drag and drop the required product to the right side.
5. Enter details in one or more of the following fields:

Field	Description
Charge Type	Select a charge type (mandatory). These are a separate, identifiable element of charges that can be used for pricing. For example: Standard Price, Maintenance Fee, Installation Fee, and more. This is a way to capture multiple types of fees for a product or line item. For instance, a product has a product fee but can also have a separate fee for installation. In such a case, you will create two price list item entries for the same product (by clicking the + icon) and specify the charge type for each one of the two entries. The charge type will be automatically added to the cart when you add the product so that all charges associated with the product are included in the pricing. Conga does not recommend setting up the bundle and its option to have different frequencies with the same charge type. ⓘ CPQ does restrict you from creating a PLI without a charge type, but you might not get appropriate results without a charge type.
List Price	Type a list price for a product in the price list item. If the price list item has matrices or may be overridden, enter **0** as the list price. You must enter the list price in the accordance with the value defined in the **Currency Field Precision** setting. The precision values for list price must be entered less than or equal to the defined currency precision. If you enter the list price precision values greater than the defined currency precision, CPQ might perform incorrect pricing calculations.

Field	Description
Price Type	Select *One Time*, *Recurring*, or *Usage*. Selecting *Recurring* or *Usage* enables the **Frequency** field. For instance, if you add a subscription product, you will choose type as recurring and then specify the frequency as monthly, yearly, and so on. Example: You charge 1$ per month for a term of 1year, the price is calculated accordingly.
Frequency	This is based on your selection from the Price Type field. Select an option to indicate how often the product will be charged.
Price Method	Select one of the following: • Per Unit: Multiplies the price with the quantity. • Flat Price: Applies a flat price. • Percentage: Enables the Percent, Percent Applies To, Related Price List Item fields. • Related Price: Used when the price is calculated as a function of the price of other lines in the same shopping cart. • Tiered Rate: Default price method for Usage Price Type. ⓘ If you select *Usage* in the **Price Type** field, you must ideally select *Tiered Rate* in the **Price Method** field. This is the right combination for the **Price Type** and **Price Method** fields. CPQ does not prevent you from creating other combinations, but you might not get appropriate results.
Active	To make the price list item active, select this check box.

Field	Description
Charge Type Criteria	Click the  icon and configure a charge type criteria and click **Save**. This is used for conditional charge types that are included only if a certain criteria is met and as part of adding the main product to the cart. Setting this criteria either by Line Item or Attribute field will only apply this price list item under those conditions. For example, this may be a charge type based on workstations; whereas the product may have a charge type based on users. A product should have at least one Price List Item without any charge type criteria, only then the pricing happens to the product in the Cart. If you are using the **Charge Type Criteria** setting, you must define a criteria for all PLIs. Ensure that you do not define that same criteria for more than one PLI. The criteria must be mutually exclusive so that CPQ does not apply two PLIs while pricing a product.
Percent, Percent Applies To, Related Price List Item	These allow you to set price dependencies such that one price is a percentage of the unit or extended price of another product. Refer Configuring Related Pricing for more information.
Description	Type a brief description to specify the purpose of the price list item.
Price Uom (Unit of measure)	Select an option to specify the unit price you are charging is for an hour, day, year, and so on. This helps the end user while pricing products in the cart. You can modify this in the price list item object.

Field	Description
Min Price/Net Adj %	Enter the minimum price or minimum net adjustment percentage. If the adjustment is a discount, use a negative number (for example, -5) When you select an option from the **Min/Max Price Applies To** drop-down, the pricing engine considers the value you enter in this field as the minimum price of the product. When **Min/Max Price Applies To** = *None*, the pricing engine considers the value you enter in this field as the minimum net adjustment percentage. The minimum limit is not applicable to manual adjustments.
Max Price/Net Adj %	Enter the maximum price or maximum net adjustment percentage. When you select an option from the **Min/Max Price Applies To** drop-down, the pricing engine considers the value you enter in this field as the maximum price of the product. When **Min/Max Price Applies To** = *None*, the pricing engine considers the value you enter in this field as the maximum net adjustment percentage. The maximum limit is not applicable to manual adjustments.
Min/Max Price Applies To	Select one of the following: • *Base Price* - This is the derived price for a product when you add it to the cart. This can be same as the list price or different from the list price, if you have defined a price matrix or a price rule that dynamically arrives at the price of the product based on other factors. • *Extended Price* - This is derived by (Base Price * Quantity * Term) + (Option Price * Quantity) where (Option Price * Quantity) is applicable in case of a bundle scenario. When you select an option from the **Min/Max Price Applies To** drop-down, the pricing engine considers the value you enter in the **Min Price/Net Adj %** or **Max Price/Net Adj %** field as minimum or maximum price of the product. When **Min/Max Price Applies To** = *None*, the pricing engine considers the value you enter in the **Min Price/Net Adj %** or **Max Price/Net Adj %** field as a minimum or maximum net adjustment percentage.

CPQ Pricing Engine does not support custom values for **Price Type**, **Frequency** and **Price Method** at the Price List Item level or in the Pricing Callback classes. Using custom values may cause unexpected system behavior.

6. Click **Save**. A price list item is created for a product or option. The price list items created display the products and options to the end user in the pricing cart page.

CPQ uses charge types as the consolidation factor for grouping. Such grouping can be at the bundle level, Summary Group level, or Sub-total level.
In this way, we ensure that the line items with the same **Charge Type** and **Frequency** calculate the following fields on the Cart appropriately:
- Selling Term
- Adjustments applied on a bundle line or a Summary group line

- Adjusting Options Pricing for a Bundle Manually
  The Options tab display the option product and associated price for a bundle. You can control how the options price rollup to the bundle price. You can also control the display of options adjustments on the pricing cart page.
- Applying Price Matrices from the Matrices Tab
  If the product has a matrix based price, click the Matrices tab.
- Applying the Default Pricing to Products
  You can apply default pricing to a product from the Defaults tab.
- Defining Tax and Billing for Products
  You can define the way your product should be billed and taxed from the Tax & Billing tab.
- Applying Miscellaneous Pricing to Products
  You can apply miscellaneous pricing to a product from the Miscellaneous tab.
- Configuring Charge Type Frequency as Weekly
  Products can now be configured to have recurring charges with a pricing frequency set as weekly. The customer would then be charged based on the number of weeks between the start and end dates of the product.

Adding Additional Details to a Price List Item

1. Select an existing price list item, click the more icon (⋮), and click **Advanced**.
2. Click the **DETAILS** tab.
3. Enter the following details:

Field	Description
Price List	Displays the price list, which is disabled for editing.
Charge Type Criteria	Click the  icon and configure a charge type criteria and click **Save**. This is used for conditional charge types that are included only if a certain criteria is met and as part of adding the main product to the cart. Setting this criteria either by Line Item or Attribute field will only apply this price list item under those conditions. For example, this may be a charge type based on workstations; whereas the product may have a charge type based on users. A product should have at least one Price List Item without any charge type criteria, only then the pricing happens to the product in the Cart. If you are using the **Charge Type Criteria** setting, you must define a criteria for all PLIs. Ensure that you do not define that same criteria for more than one PLI. The criteria must be mutually exclusive so that CPQ does not apply two PLIs while pricing a product.
Product	Search and select a product.
Charge Type	Select a charge type. These are a separate, identifiable element of charges that can be used for pricing. For example: Standard Price, Maintenance Fee, Installation Fee, and more.
Active	To make the price list item active, turn on the toggle button.
Sequence	Enter a sequence for the price list item.
List Price	Type a list price for a product in the price list item. If the price list item has matrices or may be overridden, enter **0** as the list price.

Field	Description
Cost	Type a cost of a product. You can set the cost in the price list currency. If the quote price list is in a different currency, the sales user will see the cost with the correct conversion rate applied. The subsequent fields of cost such as base cost, base extended cost, and extended cost are calculated after the currency conversion. If the **Disable Based On Currency Adjustment** setting is enabled on the price list, the currency conversion is not applied.
Effective Date	Enter an effective date for the price list item.
Expiration Date	Enter an expiry date for the price list item.
Price Type	Select *One Time*, *Recurring*, or *Usage*. Selecting *Recurring* or *Usage* enables the **Frequency** field. For instance, if you add a subscription product, you will choose type as recurring and then specify the frequency as monthly, yearly, and so on. Example: You charge 1$ per month for a term of 1year, the price is calculated accordingly.
Frequency	This is based on your selection from the **Price Type** field. Select an option to indicate how often the product will be charged.

Field	Description
Price Method	Select one of the following: • Per Unit: Multiplies the price with the quantity. • Flat Price: Applies a flat price. • Percentage: Enables the Percent, Percent Applies To, Related Price List Item fields. • Related Price: Used when the price is calculated as a function of the price of other lines in the same shopping cart. • Tiered Rate: Default price method for Usage Price Type. ⓘ If you select *Usage* in the **Price Type** field, you must ideally select *Tiered Rate* in the **Price Method** field. This is the right combination for the **Price Type** and **Price Method** fields. CPQ does not prevent you from creating other combinations, but you might not get appropriate results.

4. Click **Save**.

Adjusting Options Pricing for a Bundle Manually

The **Options** tab display the option product and associated price for a bundle. You can control how the options price rollup to the bundle price. You can also control the display of options adjustments on the pricing cart page.

To manually adjust options pricing for a bundle

You must have an existing price list item.

1. Select an existing price list item, click the more icon (⋮), and click **Advanced**.
2. Click the **OPTIONS** tab.
3. Enter information in one or more of the following editable columns:

Field	Description
Adjustment Amount	Enter an amount to adjust the options price. For Optional bundle, if you apply a discount on the bundle level, it is not populated in the Adjustment Amount and Adjustment Type field on the option. Discount is getting applied on the line Items, however, Adj Type and Amount fields are not getting populated for line items and Adj Type and Amount field are not getting cleared from the bundle product.
Adjustment Type	Select an adjustment type.
Price Included In Bundle	Select to include the price with the bundle product. Selecting Price Included in Bundle sets the Adjustment Amount to zero and Adjustment Type to Price Override. This means the option price is included in the bundle price and therefore is free.
Allow Manual Adjustment	Select this to enable the end user to make manual adjustments. Manual adjustments are done through the adjustment type and amount fields on the pricing cart page.
Allocate Group Adjustment	Select this to include the product in a manual group discount, if applied, on the pricing cart page.

4. Click **Save**.

The option price for a bundle is adjusted and saved.

Enabled Group Adjustment Spread and Enforced Min/Max Net %

- You can prevent the sales reps from applying any line level adjustments on one or more items in the cart because these items are sold at the price that CPQ calculates.
- If you have enabled the `Enable Adjustment Spread` custom setting and disabled the `Allow Manual Adjustment` on a line item, then CPQ sets the `Allocate Group Adjustment` as read-only for that line item. Note that

the sales reps cannot use **Mass Update** to update the **Allocate Group Adjustment** flag.

- This feature is based on client side pricing framework, hence you must set `Enable Auto Reprice` to True.
- After configuring the Min Net % and Max Net % on a price list item, you cannot apply the manual price adjustment outside of Min/Max Net % range.
- You can give percentage-based discounts for configuring a quote with a desired price by using group adjustment spread when you have Min/Max Net % associated with the line items.
- This feature can be used only if the bundle has no price of its own. This is because the allocation of group adjustments is only to the options and the bundle's own price will not be adjusted.
- Ability to set Minimum and Maximum Net Adjustment % is enabled only when "Enable Auto Reprice" is turned ON.

> To use Min/Max Net %, you must ensure that the **Adjustment Applies To** field is empty.

Conversion of Group Adjustments when bundle and options have ramps

Group Adjustment Spread provides you easiness of applying a group adjustment with the flexibility of a line level adjustment. You can update the adjustment type and amount on a bundle or summary group line and on reprice, the group adjustment gets replaced by line level adjustments on the options or standalone products.

Use Case 1:

When the bundle has three ramps with Start Date and End Date as:

Start Date	End Date
1-Jan-2017	31-Dec-2017
1-Jan-2018	31-Dec-2018
1-Jan-2019	31-Dec-2019

Then suppose there are options with ramp lines with the same dates. In that scenario, if you apply an adjustment to one of the ramp lines, only the corresponding ramp line gets adjusted and the other ramp lines remain the same. If a bundle does not have ramps, but the options have ramps in that case adjustment spreads to all the ramp lines.

Use Case 2:

If a bundle has a total price of $600, price override of $300, and one of the options has three ramp lines with extended prices of $100 each, then each of the option gets a price override of $50 (assuming all the options of the bundle are eligible and the corresponding option has flat rollup method).

- You can use the Group Adjustment Spread functionality even if the bundles have sub-bundles.
- Group Adjusument Spread functionality works for Discount Amount, Markup Amount and Price Override as well for the conversion of amount-based bundle adjustments to option adjustments.
- You can quicly apply adjustments using the Group Adjustment Spread functionality, althogh you have optional cart line items.

Applying Price Matrices from the Matrices Tab

If the product has a matrix-based price, click the Matrices tab.

A price list item can have more than one matrix evaluated in a sequence. For more information, see Defining or Creating Price Matrices.

In order to work with matrix-based pricing, you must create price dimensions for a product. You can create up to six dimensions, bringing in attributes from the line item or header objects of any data type within Salesforce. The dimension value type has three values: Discrete, Range, and Cumulative Range - Line Item.

Applying the Default Pricing to Products

You can apply default pricing to a product from the Defaults tab. This tab enables you to apply default quantity and pricing to a product.

To apply default pricing to a product

You must have an existing price list item.

1. Select an existing price list item, click the more icon (⋮), and click **Advanced**.
2. Click the **DEFAULTS** tab.
3. Enter the following details:

Field	Description
Default Quantity From	Select an option from where default quantity is to be considered. If the price list item derives its default quantity from a specific attribute, when the user navigates to the catalog page and selects a product and its attribute, the default quantity of the product is derived based on a number selected from an attribute of type picklist or number.
Default Quantity Field	This field is dependent on the Default Quantity From field. Select the Product Attribute using which the default quantity of the product is derived.
Default Quantity	Enter the default quantity.
Default Price From	Select an option from where default price can be considered. Supported values are: • Advanced Formula • Product Attribute
Default Price Field	Enter the name of the CPQ formula field (it can be normal or based on numeric expression) and select it from the search result. Whatever the value this field returns, that will be the list price of the product. ⓘ You must have selected *Advanced Formula* in the **Default Price From** for this field to work.

Field	Description
Default Selling Term	Enter the default selling term. If a product has two PLIs of *Recurring* type and PLI 1 has Default Selling Term mentioned on it while PLI 2 does not have Default Selling Term, then on product configuration, the selling term of PLI 2 gets auto-updated in line with the selling term of PLI 1. When Price Type = Recurring, CPQ calculates the selling term on the cart as follows: • If you provide the Default Selling Term (but the cart line item does not have start date and end date), CPQ calculates the selling term based on the default selling term. • If the cart line item has start date and end date (but you do not provide the Default Selling Term), CPQ calculates the selling term based on the start date and end date. • If you provide the default selling term and the cart line item also has start date and end date, CPQ gives precedence to the start and end dates, and calculates the selling term accordingly.
Auto Update Quantity	Select to automatically update the quantity on the Cart. This is useful when the quantity of your product is derived using a Numeric Expression and you want to reflect the changes made by the Numeric Expression on the Cart automatically.
Auto Cascade Quantity	Select to automatically cascade quantity.
Is Quantity Read Only	Select this to make the Quantity field non-editable on the cart.

Field	Description
Auto Cascade Selling Term	Select to automatically cascade selling term. Suppose you have one bundle with two options. Option 1 has Auto Cascade Selling Term set to False. Option 2 has Auto Cascade Selling Term set to True. If you change the bundle selling term, it will only cascade to Option 2, but not Option 1.
Is Selling Term Read Only	Select this to make the Selling Term field non-editable on the cart.
Auto Renew	Select this setting only when you want CPQ to automatically close the renewal quote that has an asset with **Auto Renew = True**. On the expiry date of the asset, CPQ automatically processes the renewal quote and renews the asset for the next term. The quote will be automatically set to *Accepted* without any intervention (this is called *Touchless Renewal*). This setting is used to group assets into different renewal quotes, along with the **Renewal Group Fields**. CPQ allows you to configure this setting at the price list item level, which will cascade to the cart and asset level. Do not use this setting to create auto renewal quote.

Field	Description
Auto Renewal Type	CPQ allows you to configure an asset as renewable or non-renewable at the price list item level. The supported values are: • *Fixed*: Select this option to make the asset renewable. CPQ can pull the product into Auto and OnDemand renewal quotes. • *Evergreen*: Select this option to define the selling term without an expiration date. This option makes the asset billable forever until the subscription is cancelled (evergreen billing). For example, if the product is of monthly subscription type, the customer keeps receiving the bills for the asset monthly (without requiring renewals). You cannot pull the product into an Auto or OnDemand renewal quote. For such assets, CPQ disables the **Renew** action on the Installed Products page (Sales Representative cannot renew the asset manually also). ⓘ Asset-Based Ordering is not supported for Evergreen assets. • *Do Not Renew*: Select this option to disable CPQ from pulling the asset into Auto or OnDemand renewal quotes. This value makes the asset non-renewable. ⓘ If you do not select any option for **Auto Renewal Type**, CPQ considers the asset as renewable. CPQ allows you to configure this setting at the price list item level, which will cascade to the cart and asset level.
Auto Renewal Term	Enter a value to define the auto renewal term that CPQ must consider while renewing assets. If you do not specify any value, CPQ considers the **Default Renewal Term** specified in Installed Product Settings. If both **Auto Renewal Term** and **Default Renewal Term** are null, CPQ considers the selling term on the original asset during renewal. CPQ allows you to configure this setting at the price list item level, which will cascade to the cart and asset level.

4. Click **Save**. Default pricing is applied and saved.

Defining Tax and Billing for Products

You can configure all Billing and Tax related information for your product from the Tax & Billing tab.

To define Taxes and Billing

You must have an existing price list item.

1. Select an existing price list item, click the more icon (⋮), and click **Advanced**.
2. Click the **TAX & BILLING** tab.
3. Enter the following details:

Field	Description
Taxable?	Select this check box if the order is a taxable product or service.
Tax Inclusive?	Select the check box if the price of the product is inclusive of taxes.
Tax Code	Lookup to select the relevant tax code.
Billing Rule	Select one of the following options • **Bill in Advance** - To bill your customer before the product is delivered • **Bill in Arrears** - To bill your customer after the product is delivered • **Ready for Billing Date** - To bill your customer on the **Ready for Billing Date** you defined for the Order Line Item • **Payment on Receipt** - To bill your customer when the invoice is presented

Field	Description
Billing Frequency	Select one of the following options • **Monthly** - To generate a bill once every month • **Quarterly** - To generate a bill once every three months • **Yearly** - To generate a bill once every year • **Usage** - To generate a bill based on usage
Min Usage Quantity	If you defined the **Billing Frequency** to be usage based, enter a value to set the minimum quantity of billable consumption.
Max Usage Quantity	If you defined the **Billing Frequency** to be usage based, enter a value to set the maximum quantity of billable consumption.

4. Click **Save**.

Defining Finance and Revenue for Products

You can configure all finance and billing related information for your product from the FINANCE & REVENUE tab.

To define finance and revenue

You must have an existing price list item.

1. Select an existing price list item, click the more icon (), and click **Advanced**.
2. Click the **FINANCE & REVENUE** tab.
3. Enter the following details:

Field	Description
Revenue Recognition Policy	Enter a revenue recognition policy associated with a product or service.

Field	Description
Revenue Split Policy	Enter an order revenue split policy associated with a product or service.

4. Click **Save**.

Applying Miscellaneous Pricing to Products

You can apply miscellaneous pricing to a product from the Miscellaneous tab.

This tab enables you to do the following:

- You can include an option price in a bundle.
- You can enable price ramps if a product is rampable, and enable commitments.
- You can control if a usage tier can be modified or not.
- You can manually adjust options pricing for a bundle. You can also control how the options price rollup to the bundle price and display of options adjustments on the pricing cart page.

To apply miscellaneous pricing to a product

You must have an existing price list item.

1. Select an existing price list item, click the more icon (⋮), and click **Advanced**.
2. Click the **MISCELLANEOUS** tab.
3. Select the following options as required:

Field	Description
Price Included In Bundle	Select to include the price with the bundle product. Selecting Price Included in Bundle sets the Adjustment Amount to zero and Adjustment Type to Price Override. This means the option price is included in the bundle price and therefore is free.
Enable Price Ramps	Select this check box to make a product price rampable.
Enable Commitment	Select this to enable commitment.
Is Usage Tier Modifiable	Select this check box to enable an end user to modify the usage tier.

Field	Description
Disable Asset Integration	Select to prevent the creation of asset corresponding to this product during the quote process. For now, you can disable asset creation only for standalone products. It is recommended not to block asset creation because it hinders the preferable system flow of asset creation on quote acceptance. Once you install Conga Billing version 4.97 installed or later, order activation continues as billing now ignores any line without an asset id. You cannot see bills created for those orders because there is no asset. CPQ does not support suppression of bundle products and does not encourage suppression of standalone products.
Allow Manual Adjustment	Select this to enable the end user to make manual adjustments. Manual adjustments are done through the adjustment type and amount fields on the pricing cart page.
Allocate Group Adjustment	Select this to include the product in a manual group discount, if applied, on the pricing cart page.
Enable Auto Ramp Creation	Select this check box to auto generate ramp line creation for the 2nd charge type, if price ramp is enabled for the first charge type.
Allow Proration	Select to allow proration. On the Cart page, to make Selling Frequency editable by the user, this check box must be selected.
Disable Cost Model	Select this check box to disable cost model and therefore, cost line items are not created when a product is added to the cart.

4. Click **Save**.

Miscellaneous pricing items applied to a product and saved.

Configuring Charge Type Frequency as Weekly

Products can now be configured to have recurring charges with a pricing frequency set as weekly. The customer would then be charged based on the number of weeks between the start and end dates of the product.

To set charge type frequency as weekly

A valid Price List should exist. If you are upgrading from a previous version follow the Post Upgrade Steps: Add the value Weekly to the picklists of the following objects:

Object	Picklist to add the value to
Price List Item	Frequency (Update the Price Type dependency picklist accordingly)
Asset Line Item	Pricing Frequency, Selling Frequency
Line Item	Pricing Frequency, Selling Frequency
Order Line Item	Pricing Frequency, Selling Frequency
Summary Group	Frequency
Frequency Conversion Rate	From Frequency, To Frequency

1. Create a Price List Item for a Product.
2. Set the **Price Type** as *Recurring*.
3. Set the **Frequency** as *Weekly*.
4. Specify the **List Price** and other requisite fields and click **Save**.

When a product having a charge with frequency as weekly is added to the cart, the term for that charge is calculated based on the number of weeks between the start date and end date. The final price will reflect the term.

Disabling Time Adjustment in Price List Item Fields

You can disable the time zone adjustment on the **Effective Date** and **Expiration Date** fields in the Price List Item. By default, the time on the **Effective Date** and **Expiration Date** fields is adjusted based on the Time Zone of the Locale in the org. If you disable the adjustment, CPQ does not change the time you manually configured in the **Effective Date** and

Expiration Date fields. Refer to Adding Additional Details to a Price List Item for information about the Price List Item fields.

To disable time adjustment in Price List Item fields

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record. The New Admin page is displayed.
3. Fill in the following details

Field	Value
Name	*APTS_DisablePLITimeZoneAdj*
Value	The following are the valid values: • *true*: disables time zone adjustment • *false*: enables time zone adjustment
Code	Leave the field blank

4. Click **Save**.

Configuring Price Escalators

Price Escalator is a new mechanism to define a markup on the different prices for a product. The markup can be defined on a previous or first ramp line on the cart.

On which prices is it applicable?

The markup is applied on the List Price, Base Price or Extended Price.

Applicable for which products?

The Price Escalator can be applied on Bundle and Option products. Please note that the bundle level escalator does not cascade to the option products.

Where is this applied?

The markup is applied on the first or previous ramp line (defined by the **Period** field) of bundle or option products on the cart.

When is this applied?

For bundle products, the price escalator is applied after the Price Matrices.

For option products, the price of the ramps is calculated depending on the value in the **Product Option Price Order** field.

Usecase

If the matrix gives a base price is 60 and if the YOY ramp markup has been defined as 10% over base price on period 2 and period 3, then the first ramp line will be $60, second will be $66 and the third will be $72.60.

To configure a Price Escalator

1. Navigate to your Price List Item record and search for **Price Escalator** related list.
2. Click **New Price Escalator** and enter the following details:

Field	Description
Period Number	Defines the ramp line number.
Adjustment Source	Defines the ramp on which the adjustment is applied.
Adjustment Applies To	Values for the different prices to which the adjustment is applied. Valid values are List Price, Base Price and Extended Price.
Adjustment Type	Values for manual adjustments whether you want to apply discount or markup on the price.
Adjustment Amount	Field to enter in the amount or percentage of manual adjustment.
Price List Item	This is a lookup to the Price List Item, containing the PLI Id by default.

3. Click **Save**.

Managing Price Rules

In addition to managing prices for products using price lists and price list items, you can manage prices using pricing rules. Pricing rules manage special offers, discounts, and charges at either the line item level or the summary line item level. Many pricing rules can be associated to a Ruleset and through to products.

Price Rulesets are a mechanism to allow particular families, categories or groupings of products to have either line item pricing adjustments applied or summary pricing

adjustments applied. Typical examples of these are volume discounting rules or promotional pricing rules.

- Creating Price Rulesets
  Price Rulesets allow for pricing adjustments across a range of products through price rules. Both line item pricing adjustments and summary pricing adjustments can be applied. For example: Discounting rules, promotional pricing, adjustments in aggregate based on a Product Family, and more.
- Creating Price Rules
  You can associate multiple rules to a ruleset or through to a product. Price matrices can be defined in price rules as well as conditional price rules (rules that use boolean logic such as AND/OR). Price rules can be dimensional, which use similar concept as Price Matrices or Conditional, that relate to fields for pricing adjustments.

Creating Price Rulesets

Price Ruleset allows pricing adjustments across range of products through price rules. Price Ruleset manages special offers, promotions and enables discounting based on the specific product attribute. A price ruleset can contain one or more rule entries and it can be dimensional, which employs a similar concept as Price Matrices. You can apply line item pricing adjustments as well as summary pricing adjustments. For example: Discounting rules, promotional pricing, adjustments in aggregate based on a Product Family, and more. Also, if your customer has Gold Rating, and you want to automatically discount specific line items on a proposal by 30%, you should use Price Ruleset.

To create a price ruleset

You must have an existing price list.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Ruleset**.
3. Click **New Price Ruleset**.
4. Enter details in one or more of the following fields, as required:

Field	Description
Price Rule Set Name	Enter a mandatory **Ruleset Name** and a mandatory **Sequence** in which the system will evaluate multiple rulesets. Typically, you will perform line item adjustments first and then any summary adjustments

Field	Description
Is Active	Select this to set the ruleset as active.
Enable Currency	Select this to enable currency for the price ruleset.
Currency	Select the currency for the price ruleset. If the price ruleset and a line item have different currencies, CPQ applies the currency conversion rate (defined at **Setup** > **Company Profile** > **Manage Currencies**) when applying the rule adjustment.
Short Description	Enter a description for the ruleset.
Effective Date	Not required as long as the price ruleset is Active, but can be used for promotional rules, and more.
Price List	Select a price list. The ruleset will only source product prices with the selected price list and then apply adjustment criteria.
Product Category	Select a category. The ruleset will only source product prices within the selected category.
Category	Select All, Agreement, or Proposal. This indicates if the ruleset is relevant to Agreements, Proposals, or both.
Application Level	Select a level to which this ruleset will be applied. The supported values are: • *Line Item*: Selecting this will apply the line item adjustment within the line item net price in the shopping cart. • *Bundle*: Selecting this will apply the adjustment to a bundle and its options and the adjustment is displayed in the Totals tab in the shopping cart. • *Aggregate:* Selecting this enables the **Application Method** field.

Field	Description
Application Method	This field is enabled if you selected *Aggregate* from the **Application Level** drop-down list. This indicates that you want the adjustment to select products in aggregate but apply an adjustment as a summary line in the Totals tab or spread the adjustment over numerous products. The supported values are: • *Apply to Line Items*: Applies the adjustment to line items on the cart. • *Create Summary Lines*: Applies the adjustment as a summary line in the Totals tab on the cart.
Ruleset Criteria	Allows you to set criteria for a line item rule or a bundle, depending on the Application level you have selected, such that the ruleset only applies when it satisfies a line level field value or a product attribute value. Click **New** to fill in your criteria.
Stop Processing more Rules	Selecting this indicates that if the ruleset is satisfied, the system will not process any more rules.
Enable Date Range	Selecting this enables you to set the effective date and expiration date on the rule entry.
Expiration Date	Select an expiration date.
Product Family	Select the Product Family. This is the Product Family field on the products object. The ruleset will only source product prices with the selected Product Family.
Product Group	Click to search and select a custom product group to the ruleset will apply the pricing adjustments. These custom product groups have no relation to a category, a Product Family, or any other product designation.
Charge Type	Select a charge type to which the ruleset will apply adjustments.

5. Click **Save**.

By filling out these criteria, the source products and prices are then designated and the ruleset can apply adjustments through price rules.

Creating Price Rules

You can associate multiple rules to a rule set or through to a product. Price matrices can be defined in price rules as well as conditional price rules (rules that use boolean logic such as AND/OR). Price rules can be dimensional, which use a similar concept as Price Matrices or Conditional, that relate to fields for pricing adjustments.

One ruleset can have multiple rules and are evaluated in order of the rule sequence. A rule can be dimensional or conditional. A price rule determines the actual price adjustment made.

To create a price rule

You must have an existing Price Ruleset.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Ruleset**.
3. Select an existing price ruleset.
4. On the **PRICE RULES** tab click **New Price Rule**.
5. On the **DETAILS** tab, enter the following details.
    a. In **Rule Name**, enter a mandatory rule name.
    b. In **Short Description**, enter a description for the rule.
    c. Selecting **Stop Processing More Rules** indicates that if the ruleset is satisfied, the system will not process any more rules.
    d. From **Rule Type**, select a type. The supported values are *Dimension* and *Condition*. In setting a dimensional price rule, the process is exactly the same as creating a price matrix for a product.

> The values on the **Rule Type** drop-down are based on what you add in **Setup** > **App Setup** > **Create** > **Objects** > **Price Rule** > **Rule Type** > **Values**. The same values are available on the **Rate Type** drop-down on the Benefit section of promotion creation. Some values that you add for **Rate Type** on the Benefit section of promotion cannot be used for **Rule Type** while creating a price rule. Ensure that you select only the supported values while creating a price rule and while defining promotions. On the **Rule Type** drop-down, only Dimension and Condition are supported.
> If you remove some values that are not supported for **Rule Type** from the **Price Rule** object > **Rule Type** field, you cannot see the removed values on the **Rate Type** drop-down while creating a promotion.

e. From **Adjustment Applies To**, select an option to which the adjustment is applied. This is for line item adjustments only. The supported values are Base Price and Base Extended Price.
f. By default, the price rule inherits the value of **Currency**, from the price ruleset.
g. The **Sequence** is auto-generated.
h. To make the rule active, select **Active**.
i. From **Adjustment Charge Type**, select an option to set summary level adjustments as a charge type. You can change this in the Price Rule object.
j. To allow complete removal of a price adjustment, select **Allow Removal of Adjustment**.

6. On the **DIMENSIONS** tab, enter the following details.
   a. From **Dimension**, select a price dimension. You can enter up to six dimensions, which bring in attributes from the line items or headers of any data type within Salesforce. For example, you can select a dimension on quantity.
   b. From **Dimension Value Type**, select one of the following:

Option	Description
Discrete	The system only considers the specific matrix values.
Range	The system considers the matrix values as inclusive of the numbers in the range.

7. Click **Save**. The price rule is successfully created and listed under the Price Rule list.
8. Click the add icon (+) in the price rule details section to add an adjustment to the price rule. You can add more than one price rule rows by clicking  and delete rows by clicking  . You can also move a row up or down by clicking or  .
9. Enter the following details and click **Save**.
   a. In the ***<Dimension>*** column, enter a value for the dimension you created.
   b. In the **Adjustment Amount** column, enter a percentage or a number to adjust the price.

      > In this column you enter the adjustment amount directly. Use this column if you are not using the **Adjustment Amount Source** column.

   c. In the **Adjustment Type** column, select the kind of adjustment required.
   d. In the **Adjustment Amount Source** column, select an adjustment source.

> Use this column if you are not using the **Adjustment Amount** column. In this column, you can select an adjustment amount that will be fetched from CPQ formula fields created on the required business objects and fields. CPQ fetches the adjustment value from those fields instead of fetching the adjustment amount directly from the **Adjustment Amount** column. However, you must select such formula fields that return only numeric values. CPQ applies the adjustment on the list price to derive the base price.

e. In the **Effective Date** column, select the date from when the price rule entry will be effective.

> This column is available only if the **Enable Date Range** is turned on. For more information, see Creating Price Rulesets.

f. In the **Expiration Date** column, select the date on which the price rule entry will expire.

> This column is available only if the **Enable Date Range** is turned on. For more information, see Creating Price Rulesets.

g. In the **UOM** column, enter the UOM for the price rule entry so that CPQ applies the UOM conversion rate if the price rule entry and the line item have different UOMs.

> You can set the UOM of a price rule entry when the **Adjustment Type** is Discount Amount, Markup Amount, or Price Override. Also, you must have configured the conversion rate for UOMs. For more information, see Configuring Unit of Measure (UOM) and Frequency Conversion Rate.
> The UOM you enter in the price rule entry here must be the source during conversion (**From Uom**).

Creating Price Matrices and Dimensions

Price Matrices are an advanced pricing concept used to define tiered pricing paradigms or complex pricing structures with multiple criteria. Common examples are pricing tiers for a product based on user count or particular customer or transactional dimensions.

- Creating Price Dimensions
  Price Dimensions are an important concept in order to link any field within Salesforce

into the pricing tables. Any field that will determine a pricing adjustment such as Quantity, Term, and more, needs a dimension created for it.

- Creating Price Matrices
  These are an advanced pricing concept used to define tiered pricing paradigms or complex pricing structures with multiple criteria. Common examples are pricing tiers for a product based on user count or particular customer or transactional dimensions. A price list can have more than one price matrix.

Creating Price Dimensions

Price Dimensions are an important concept in order to link any field within Salesforce into the pricing tables. Any field that will determine a pricing adjustment such as Quantity, Term, and more, needs a dimension created for it.

To create A price dimension

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Dimensions**. A list of existing price dimensions is displayed.
3. Click **New Price Dimension**.
4. In the **Name** field, enter a mandatory name for the dimension.
5. From the **Context Type** drop-down, select a context type.

   CPQ supports the out-of-the-box context types described in the following table. You can manage these context types at **Setup** > **App Setup** > **Create** > **Objects** > **Price Dimension** (object) > **Context Type** (custom field) > **Values** section.

Context Type	Description
Line Item	Allows you to choose any field on the line item shopping cart. For example: Quantity, Term, and more. When you select this context type, the following fields are enabled. • **Business Object**. This field is populated by default based on the Context Type selection. You cannot modify it. • **Field**: Select a line item field. • **Relation Type**: Select a relation type between the context type and the line item field. • **Cumulative Dimension**: Cumulative Dimension is applicable for Asset operations such as increment quantity. If you want to use a value from another dimension with the current dimension and use the resultant value as the final value, select a *Dimension Name* from the drop-down. The value from another dimension will be added to the value of the current dimension during evaluation. For example, you are creating a dimension on quantity and want to add the value of existing asset quantity to get the total quantity (Asset Quantity + Current Quantity) value, which will be the value evaluated when your current dimension is evaluated.

Context Type	Description
Product Attribute	Allows you to choose any product attribute that has been associated to a product through an attribute group. From **Attribute**, click to select an attribute group. When you select this context type, the following fields are enabled. • **Business Object**. This field is populated by default based on the Context Type selection. You cannot modify it. • **Attribute Group**: Search and select an attribute group. • **Attribute**: Select an attribute in the attribute group selected above. • **Cumulative Dimension**: Cumulative Dimension is applicable for Asset operations such as increment quantity. If you want to use a value from another dimension with the current dimension and use the resultant value as the final value, select a *Dimension Name* from the drop-down. The value from another dimension will be added to the value of the current dimension during evaluation. For example, you are creating a dimension on quantity and want to add the value of existing asset quantity to get the total quantity (Asset Quantity + Current Quantity) value, which will be the value evaluated when your current dimension is evaluated.

Context Type	Description

Custom	Allows you to define a custom syntax for CPQ to retrieve the value of a field. It is most commonly used to retrieve Quote or Agreement Header fields.
	This is always from the context of the Line Item, so the format for the fields must be:
	For Quote Header fields:
	`Apttus_Config2__ConfigurationId__r.Apttus_QPConfig__Proposald__r.INSERTYOURFIELDAPINAMEHERE`
	-or-
	For Agreement Header field:
	`Apttus_Config2__ConfigurationId r.Apttus_CMConfig__AgreementId__r.INSERTYOURFIELDAPINAMEHERE`
	ⓘ The field must be the fields API name and not the field label or name.
	When you select this context type, the following fields are enabled.
	• **Business Object**. This field is populated by default based on the Context Type selection. You cannot modify it.
	• **Business Formula**: Define a formula field.
	• **Cumulative Dimension**: Cumulative Dimension is applicable for Asset operations such as increment quantity. If you want to use a value from another dimension with the current dimension and use the resultant value as the final value, select a *Dimension Name* from the drop-down. The value from another dimension will be added to the value of the current dimension during evaluation.
	For example, you are creating a dimension on quantity and want to add the value of existing asset quantity to get the total quantity (Asset Quantity + Current Quantity) value, which will be the value evaluated when your current dimension is evaluated.

Context Type	Description
Formula Field	Allows you to create a formula field containing the syntax based on the custom syntax (refer to the context type *Custom*). You can define a syntax once and can reuse it across multiple price dimensions. There may be scenarios where pricing varies based on Customer Priority status. For example, a customer can have three statuses Gold, Silver, and Platinum. You can define a pricing dimension based on the status of the customer. The Customer Priority field is a custom field you define in the opportunity. You can create a formula field on the *Apttus_Config2__LineItem__c* object referencing the Customer Priority field. You can later define a pricing dimension for the same formula field and use it within the rulesets to define different pricing tiers for each customer priority. Even though the same Customer Priority field is available in the Line Item context type as well, in scenarios where you have a large number of line items defined with different data types, searching and setting specific values for the context type can be cumbersome. Searching and setting a formula field context type within Apttus_Config2__LineItem__c filters only the fields having formula as a datatype. When you select this context type, the following fields are enabled. • **Business Object**. This field is populated by default based on the Context Type selection. You cannot modify it. • **Formula Field**: Select a predefined formula field. • **Cumulative Dimension**: Cumulative Dimension is applicable for Asset operations such as increment quantity. If you want to use a value from another dimension with the current dimension and use the resultant value as the final value, select a *Dimension Name* from the drop-down. The value from another dimension will be added to the value of the current dimension during evaluation. For example, you are creating a dimension on quantity and want to add the value of existing asset quantity to get the total quantity (Asset Quantity + Current Quantity) value, which will be the value evaluated when your current dimension is evaluated.

Context Type	Description
Asset Line Item	Applicable for Service CPQ. Allows you to choose any field on the asset line item. When you select this context type, the following fields are enabled. • **Business Object**. This field is populated by default based on the Context Type selection. You cannot modify it. • **Field**: Select an asset line item field. • **Cumulative Dimension**: Cumulative Dimension is applicable for Asset operations such as increment quantity. If you want to use a value from another dimension with the current dimension and use the resultant value as the final value, select a *Dimension Name* from the drop-down. The value from another dimension will be added to the value of the current dimension during evaluation. For example, you are creating a dimension on quantity and want to add the value of existing asset quantity to get the total quantity (Asset Quantity + Current Quantity) value, which will be the value evaluated when your current dimension is evaluated. ⓘ For Service CPQ, you must manually add the following values for using a price dimension: • **Object**: Price Dimension (Apttus_Config2__PriceDimension__c) • **Picklist Field**: Context Type (Apttus_Config2__ContextType__c) • **Value:** Asset Line Item

Context Type	Description
Asset Attribute	Applicable for Service CPQ. Allows you to choose any asset attribute that has been associated to an asset. When you select this context type, the following fields are enabled. • **Business Object**. This field is populated by default based on the Context Type selection. You cannot modify it. • **Attribute**: Select an asset attribute. For Service CPQ, you must manually add the following values for using a price dimension: • **Object**: Price Dimension (Apttus_Config2__PriceDimension__c) • **Picklist Field**: Context Type (Apttus_Config2__ContextType__c) • **Value:** Asset Attribute

6. From the **Type** drop-down, select one of the following types of price dimensions:

The field **Type** is related to promotions. Based on the selected value, CPQ creates a dimension type for promotion. For example, you can create a quantity-based dimension by selecting *Quantity* from the **Type** drop-down. For more information, see Creating Price Dimensions for a Promotion.
To set up values for the **Type** field, go to **Setup** > **Create** > **Objects** > **Price Dimension** (Apttus_Config2__PriceDimension__c) > **Type** (Apttus_Config2__Type__c) field.

Type	Description
Amount	Select this if the price dimension is based on purchase amount.
Equipment	Applicable for Service CPQ. Select this value if the price dimension is for Service CPQ.
Quantity	Select this if the price dimension is based on quantity.
Standard	Select this for a standard price dimension.

Type	Description
Term	Select this if the price dimension is based on term.

7. In the **Description** field, enter a description for the price dimension.
8. Click **Save**.

Creating Price Matrices

These are an advanced pricing concept used to define tiered pricing paradigms, or complex pricing structures with multiple criteria. Common examples are pricing tiers for a product based on user count or particular customer or transactional dimensions.

As an administrator, you can associate a cost (custom field) to each price matrix entry because the cost varies with the price and hence the margin is affected. The price matrix entries that are applied on a line item can be accessed in the line item. A price list can have more than one price matrix. CPQ supports multiple matrix entries within a matrix (cumulative range matrix) to be applied.

To create price matrices

You must have existing price dimensions.

1. Select an existing price list item, click the more icon (⋮), and click **Advanced**.
2. On the **MATRICES** tab, click **New Matrix**.
3. On the **DETAILS** tab, enter the following details.
    a. **Matrix Name**: Enter a name for the matrix.
    b. **Sequence**: Enter a sequence number for the matrix.
    c. **Short Description**: Enter a description for the matrix.
    d. **Matrix Type**: Select *Dimension*. If you choose *Condition*, note that the option or attribute pricing at the 1st level sub-bundle is not supported.

    > ⓘ If **Price Type** is *Usage* on a price line item, then the **Matrix Type** should also be *Usage* while creating a price matrix. CPQ supports only this combination.
    > If **Price Type** is *Usage* on a price line item and **Matrix Type** is *Dimension*, and you try to use the **Adjustment Amount Source** field, it will not work. CPQ does not support this combination.

    e. **Stop Processing More Matrices**: Select this checkbox to stop the system from evaluating further matrices if a match is found within this matrix.

f. **Enable Date Range**: Select this checkbox to enable the start and end date for price matrix to execute.

> In Classic CPQ, the price matrix date is mapped with the start date of line item. If the start date of the line item on the Cart falls under the price matrix entry, CPQ applies the correct price matrix.

4. On the **DIMENSIONS** tab, enter the following details.
   a. From **Dimension**, select a price dimension. You can enter up to six dimensions, which bring in attributes from the line items or headers of any data type within Salesforce.
   b. From **Dimension Value Type**, select one of the following:

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Cumulative Range - Line Item</td><td>The values will be evaluated cumulatively. For example: If quantity = 15 and price is $1 for a range of is 0-10, and price is $2 for a range of 11-15, then 10x$1+5x$2 = $20 will be the price of the product.

As an administrator, you define step pricing so that users (sales representatives) can quote correct prices to customers. You can create a matrix where the Dimension Type is Cumulative Range. If the admin setting APTS_DisableCumulativeQuantityAcrossCart is set to True, CPQ applies the cumulative range matrix such that it does not span multiple line items of the same product.
The Admin Setting APTS_DisableCumulativeQuantityAcrossCart is not available in Admin tab by default. You can add the entry by following the steps below:
i. Go to All Tabs > Admin
ii. Click New, to create a new record.
iii. Fill the following details
<table><tr><th>Field</th><th>Value</th></tr><tr><td>Name</td><td>APTS_DisableCumulativeQuantityAcrossCart</td></tr><tr><td>Value</td><td>true or false</td></tr><tr><td>Code</td><td>Leave the field blank</td></tr></table>iv. Click Save.</td></tr>
</table>

Option	Description
	When you have applied *Cumulative Range - Line Item price* matrix on a product, if you create ramps for that line item, those ramps will have the matrix applied by default. Ramps will inherit the matrix from that line item. (If you create five ramps, all five ramps will have the matrix.)
Discrete	The system only considers the specific matrix values.
Range	The system considers the matrix values as inclusive of the numbers in the range.

5. Click **Save**. The price matrix is successfully created and listed under Matrices.
6. Click the add icon (+) in the matrix detail section to add an adjustment to the price matrix. You can add more than one matrix rows by clicking + and delete rows by clicking −. You can also move a row up or down by clicking ▲ or ▼.
7. Enter the following details and click **Save**.
   a. In the ***<Dimension_Name>*** column, enter a value for the dimension you created. The values are inclusive. For example, the first line here indicates a range of 0-10. The second line indicates a range of 11-20, etc. All adjustments are adjusting the price list item-list price. To account for an infinite number of values, set the quantity to 999999, which will pull a range of "infinity." In this example, setting 999999 indicates the range 41-999999.
   b. In the **Adjustment Amount** column, enter a percentage or a number to adjust the price.

      > ⓘ In this field you enter a static adjustment amount value. If you do not want the adjustment amount to be static, use the **Adjustment Amount Source** column.

   c. From **Adjustment Type**, select the kind of adjustment required, if the criteria is met. An example is an override to a list price, markup, or discount.
   d. In the **Adjustment Amount Source** column, select a CPQ formula field.

      > ⓘ Use this column if you want to make the adjustment amount as a variable that is evaluated in run-time. If you want to provide a static adjustment amount, use the **Adjustment Amount** column.

In the **Adjustment Amount Source** column, you can select a CPQ formula field that fetches a value from a numeric or currency type of custom field on the line item object. For more information, see Managing CPQ Formula Fields.
However, you must select such formula fields that return only numeric values. CPQ applies the adjustment on the list price to derive the base price. For more information on how to use a formula field in creating a price matrix, see Use Case: Using a CPQ Formula Field in Price Matrix.

Use Case: Using a CPQ Formula Field in Price Matrix

In this use case, you will create a price matrix using a CPQ formula field in the **Adjustment Amount Source** column. You want

1. Create a formula field *Auto_Pricing_FormulaPricing_NumericField* on the **Number of Beds** (custom) field on the line item.
2. Create a price matrix with **Matrix Type** as *Dimension*.
3. From the **Dimension** drop-down, select a price dimension on quantity.
4. From **Dimension Value Type**, select *Range* and save the matrix.
5. In the ***<Dimension_Name>*** column, enter 10.
6. From **Adjustment Type**, select *% Discount*.
7. In the **Adjustment Amount Source** column, select *Auto_Pricing_FormulaPricing_NumericField*.

If the Sales Representative enters 20 in the **Number of Beds** field, the value of the formula field is 20. CPQ populates the same value of 20 in the **Adjustment Source Amount** field. CPQ applies 20% discount on the quantity from 1 to 10.

Managing CPQ Formula Fields

You can create a CPQ formula field using a simple formula or using the Apttus Numeric Expressions Framework, so that you can use the formula field for configuring prices and adjustments. You can create a formula on any numeric or currency type of custom field on the line item object and use that formula as a dimension while calculating adjustments. Instead of providing the adjustment amount as a static number, you can use a CPQ formula to make the adjustment amount as a variable that is evaluated in run-time. You can use the CPQ formula field while creating a price rule, price matrix, related pricing, and default pricing (as a list price).

For example, you can create a formula field on the **Number of Beds** field on the line item object as follow:

- Name of the formula field: *Auto_Pricing_FormulaPricing_NumericField*
- Business Object: *Apttus_Config2__LineItem__c*
- Field: *Number_of_Beds__c*

You can use this formula field *Auto_Pricing_FormulaPricing_NumericField* in the **Adjustment Amount Source** field while creating a price matrix. If the Sales Representative enters 10 in the **Number of Beds** field, the value of the formula field is 10. CPQ populates the same value of 10 in the **Adjustment Source Amount** field.

> ⓘ To use numeric expressions while creating CPQ formula fields, you must add the Picklist value **Expression**, to the Picklist field **Use Type**. Go to **Setup** > **Create** > **Objects** > **Formula Field (CPQ)** > **Use Type** field.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage CPQ Formula Fields**.
3. Click **New Formula Field**. The DETAILS tab is displayed.
4. Enter the following details and click **Save**.
    a. **Name**: Enter a name for the formula field.
    b. **Business Object**: Select a business object on which you want to create the formula field.
    c. **Description**: Enter a description for the formula field.
    d. **Usage**: Select the type of usage for the formula. Supported values are:
        - **Reference** (default): If you select this option, the **Formula** field is displayed.
        - **Expression**: If you select this option, the **Expression** field is displayed.
    e. **Formula**: Enter a formula as a text.
    f. **Expression**: Click the expression lookup. The Edit Expression dialog box is displayed. Configure a formula using numeric expressions and click **Ok**. For more information, see Configuring Numeric Expressions.

Configuring Related Pricing

Related pricing is used when you want to derive the price of a Line Item depending on the price of another Line Item. The Line Item can be of a Standalone, Option or Bundle product. For example, if the maintenance fee of an Option product is 10% of the over all maintenance fee of the main Bundle product, you must create two separate Price List Items for both the products. The Price List Item of the Option product should contain the related pricing fields populated with the values from the Bundle product.

You must perform the necessary configurations at the Price List Item to enable related pricing.

- Add related pricing information on a Price List Item
- Create a record in Related Price List Item (To) related list

Adding related pricing information on a Price List Item

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**.
3. Click **New PriceList**.
4. Select **Price Method** as *Related Price*.
5. Click the **DETAILS** tab. The different fields used for related pricing, such as **Related Adjustment Amount** and **Related Adjustment Type** are displayed.
6. Enter the following details and click **Save**.

Field	Description
Related Adjustment Amount	Indicates the adjustment amount that you want to apply on the related product. In our example, the value should be 10. ⓘ In this field you enter the adjustment amount directly. Use this field if you are not using the **Related Adjustment Amount Source** field.
Related Adjustment Type	Indicates the type of adjustment that you want to apply on the related product. Available values are: *%Discount*, *%Markup, Discount Amount*, *Markup Amount*, *Percentage*, *Price Factor*, and *Price Override*. In our example, choose *Percentage*.
Related Adjustment Applies To	Indicates on which price the adjustments for the related price should be applied. Available values are: *Base Extended Price*, *Base Price*, *Extended Price*, *List Price*, *Net Price*, and *Net Unit Price*. In our example, choose *Base Price*.

Field	Description
Related Adjustment Amount Source	Indicates the adjustment amount source that you want to apply on the related product. Use this column if you are not using the **Related Adjustment Amount** column. In this column, you can select an adjustment amount that will be fetched from CPQ formula fields created on the required business objects and fields. CPQ fetches the adjustment value from those fields instead of fetching the adjustment amount directly from the **Related Adjustment Amount** column. However, you must select such formula fields that return only numeric values. CPQ applies the adjustment on the list price to derive the base price. In our example, if the formula returns 10, CPQ applies 10% discount to the base price of the related product.

Creating a record in Related Price List Item related list

In order to relate two Price List Items and implement related pricing, you must create a record in the Related Price List Item (To) record.

1. Navigate to Related Price List Item (To) related list and click **New Related Price List Item**.
2. Enter the requisite information:

Field	Description
Price List Item	Denotes the ID of the Price List Item for which you are creating this record.
Related Price List Item	Denotes the ID of the Price List Item whose price is used as a basis for the related pricing.
Relation Type	Denotes how the Price List Items are related with each other. Currently, *Related Price* is supported.

Field	Description
Related Adjustment Amount	Indicates the adjustment amount that you want to apply on the related product. In our example, the value should be 10. You can specify this amount either here or at the Price List Item level.
Related Product	Indicates a filter that you can consider as a source for related pricing. This field must be clubbed with Related Charge Type for correct functioning.
Related Charge Type	Indicates the Charge Type for the Related Price List Item. This should be validated with the Charge Type on the Price List Item record.
Related Product Family	Indicates a filter that you can consider as a source for related pricing. This field must be clubbed with Related Charge Type for correct functioning.
Related Product Group	Indicates a filter that you can consider as a source for related pricing. This field must be clubbed with Related Charge Type for correct functioning.
Related Adjustment Type	Indicates the type of adjustment that you want to apply on the related product. Available values are: Percentage, % Discount, Discount Amount, %Markup, Markup Amount. In our example, choose *Percentage*.

Currently, the combination with Related Charge Type and Related Price List Item is supported and the combination with Price List Item and Related Price List Item is not supported.

To configure the context for Related Pricing

You can set the related pricing context for Line Items using the custom setting, **Related Price Scope**, available at **Setup** > **Build** > **Develop** > **Custom Settings** > **Config System Properties** > **Manage**. This setting enables you to choose if you want to perform the related pricing calculations over the entire Cart or confine the calculations to a Bundle product only. For example, if the price of an Option product is a percentage of another Option product in the same bundle, consider the following scenarios with **Related Price Scope** = *Cart* and *Bundle*:

- **Related Price Scope** = *Cart*, the Option price is based on the price of all the instances of the related product in the Cart, whether it is in the same Bundle or a Standalone or in an another Bundle.
- **Related Price Scope** = *Bundle*, the Option price is based on the related product instances within the same Bundle. You must also ensure that the Option products are a part of that bundle.

Use Cases: Related Pricing

When **Related Price Scope** = Cart, CPQ calculates the price of the dependent product based on the instances of the related product in the entire cart. CPQ checks for the related product in the whole cart.

When **Related Price Scope** = Bundle, CPQ calculates the price of the dependent product based on the instances of the related product in the same bundle. CPQ checks only for the bundle and its options.

In all these use cases, if you add the dependent product to the cart first, its price is zero because there are no primary products in the cart. If there is only one primary product, the dependent product's price is calculated based on one primary product. If there are two primary products, the dependent product's price is calculated based on two primary products. If you delete primary products, the price of the dependent product is zero.

Use Case 1

Primary Product 1 = USD 1000

Primary Product 2 = USD 1000

Dependent Product 1 is related to both Primary Product 1 and Primary Product 2 (at 10% Discount)

When **Related Price Scope** = Cart, if you add all three products to the cart, CPQ calculates the price of Dependent Product 1 based on the instances of Primary Product 1 and Primary Product 2 in the entire cart, at 10% discount on the total. Sum of Primary Product 1 and Primary Product 2 = 2000 and 10% discount on 2000 is 200. So the price of Dependent Product 1 = USD 1800 (2000 – 200).

When **Related Price Scope** = Bundle, if you add all three products to the cart, CPQ calculates the price of Dependent Product 1 based on the instances of Primary Product 1 and Primary Product 2 in the same bundle Dependent Product 1 (be it a bundle or option). CPQ looks only for that bundle where Dependent Product 1 exists and there is no instance of

related products Primary Product 1 and Primary Product 2 in the bundle. So the price of Dependent Product 1 = USD 0.

Use Case 2

Bundle 1 = USD 1000

Option 1 = USD 100

Option 2 is related to both Bundle 1 and Option 1 (at 10% Discount)

When **Related Price Scope** = Cart, CPQ calculates the price of Option 2 based on the instances of Bundle 1 and Option 1 in the entire cart, at 10% discount. If you add Bundle 1 to the cart once, CPQ calculates the price of Option 2 based on the instances of Bundle 1 and Option 1, at 10% discount. Because there is only one instance of Bundle 1 and Option 1 in the cart, the price of the Option 2 is (1000 + 100) – 110 = 990. If you add Bundle 1 to the cart a second time, the price of Option 2 in both instances is 1980, because there are two instances of Bundle 1 and option 1 in the cart (990 * 2 = 1980).

When **Related Price Scope** = Bundle, if you add Bundle 1 to the cart, CPQ calculates the price of Option 2 based on the instances of Bundle 1 and Option 1 (in Bundle 1), at 10% discount on the total. Sum of Bundle 1 and Option 1 = 1100 and 10% discount on 1100 is 110. So the price of Option 2 = 990 (1100 – 110). If you add Bundle 1 to the cart a second time, the price of Option 2 remains 990 because CPQ looks inside Bundle 1 only (though this is the second instance of Bundle 1).

Use Case 3: Discount Amount

Primary Product 1 = USD 1000

Primary Product 2 = USD 1000

Dependent Product 1 is related to both Primary Product 1 and Primary Product 2 (at 10 Discount Amount)

When **Related Price Scope** = Cart, CPQ calculates the price of Dependent Product 1 based on 10 discount amount on the instances of Primary Product 1 and Primary Product 2 in the entire cart. If Primary Product 1 and Primary Product 2 are not added to the cart, but only Dependent Product 1 is added, the price of Dependent Product 1 is USD – 10.

When **Related Price Scope** = Bundle, CPQ calculates the price of Dependent Product 1 based on 10 discount amount on the instances of Primary Product 1 and Primary Product 2 in the same bundle where Dependent Product 1 exists. There is no instance of Primary Product 1 and Primary Product 2 in the bundle where Dependent Product 1 exists, so the price of the Dependent Product 1 is USD – 10.

Use Case 4: More Scope (product group, product family, or charge type)

On related price, you can define the scope such as product group, product family, or charge type.

Primary Product 1 = USD 1000 (Charge Type = Standard Price)

Primary Product 2 = USD 1000

Dependent Product 1 is related to both Primary Product 1 and Primary Product 2 (at 10% Discount) and Charge Type = Standard Price

When **Related Price Scope** = Cart, if you add Primary Product 1 and Primary Product 2 to the cart, CPQ considers only those products that have the same charge type for calculating the price of Dependent Product 1. In this case, only Primary Product 1 has Charge Type = Standard Price. So the price of Dependent Product 1 is 10% discount of the instances of Primary Product 1 in the cart. 10% discount of 1000 is 100. So the price of Dependent Product 1 = USD 900.

CPQ skips Primary Product 2 even though it is related to Dependent Product 1.

Use Case 5: Custom Group

On related price, you can create a custom group by defining some formula fields.

Primary Product 1 = USD 1000 (PLI_CustomField = 10)

Primary Product 2 = USD 1000 (PLI_CustomField = 20)

Dependent Product 1 is related to both Primary Product 1 and Primary Product 2 (at 10% Discount) - You can create a custom field where (PLI_CustomField = 20)

When **Related Price Scope** = Cart, if you add Primary Product 1 and Primary Product 2 to the cart, CPQ considers only those products that have the same custom field for calculating the price of Dependent Product 1. In this case, only Primary Product 2 has PLI_CustomField = 20. So the price of Dependent Product 1 is 10% discount of the instances of Primary Product 2 in the cart. 10% discount of 1000 is 100. So the price of the Dependent Product 1 = USD 900.

CPQ skips Primary Product 1 even though it is related to Dependent Product 1.

Use Case 6: Locations

If primary product and dependent product have different locations, related pricing does not work. In that case, the net price of the dependent product will be 0. For related price to work, both products must be in the same location.

- Dependent Product 1 is related to Primary Product 1 (at 10% Discount)
- Related Adjustment Applies To: Base Price or Net Price

When you add both products to the cart:

Product	Net Price	Base Price	Location
Primary Product 1	1000	1000	Australia
Dependent Product 1	100	100	Australia

After you change the location of Dependent Product 1:

Product	Net Price	Base Price	Location
Primary Product 1	1000	1000	Australia
Dependent Product 1	0	0	India

ⓘ If there is a requirement for related price to work even if the location is different for both products, you must use a custom location field.

Enabling Price Ramps

Price Ramps are enhanced pricing methods for service/subscription-based industry.

As an administrator, you must enable price ramps from the price list item details page. Once you enable price ramps, an end-user can create a price ramp for a product to spread pricing across time periods. The user can also create user quantity ramps to define different pricing across different quantities.

ⓘ You cannot make a price list item rampable and tierable at the same time.

When you have applied *Cumulative Range - Line Item* price matrix on a product, if you create ramps for that line item, those ramps will have the matrix applied by default. Ramps will inherit the matrix from that line item. (If you create five ramps, all five ramps will have the matrix.)

To enable price ramps

You must have an existing price list item.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**. A list of existing price lists is displayed.
3. Select a price list and click the **PRICELIST ITEMS** tab.
4. Select an existing price list item, click the more icon (), and click **Advanced**.
5. Click the **MISCELLANEOUS** tab.
6. Select the **Enable Price Ramps** checkbox.
   The price becomes rampable for a product. After the price is rampable, you can create and save price ramps from the configuration page and the pricing cart page.

Support for Price Ramp on the Cart Grid UI
Cart Grid UI shows a Price Ramp icon. When you click the icon, it opens an Auto Ramp wizard if you have enabled auto cascade ramp setting on the PLI.

Enabling Auto Ramp Creation

As a sales representative, you might want to auto generate ramp line creation based on general information, like Start Date, End Date, and Term. In the previous releases, when you created price ramps, every line was created individually. Also, the sales representative might want to set up some custom date to adjust charge lines for customers. To address the first scenario, check the **Enable Auto Ramp Creation** under the Miscellaneous tab on a Price List Item page to enable automatic price ramp on the other charge type.

Consider the following scenarios:

Scenario 1

Consider a scenario where a product has two charge types. When using price ramp on any one of the charge types, the price ramp should be created for the other charge type too. In the previous releases, the system could create a price ramp for

only one charge type with the help of an API. This issue has been addressed in the current release, with the introduction of **Enable Auto Ramp Creation** in the Price List Item under the Miscellaneous tab to enable automatic price ramp on the other charge type.

Scenario 2

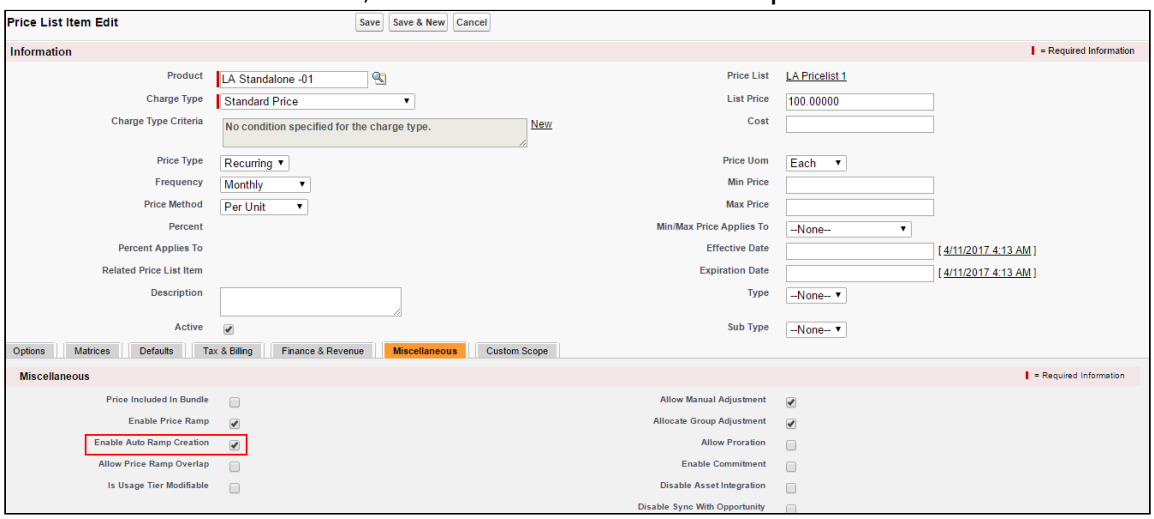
As a sales representative, you might want to auto generate ramp line creation based on general information, like Start Date, End Date, and Term. Also, you might want to set up some custom date to adjust charge lines for customers. To auto generate ramp line creation, check the **Enable Auto Ramp Creation** in the Price List Item page under the Miscellaneous tab to enable automatic price ramp on the other charge type. To set up custom date, you can set the start date, end date, terms and frequency period during ramp creation. You can select the frequency period as 'Monthly', 'Quarterly, 'Yearly' or as 'Custom'. If you select 'Custom' as frequency period, then mention the number of ramps in the text field. The start date, end date, and terms are available when added to the cart. When you click 'save', all the lines are created at the same time. When you select 'Monthly', 'Quarterly, or 'Yearly', the system creates a number of ramps based on Start Date, End Date, and Term. If 'Custom' is selected as the frequency, the system creates a number of lines based on the number of ramps.

To enable Auto Ramp creation

You must have an existing price list item.

1. From CPQ Console, go to your target price list.
2. Under the related list for *Items*, click **Edit** for the required price list item. The Price List Item Edit page is displayed.

3. On the **Miscellaneous** tab, select the **Enable Auto Ramp Creation** check box.



This auto generates the ramp line creation based on general information, such as Start Date, End Date, Term, and Frequency period.

Enabling Tiered Pricing

Price Tiers enable an end user to view or set up tiers on the configuration page and the pricing cart page. This ensures there are pricing settings (product line item) that determine if tiers can be modifiable or just viewable on the configuration page and the pricing cart page.

You can view existing pricing tiers and adjust pricing on the tier lines (if allowed). This also provides the capability to do tiers per units or usage and also capture additional usage charges for tiers.

If you have defined price matrices, the tiered pricing for a product is inherited from there. If you make any adjustments on the line item that has a tiered rate defined, the adjustments apply to tiered rate and also affect the price matrices.

If you are using usage-based pricing and have a matrix defined, the first dimension should always be a numeric value such as Quantity and not a string value.

> ⓘ You cannot make a price list item rampable and tierable at the same time.

To enable tiered pricing

You must have an existing price list item.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**. A list of existing price lists is displayed.
3. Select a price list and click the **PRICELIST ITEMS** tab.
4. From **Price Type**, select *Usage*.
5. From **Price Method**, select *Tiered Rate*. If Tiered Rate is not available as a choice, you can add it to the custom field object.
6. To enable a user to modify a usage tier.
    a. Click the more icon (), and click **Advanced**.
    b. On the **MISCELLANEOUS** tab, select the **Is Usage Tier Modifiable** checkbox.
7. Click **Save**.

Tiered pricing is enabled. If the price list item has tiers defined on the price list already, predefined tiers are displayed on the options/configuration or the pricing cart page.
You can create or modify tiers for pricing on the options/configuration and the pricing cart page.

To include a picklist value in an object

1. Navigate to **Setup** > **Create** > **Objects**.
2. Click the custom object. For example, click **Price List Item** label next to the Conga Configuration & Pricing installed package.
3. Scroll down to the Custom Fields & Relationships related list and click the custom field. For example, click the **Price Method** field.
4. Scroll down to the Picklist Values related list and click **New**.
5. Type a picklist value in the text area. For example, *Tiered Rate*.

    > ⓘ You can add more than one picklist values. Each picklist value must be on a separate line.

6. Click **Save**.

A picklist value is added to an object field.

Running a Criteria Maintenance Job

Criteria maintenance is a batch job that must be run whenever changes are made to pricing criteria fields or constraint criteria fields.

This includes:

- Adding or removing criteria in pricing fields.

- Any modification to the price list, price list item, price matrix, price rule, a default rule, or price dimension.
- Adding or removing a constraint rule.
- Adding or removing criteria in constraint rules.
- Any change to the constraint criteria fields.

To run a criteria maintenance job

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. Click the menu icon() on the main menu bar at the top.
3. Click **Criteria Maintenance**. The Criteria Maintenance popup is displayed.
4. To update all pricing criteria fields, click **Update Pricing Fields** or to update all constraint criteria fields, click **Update Constraint Fields**.

This executes an asynchronous batch job that maintains the criteria changes. The administration task is complete and an updated history for all the batch jobs is displayed. The key item to observe is Status. When Completed is displayed it means the job has run successfully, even if the percentage indicator remains at 0%.

> **Incremental Update for Criteria Maintenance Job**
>
> The Criteria Maintenance Jobs are incremental instead of rebuild so that existing records are not impacted.
>
> - When you execute the constraint field maintenance job, CPQ generates new constraint field records as an insert operation.
> - When you execute the pricing fields maintenance job, CPQ generates new pricing fields records as an insert operation.
> - When you execute the expression maintenance job, CPQ generates new expression records as an insert operation.
> - Existing records remain as is.
> - CPQ does not have downtime for quotes with existing records.

Configuring Contract Pricing

As a Pricing Administrator, you can store contracted prices in CPQ. The contract pricing feature offers a way to negotiate a price for a product that is not being purchased in the

current quote. CPQ uses a set of agreed upon prices for a customer for all the subsequent transactions. CPQ provides the following options to implement contract pricing.

CPQ supports contracted price lists specific to customers. Each customer might have different pricing information. Whenever customers order products or services with the same contracted price list, agreed prices will be given to them.

CPQ can maintain contract price lists (customer specific price lists) to keep a track of negotiated prices for each customer. The customer provides a set of agreed upon prices using an offline process or mechanism. You must set up this information as a price list using the standard CPQ capability. After you set up this price list, any new quote or order coming into CPQ will specify the customer specific price list as the price list on the quote or order.

When to use Contract Price Lists?

- Low customer base: Customer specific price lists make perfect sense, when the number of customers is not huge, as creating customer specific price lists creates a lot of data in CPQ and over a period of time, with changes and renewals, the number can grow exponentially.
- Low number of SKUs in catalog: You must use customer specific price lists when the number of products being negotiated with a customer is of the magnitude of less than 1000 products. If the product catalog contains a large number of products, customer specific price lists do not offer the flexibility as price programs. In such cases, price programs make better sense.
- Better suited for standalone products: If the product catalog contains a large number of highly complex configurable products, customer specific price lists do not make sense as each product can be configured a lot of different ways.

Contract Pricing is supported for standalone and bundles. Contract Pricing for Options is supported provided options are defined in the same contract price list as the bundle. Options inherit the contract price list from the bundle. If an option is included in two different bundles in a quote, CPQ adds the option twice to the Contract Price List (resulting in two contract price list items). CPQ then applies pricing on the quote based on the selection of the bundle and options, and calculates the price of the option from the price list item that has relation to the current bundle under which it is being configured.

> ⓘ The contract pricing for bundles works fine for a single-level bundle and same option, however it does not work for same sub-bundles.
> Contract pricing does not work for when an option is part of different bundles and inherit price from different price lists.

Configuring Contract Price Lists

To enable contract pricing in CPQ, you must create a price list and store the contract prices. You must set the Type = Contract and associate the price list a *Contract number*.

To configure a contract price list

You must have an existing price list. You must have an existing contract number or agreement number.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQAdmin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**. A list of existing price lists is displayed.
3. Click **New PriceList**.
4. From the **Type** drop-down list, select Contract. The quote derives its pricing from a contract.
5. In the **Contract Number** field, enter a contract number or an agreement number.
6. Enter values in the required fields.
7. Click **Save**.

A contract price list is created.

Defining Contracts to be Used in a Quote

You must define (explicitly) which contracts to be used in quotes.

You can define the contracts to be used in a quote at:

- Quote Level: Use a specific contract or a set of contracts to explicitly pass the contract number(s) in the **Configure Products** action. You must create a custom button and add the parameter **cntrNbr_1** to the button formula. CPQ determines the contract pricing for quotes based on the parameter in the cart.
- Quote Line Item Level: Sales Representatives can add the contract number at the quote line item level. The **Contract Numbers** field is available on the Line Item record. This applies contract pricing on specific line item only. This parameter is useful when users want to use different contracts for different products. They can apply contract pricing on specific set of products or some other criteria based on condition and not on all products.

When the quote is priced, for each line item, first the price list associated with the first contract on the list is evaluated. If the part is not found on the price list, then the next

contract number is evaluated, and so on. If part is not found in any of the price lists associated with the contracts on the list, then the standard price list is used.

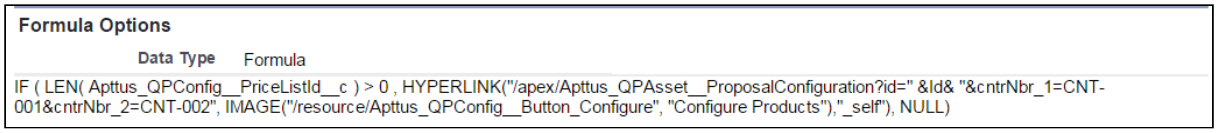
Prerequisites

- The contract must be active.
- Contract has the contract number populated and a Price List associated with it (the Related List will have the Price List).
- The contract number on the price list must be the same as the contract number on the Contract.

To define a contract to be used in quote

You must be in the Quote/Proposal object.

1. In **CPQ Console**, go to **Setup** > **App Setup** > **Create** > **Objects**.
2. Scroll down the list and click **Quote/Proposal**.
3. From **Custom Fields & Relationships**, click **New**.
4. From **Data Type**, select **Formula** and click **Next**.
5. Enter a mandatory **Field Label** and press **Tab** on your keyboard, the **Field Name** is auto-populated. For example: In Field Label, enter **Configure Products (Contract Pricing)**, the Field Name is auto-populated with **Configure_Products_Contract_Pricing**.
6. From **Formula Return Type**, select **Text** and click **Next**. The Step 3. Enter formula page is displayed.
7. Type the code in the area displayed and click **Next**. See example code.



8. On Step 4. Establish field-level security page, make the necessary changes and click **Next**.
9. On Step 5. Add to page layouts page, select the page layout that should include this field and click **Save**.

The **Configure Products (Contract Pricing)** action button is displayed on the Quote/Proposal page.

Alternately, you can also create a field on the Quote header and enter comma separated contract numbers in that field. Use the following formula text for the Configure Products button. The system automatically looks into contract numbers provided in sequence.

```
IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/
Apttus_QPConfig__ProposalConfiguration?id=" &Id& "&cntrNbr_1="&Contract_price_list_1_
_c,IMAGE ("/resource/Apttus_QPConfig__Button_Configure", "Configure Products"),"_self
"), NULL)
```

Defining Multiple Contracts to be Used in a Quote

Having multiple contracts on one quote

CPQ supports passing multiple price lists on a Quote/Agreement/Order to see both Non-Contracted and Contracted Products. Sales Representatives may want to display contracted and/or non-contracted products. In this case, you can pass a list of comma-separated contracted price lists (sequence is important because CPQ follows the same sequence as you pass it).
CPQ uses the default price list whenever a user creates a proposal. CPQ creates an account-specific price list when a user creates an agreement. CPQ tries to search if any account specific price list available or not. If the account-specific price list is available, CPQ uses it; if the account-specific price list is not available, CPQ uses the default price list.

> ⓘ A proposal can have the default price list but the price list item on line item record displays the applied price list item information.

Use Case

A customer has multiple accounts based on locations.

- ABC New York
- ABC California
- ABC USA Operations (parent account)

If you are quoting from ABC New York and a product's pricing is available in ABC USA Operations, the pricing should be selected from ABC USA Operations and not from ABC New York. If there is parent account for ABC USA Operations, then look for applicable parent account pricing.

Solution

There are different price lists, that is, ABC New York, ABC California, and ABC USA Operations with contract number populated on each price list.

- ABC New York Price List contract number is 23456789.
- ABC California Price List contract number is 12345678.
- ABC USA Operations Price List contract number is 01234567.

You must set the parameter **cntrNbr_1** in the correct sequence using the **Configure Products** formula button as follows:

```
IF ( LEN( Apttus_Config2__PriceListId__c ) > 0 , HYPERLINK("/apex/
Apttus_Config2__OrderConfiguration?id=" &Id &"&cntrNbr_1=23456789&cntrNbr_2=12345678"&
 "&flow=NGDefault", IMAGE("/resource/Apttus_Config2__Button_Configure", "Configure
Products"),"_self"), NULL)
```

You can provide multiple contract numbers separated by ampersand (&). If the product is found in contract number 23456789, pricing will be applied from ABC New York Price List. If not, CPQ looks for another contract number 12345678. If pricing is found in contract number 12345678, pricing will be applied from ABC California Price List. If not, pricing will be applicable from the default price list.

Configuring Contract Pricing through Price Agreements

CPQ can store pricing information when users accept a proposal or some other event occurs, as per client requirement. CPQ can retrieve the same pricing information when the same user creates a proposal for the second time or updates the same price list in case of agreement amendment. Many customers prefer old pricing during renewals or amendments.

When the user creates a quote with Intent = Price Agreement, finalizes the quote, and accepts it, CPQ automatically creates a contract price list with the products from the quote, according to the prices negotiated in the quote. The contract price list is a subset of the standard price list. CPQ assigns a contract number to the new price list generated. The user can then use the contract number and pass it to the future quotes for using the contract pricing feature. The prices in the new quote will be retrieved from contract price list and those prices are the negotiated prices of the original quote.

For more information, see Creating Quotes from Opportunities.

Configuring Pricing Profiles and Pricing Batch Size

Pricing profiles and batching price calculations enable you to specify the type of pricing configured so that CPQ can optimize run time price calculations.

The following two custom settings control this behavior:

- **Pricing Profile**
- **Pricing Batch Size**

You can specify the **Pricing Profile** as *Basic* or *Advanced*.

By definition, a pricing profile is *Basic*, if:

- There are no pricing rules used
- There are no price matrices used
- There are no related pricing setup
- There are no bundles

At present, CPQ supports only two pricing profiles: *Basic* and *Advanced*. If the Pricing Profile field is left blank, the default value is *Advanced*. If the Pricing Profile is *Basic* and **Defer Pricing** check box is selected, the performance can be improved by executing price calculations in batches. To enable and control batching of price calculations, use the Pricing Batch Size field.

The Pricing Batch Size setting defines the number of line items that can be processed in a single pricing call. Setting the Pricing Batch Size, the system runs pricing with the specified number of products as a batch, thus increasing performance. These batch calls to the database is governed by Salesforce CPU time limit and hence the number assigned for Pricing Batch Size has to be carefully evaluated.

> These two system properties are global and affect the entire CPQ implementation in an Org. You must ensure that all the products have only Basic pricing configured as defined above.

To set up pricing profile and pricing batch size

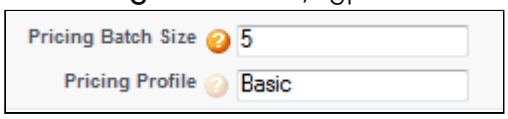
1. In **CPQ Console**, go to **Setup** > **Develop** > **Custom Settings**.
2. Click **Manage** for **Config System Properties**, and click **Edit**.
3. Select **Defer Pricing**.

   > To use **PricingProfile** and **PricingBatchSize**, **DeferPricing** must always be selected.

4. For **Pricing Profile**, type *Basic* or *Advanced*.

   > If this field is left blank, the default value is Advanced.

5. For **Pricing Batch Size**, type a value.



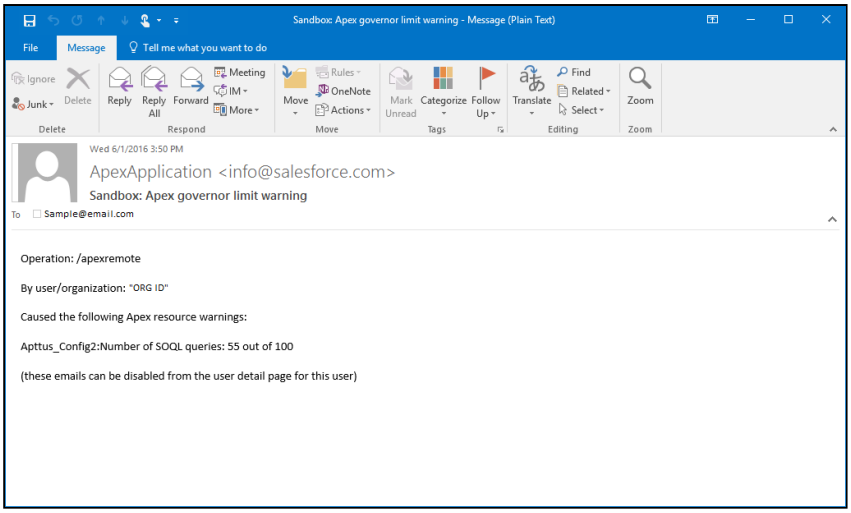
> If this field is left blank, the default value is 1.

6. Click **Save**.

Pricing is performed as defined in Pricing Profiles and batches as defined in Pricing Batch Size.

Configuring Email Notifications for Apex Governor Limit Warning

When CPQ executes any action (such as pricing or configuration), your users may receive an email notification indicating that they have reached a certain lower threshold for the Apex governor limit. This is a normal behavior and CPQ is built with these limitations in mind with a certain degree of buffer for further actions. As an Admin, you can disable the email notifications from the user detail page for a specific user.



Managing Cost and Profitability (Cost Breakdown)

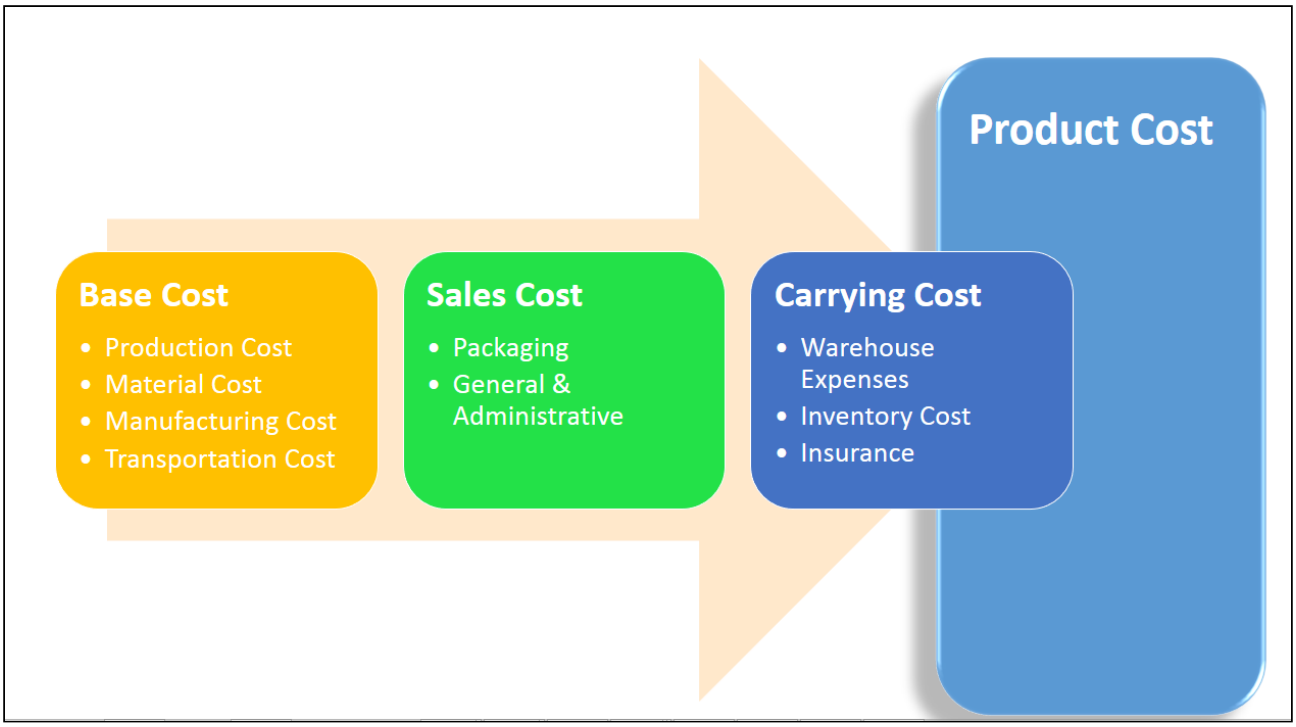
Cost of a product, as you see it, is derived from aggregating direct, indirect and variable costs. These costs could be manufacturing, Sales, Administrative, volume-based or any other miscellaneous cost.
For example, the Cost of a Drilling Machine is an amalgamation of costs from various

sources incurred during the life cycle of the product - such as raw material, production, labor, time, taxes and other marginal costs. While some of these costs remain stable, some vary depending on the volume, availability or other market conditions.

To cite another example, the price of a watch is derived from the summation of the cost of designing, manufacturing, packaging and shipping. Similarly, if your product is a service, cost can be derived from several dependent factors such as infrastructure, security, fulfillment and others. In a nutshell, for all products, such finer cost elements that are associated with the product life-cycle, together constitute the final cost.

As a Sales Rep, when you are adding products to the cart during the Quote generation process, adding margins and reaching the profitability of the deal will become easy if you can see the cost elements which is precisely the cost break-down of a product.
Visibility of the cost hierarchy for a product that captures every granular cost will empower you to prepare a profitable Quote that leaves no room for generalization and approximation.

When you know the key cost drivers and can see how what other costs contribute to the key cost, you know which costs should be tweaked or modified for arriving at an optimum cost.



The cost models, as you can see in the above illustration, are nothing but the structure of cost elements and sub-elements participating in deriving the final cost. The cost hierarchy gives you the granular level of details regarding the various aspects of product costing.
With CPQ, you can create complex Cost Models that give insights into the true profitability

of a product price. During the quotation process, this transparency of costs can provide accurate deal margin decisions that are critical for thriving in the competitive market.

Start implementing the cost model for your products to

- Achieve Effective Deal Guidance,
- Eliminate the need for an external Cost Management application, and
- See the price breakdown and assess the profitability of your quote.

Key Terms

Term	Description
Base Price	The basic product price without break-ups.
Mark-up	Additional cost incurred for a product to create a profit.
Fixed Costs	Costs that do not vary for a long period of time are considered as Fixed Costs. For example, Maintenance Costs, Renting or leasing of assets, Property Insurance etc.
Variable Costs	Costs that are volume-driven or change frequently with time are regarded as being variable. For example, raw materials, labor, social expenses, training etc.

There are three key facets of this feature:

- **Price breakdown**
  Earlier you could associate only a single cost to the product. Because the cost is dependent on various factors some of which might vary, a single field defining the cost was very limiting. With cost models in place, you can define the variable and fixed cost elements and achieve dynamic product costing. This approach is the best way to ascertain what a product should be priced at.
  For more information on how you can configure and store the granular costs associated with your product, see Storing the Price of Products.
- **Transfer Price**
  Setting the price for goods or a product between legal entities within an enterprise is known as Transfer Pricing.
  The transfer price is the price at which different goods and services are transferred between divisions within an organization. For example, a division of Motor Company-A, based in South Africa supplies Gears to their global assembly plant in Boston. The division in South Africa will charge the division in South Africa with the cost of the Gear including their profit margin. This price should be an amount that benefits the

organization and also allows each department to have a fair share of their profits. Majorly, a Business Unit or Factory "sells" to the front end sales team. There are regulatory requirements around how the transfer price is calculated. OECD (Organization of Economic Cooperation and Development) has set some guidelines and most countries follow or add-on to those guidelines. For more information on how you can create a Transfer Price mapping, see Configuring Transfer Pricing.

- **Price Waterfall**
  You can Analyze a Quote by looking at the Price Waterfall chart and Waterfall table for corresponding Line Items and Summary groups. The waterfall chart is a bar graph where Line Item Fields are plotted vertically and Cost is plotted horizontally. For more information, Analyzing Quotes.

We'll take a detailed look at each of these aspects in the later sections.

Required Configurations

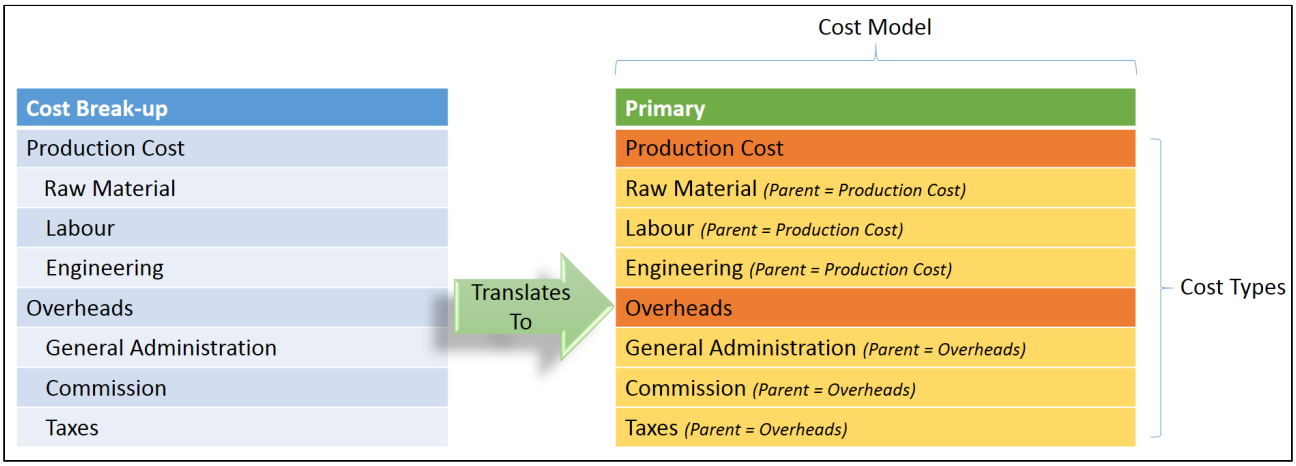
A way to arrive at accurate costing is to dissect the cost into different elements, roll-up this cost and apply markups to finalize it. These elements are identified as Cost Types in the system.

Let us see how you can start creating a cost hierarchy for your products and analyze the margins on the cart. To get started, you need to:

1. Define a **Cost Model**.
   a. Set a Cost Hierarchy by adding **Cost Types**.
2. Associate the Cost Model with a Price List.
3. Configure a Presto App (using X-Author) to store product prices.
4. Fetch prices for the Cost Types by adding a Presto Callback.
5. Analyze pricing from **Analyze Quote** button on the cart.

Defining Cost Models

Cost Model is a container holding all Cost Types. As shown in the diagram, you can create *n* Cost Types for *n* number of Cost elements.



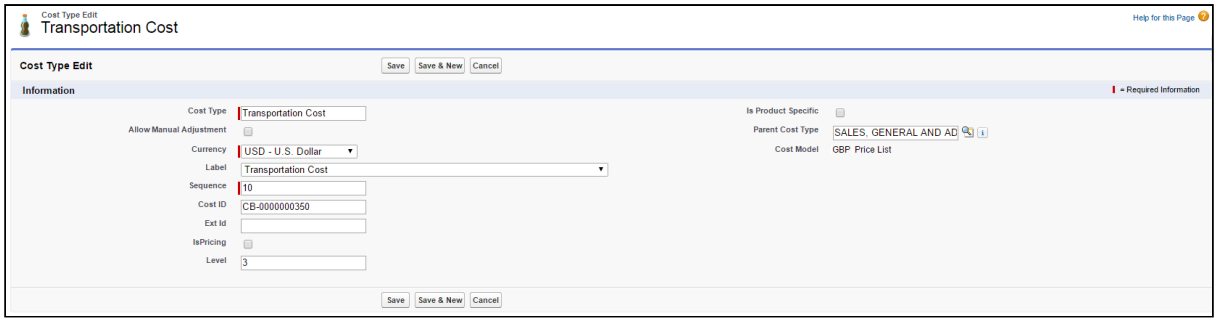
1. In **CPQ Console**, go to **+ (All Tabs)** > select **Cost Models**.
2. Click **New**. The New Cost Model page is displayed.
3. In the **Cost Model** field, enter a name for the cost model.
4. Click **Save**.

Your cost model is created and you can start adding cost types.

Define Cost Types

Cost Types correspond to individual cost entities. You can define *parent* for a Cost Type thereby creating Cost Type hierarchies.

1. From the Cost Types related list, click **New Cost Type**.



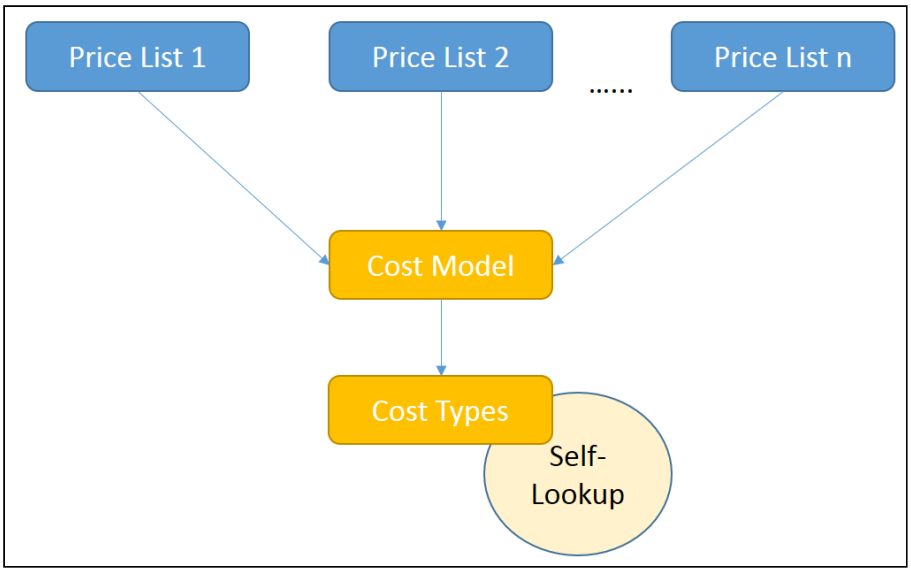
2. Enter the following details and click **Save**:

Field	Description
Cost Type	The name for a Cost Type.
Label	A Label differentiating the Cost Type.
Sequence	The order for arranging hierarchies sequentially.
Parent Cost Type	Choose a Parent Cost Type for the said Cost Type.

Field	Description
Cost Model	Read-only field hat indicates the Cost Model associated to this Cost Type.
Allow Manual Adjustment	If selected, allows you to override costs on the **Analyze Quote** page. On clicking Reprice, the base price is recalculated with the new costs.
Currency	The Currency specific to this Cost Type.
Hide Child Cost Types in Price Waterfall	If selected, allows you to hide a cost category and its children from being displayed on the Analyze Quote page.

Associating Cost Models with a Price List

You can associate one cost model with multiple price lists. However, one price list can have only a single cost model associated with it.



To associate a cost model with a price list

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Lists**.
3. Select the required price list

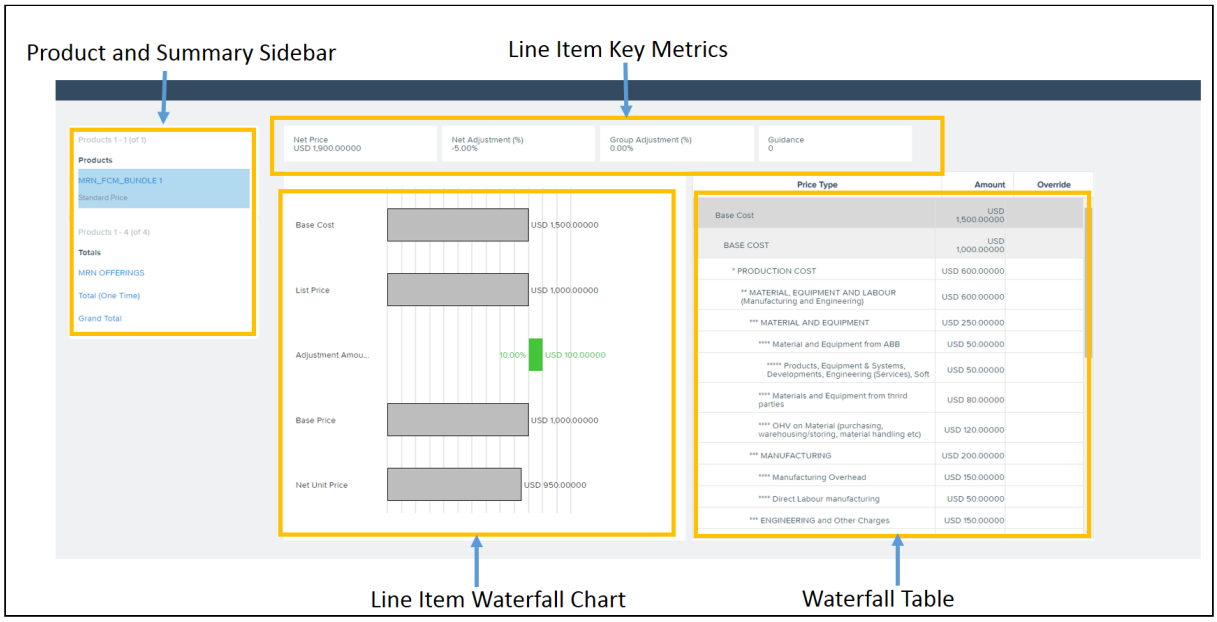
4. In the Cost Model field, search and select a cost model of your choice.
5. Click **Save**.

Configuring the Analyze Quote Page

Click **Analyze Quote** button on the cart, to load the **Analyze Quote** page.

Analyze Quote page is divided into following four components:

1. Product and Summary Sidebar
2. Line Item Key Metrics
3. Line Item Waterfall Chart
4. Waterfall Table



You can opt to display the fields of your choice in the **Line Item Key Metrics** and **Line Item Waterfall Chart** screen elements for both **Line Items** and **Summary Group options**.

Go to **Config Settings** > **Display Column Settings** and select **Display Type** as *Line Item Key Metrics* for configuring fields displayed on top of the page.

Similarly, you can configure the display for *Line Item Waterfall*, *Summary Group Key Metrics*, *Summary Group Waterfall* and *Optional Cost and Profitability Fields*.

All the fields of Cost Line Item, Adjustment Line Item, and Cost Adjustments are available to be displayed on the Analyze Quote page.

Storing the Price of Products

The Price for each Cost Type is defined in an Excel App. You have to replicate the Cost Model, defining various Cost Types and their pricing with the help of X-Author Designer. When you click Configure button on the Quote/Proposal, pricing data from this app is fetched and shown for analysis.

To get started, you will need to Install **X-Author** and **Designer** and **Runtime** packages.

App Setup

Install the following packages:

- X-Author for Excel (3.25 or higher)
- Apttus X-Author™ For Excel Document Generation (1.0)
- Apttus Document Generation (1.5)
- X-Author Designer for Excel

Getting Started

1. In your Salesforce org, create a **Cost Model** and add Cost Types.
2. Open X-Author Designer and create a pricing app.
3. For information on how to create an app, see Creating Apps with Quick Apps.
4. Ensure the Cost Type Label and Cost ID matches with the Cost Names specified in the org.
5. On the App, add Parent Code (from the Salesforce URL) for the child Cost Items.
6. **Validate** and **Save** the App.
   Go to your Salesforce org and access Apps tab to see your app.
7. Edit the app and select **Activated** to activate it.
8. Create a **Price List** and associate it with the Cost Model.
9. In order to run this App while configuring products, pass the App Id in the 'prestoAppId' field in the following formula.

```
"IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0, HYPERLINK("/apex/
Apttus_QPConfig__ProposalConfiguration?id=" &Id
+"&flow=NGDefault"&"&useAdvancedApproval=true&useDealOptimizer=true"&
"&prestoAppId=a5t2C000000GnBe",
IMAGE("/resource/Apttus_QPConfig__Button_Configure", "Configure Products"),"_se
lf"), NULL)"
```

10. Go to **+ (All Tabs)** > **Apps**, open your app and copy the App Id from the URL.
11. Create a Quote, add the product to the cart and click Reprice. To see the price breakdown Click **Analyze Quote** button.

Configuring Transfer Pricing

Transfer Pricing is the price at which two related parties perform business transactions between themselves. Every organization follows a regulatory guideline specific to their country to derive transfer prices for the products. This formula states one of the ways in which Transfer Pricing is calculated.

Transfer Pricing = **Full Cost** (sum of all fixed and variable costs) + **Markups**

Let's assume the Sales Rep exposes a currency field 'Manufacturing Cost' and sends collaboration request to another Factory user.

Cost Model available with the Sales Rep

Cost Name	Value
Packaging	$100
Transportation	$50
Manufacturing	-

Cost Model with the Factory User

Cost Name	Value
Manufacturing	$70 (Cost of Raw Material + Labor)
Raw Material	$30
Labor	$40

The administrator needs to create a Transfer Map so that Sales Reps can send out a collaboration request. For this requirement, the Transfer Price Map would look as the following:

Field	Value
Source Field	Apttus_Config2__Manufacturing__c
Source Charge Type	Standard Price
Target Cost Model	Factory Cost Model
Target Field	Requested_ManufacturingNetPrice__c

Field	Value
Target Object	Apttus_Config2__LineItem__c

Configuration

1. In **CPQ Console**, go to **+ (All Tabs)** > select **Cost Models**.
2. Select the required cost model.
3. From the Transfer Prices related list, click **New Transfer Price** to create a Transfer Price mapping.
4. Enter the following details and click **Save**.

Field	Description
Source Field	The *API Name* of the field for which you want the pricing details.
Source Charge Type	Mention the Charge Type of the Source Field, whether it is a Standard Price, License Fee, Subscription Fee or any other.
Target Cost Model	Enter the name of the destination Cost Model from which you want to get the price of a field.
Target Field	Enter the *API Name* of the field as given in the Target Cost Model.
Target Object	Enter the *API Name* of the Target Object which the Target Field is a part of. For example, if the Target Field is 'Net Unit Price' and is defined on the Line Item object, the Target Object will be Apttus_Config__LineItem__c.

Procedure

1. Create a new Quote using price list.
2. Click Configure Products button. Add products and go to Cart.
3. Using the collaborator icon send the request to Factory User.

4. Factory User receives the Collaboration Request. Applies adjustments or enters the price and clicks **Submit For Merge**. Please refer Assigning the configuration to the Collaborator or Queue for details.
5. The Transfer price is set to copy **Net Price** from Collaborator's Quote to a cost type named "Cost From Factory" of requester's Quote.
6. Open the requester's Quote. Go to Cart and click **Merge** in the Collaborator pop-up.
7. Click **Analyze Quote** to analyze the product cost.

Configuring Unit of Measure (UOM) and Frequency Conversion Rate

Unit of measure (UOM) is a way to quantify resources. It is a key factor in pricing and discounting the products on the cart. With the new object called **Frequency/UOM conversion rate**, you can define a set of unit of measures for your product families. Products under the same family can be priced differently, based on its unit of measure. Depending on the available unit of measures for the product family, the sales reps can define different discounts for different UOMs for the same product.

The current design requires all products under the same product family to share the same UOM conversion rates. So an implementation tip is to use product category to group products by business needs, but use product family to group products that share the same conversion rates. For example, different types of coffee can be grouped under a single category called Coffee, but if you want to apply the same conversion rate to dark roast and mild roast coffee products, you can create a new product family called Roast Coffee.

Frequency/UOM conversion rate is a new object, along with fields such as **Conversion Factor** and **Type**. Depending on the type of conversion type, you must define the field values for either the Frequency conversion or UOM conversion. You can also control whether to apply these conversions on a product family and product category.

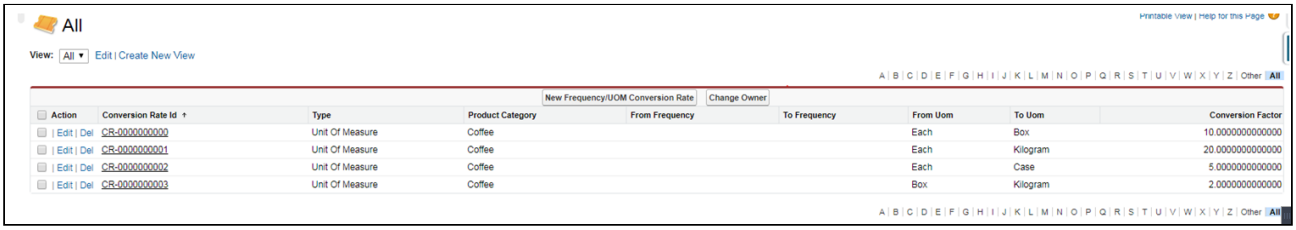
You can only define UOMs and UOM conversion rates for product families; one product can exist in multiple categories, but only in one product family.

Use case: Creating UOM and conversion rate for products

Description: This use case describes how to create frequency conversion rates and unit of measurements for pricing the products on the cart. You might use this functionality differently, depending on your business case.

Suppose you are a coffee vendor and want to sell your product by considering the customer requirements in terms of kilograms and boxes. As a system administrator, you can set up the type of measurement as kilogram and box, and define the conversion rate for these measurements.

In the following example, the sales administrator creates a conversion type for the coffee product.



The above setup represents the following UOM conversion rates for the product family Coffee

1 Box = 10 Each

1 Kilogram = 20 Each

1 Case = 5 Each

1 Kilogram = 2 Box

Pricing admin can create product list for any products under the product family Coffee and use above UOMs captured in the UOM conversion table.

For example, the pricing admin can create prices list in the following UOMs:

Table 1

Product	Price	UOM
Dark Roast	$10	Each
Dark Roast	$95	Box
Dark Roast	$50	Case
Medium Roast	$10	Each
Medium Roast	$90	Box
Medium Roast	$45	Case

And the sales rep can create the following manual adjustments on the cart:

Table 2

Product	Discount	UOM
Dark Roast	$2 off	Each
Dark Roast	$23 off	Box
Dark Roast	$12 off	Case
Medium Roast	5% off	Each
Medium Roast	8% off	Box
Medium Roast	5% off	Case

In the cart, the pricing engine converts the quantities from the Pricing UOM to the Selling UOM and the discounting UOM to the Selling UOM.

To create a Frequency Conversion Rate

Perform the following steps to create a new record of frequency conversion rate:

1. In **CPQ Console**, click ⊞ (All Tabs) and search for **Frequency/UOM Conversion Rates**.
2. Click **Frequency/UOM Conversion Rate** tab and click **New** to create a new record.
3. The record is divided into 4 sections and the fields in each section are explained below:

Information	
Field	**Description**
Type	Defines the currency metric type. Available values are *Frequency* and *Unit Of Measure*. Depending on your selection, you must enter the values in either Frequency or UOM Conversion table.
Product Family	Defines the Product family to which you want to apply the conversion rate. This helps you filter the group of products on which you want to apply the conversion rate. Ensure that you have created a product family record so as to see it populated in this picklist.

Information	
Field	**Description**
Product Category	Defines the Product category to which you want to apply the conversion rate. This helps you filter the group of products on which you want to apply the conversion rate. Ensure that you have created a product category record so as to see it populated in this picklist. In our example, the product category is *Coffee*.

Frequency Conversion > applicable for Type = *Frequency*	
Field	**Description**
From Frequency	Defines the starting frequency for a conversion rate from which the rates are valid.
To Frequency	Defines the ending frequency for a conversion rate until which the rates are valid.

UOM Conversion > applicable for Type = *Unit of Measure*	
Field	**Description**
From UOM	Defines a unit of measure that can be used as a source for rate conversion. For example, if you want to measure a coffee box in terms of Box to Kilogram conversion, use *Box* in **From UOM** and *Kilogram* in **To UOM** field.
To UOM	Defines a unit of measure that can be used as a destination for rate conversion.

Rate	
Field	**Description**
Conversion Rate	Defines the rate at which you want to convert the product. In our example, for UOM conversion from Box to Kilogram for 1 box = 2 kilograms, enter 2 in this field.

4. Click **Save**.
   You must create these records for each UOM that you want to configure. In our example, you must create one record for each row of Table 1.

Configuring the Currency Rounding Mode

This section provides information on configuring currency rounding in pricing calculations. This feature rounds adjustments before calculating the base price.

Prerequisites

Enable Custom Rounding must be True. For more information, see *Config System Properties* in Configuring Custom Settings.

To configure currency rounding

Perform the following steps to configure currency rounding.

1. In **CPQ Console**, go to **Setup** > **Develop** > **Custom Settings** > **Config System Properties** > **Manage**.
2. Click **Edit** next to **System Properties**.
3. In the **Rounding Mode** field, enter one of the following values.

Value	Description
UP	Rounds the currency to the next number. For example, 21.2 is rounded to 22.
DOWN	Rounds the currency to the previous number. For example, 21.8 is rounded to 21.

Value	Description
HALF_UP	Rounds the currency to the next number if the decimal is equal to or greater than 5. For example, 21.5 to 21.9 is rounded to 22.
HALF_DOWN	Rounds the currency to the previous number if the decimal is equal to or smaller than 5. For example, 21.1 to 21.5 is rounded to 22.
HALF_EVEN	Rounds the currency to the nearest even number. For example, 23.5 is rounded to 24 and 22.5 is rounded to 22.

4. Click **Save**.

Managing Currency Conversion

You can set the exchange rate between the currency of your organization and another currency for a certain period.

> If the Sales Representative uses the conversion rate on a date that is not within the defined period for currency exchange rate, CPQ displays an error message on the Cart.

Prerequisites

You must set the value of the **useAdvancedCurrency** parameter to *True* to enable dated currency conversion.

To manage currency conversion

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Currency Conversion**. The Dated Currency Exchange Rates page with a list of existing exchange rates is displayed.
3. Click **Add New Conversion**. A new item is added at the end of the list.
4. Enter the following details:

> If the exchange rate is already available between the same currencies and you enter the exchange rates between the same currencies for the same period, CPQ displays an error message.

Field	Description
Currency From	CPQ selects the source currency automatically based on the currency you have configured for your organization.
Currency To	Select the target currency to which you want to convert the source currency.
Exchange Rate	Enter the exchange rate between two currencies.
Rate Type	Select the rate type for currency conversion. Currently, Period End Date is supported.
Start Date	Select the start date from when the currency conversion must be effective.
End Date	CPQ enters the End Date automatically after you provide the Start Date for the next entry for the exchange rate between the same currencies. For example, you provided the first rate between USD and EURO from 01/01/2020. Now when you provide the rate between the same currencies from 01/04/2020, the End Date for the previous entry is taken automatically by as 31/03/2020.

5. Click **Save**.
6. Click the more icon (⋮) and select **+ Add** to add a new conversion or select **X Remove** to delete an existing conversion.

Managing Price Waterfall

Price Waterfall is the industry standard for Price and Margin calculation. Price waterfall defines a sequence to handle different prices with the target of reaching the net price.

> This feature is available only in the TurboPricing flow. For more information, see About TurboPricing.

A well-defined Price Waterfall provides transparency and high visibility of pricing calculation and helps protect margins and avoid incorrect pricing.

Price Waterfall comes with features for transparency and high visibility in pricing calculation and protecting margins, avoiding incorrect pricing. In addition, Price Waterfall defines a sequence to treat different prices with the end goal of arriving at the net price charged to the customer.

Price Pipeline enables administrators to configure the price execution flow and set the groundwork for Conga's Price Waterfall feature to display detailed pricing steps for end-users to visualize. Price waterfall is a tool that provides users detailed information on how much revenue the company is generating from a transaction and the pocket margin of that transaction. The waterfall is typically represented in a column chart consisting of critical price or margin points and any adjustments that were applied on the list price to arrive at the final price or margin.

The advantages of the price waterfall functionalities are as follows:

- Provides Sales Representatives with the ability to drill down on any price point visible to them so that they can make better decisions and better communicate the deal to customers giving them more leverage during a negotiation.
- Provides Approvers and Deal Desk with a complete picture of the different price points, preapproved and non-standard adjustments, and costs to better analyze the health of the deal.
- Provides Cost Accountants and Controllers' ability to drill down on the cost (if cost is included as part of the Price Pipeline) to ensure that the true profitability is calculated for the deal to ensure the company's bottom line goals are met.

Configuring Price Waterfall

To configure price waterfall, perform the following steps:

Step	Task	Description
1	Define Pricepoints	Create new price points as picklist values in the "Price Points" field in the Price Rule object in salesforce so that these price points can be used when defining any price pipeline.
2	Configure Price Pipeline	Define price points and configure price pipelines.

Step	Task	Description
3	Configure a Price Waterfall to a Price Pipeline	Configure a price waterfall to a price pipeline.

The following chapters describe the steps to configure price waterfall:

- Defining Price Points
- Configuring a Price Pipeline
- Configuring Price Pipeline Ruleset

Defining Price Points

For creating new price points, Admin must add them as new picklist values in the "Price Points" field in the Price Rule Object in Salesforce.

To add new price points as picklist values:

1. Go to **Setup** > **App Setup** > **Create** > **Manage your custom objects** > **Price Rule**.
2. Under the **Custom Fields & Relationships** section, click **Price Points**.
3. Under the **Values** section, click **New.**
4. Enter new price points in the textbox and click **Save.**

Configuring a Price Pipeline

A price pipeline is a sequence of price points.

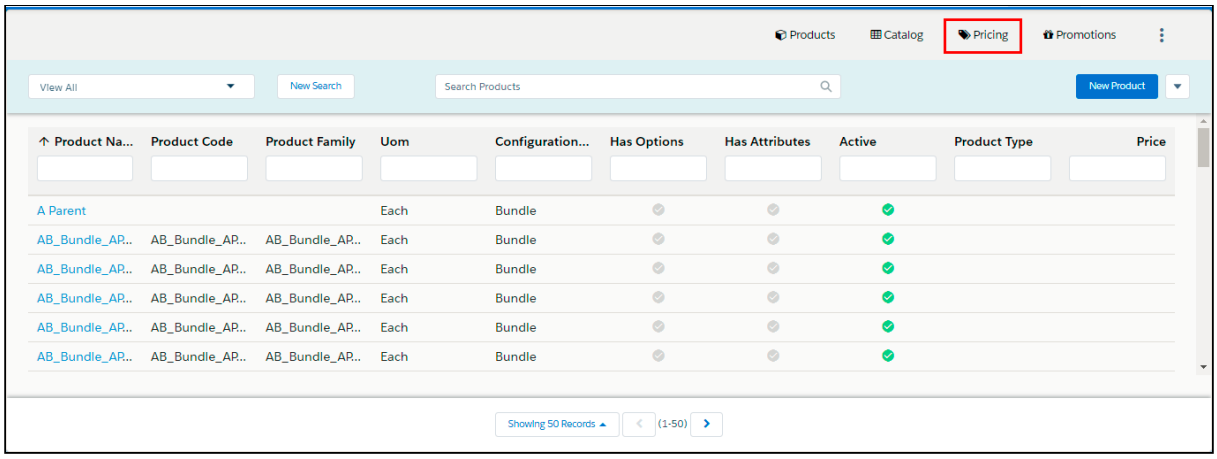
You can add multiple price points in a price pipeline definition, arrange them in a sequence to define the price pipeline, and map one price as net price. Base Price and List Price are mandatory price points, and hence those will already be added to every price pipeline by default. When the pricing engine is pricing a line item, it calculates the per-unit amount associated with each price point and the gross amount (amount incorporating the quantity and term) if the transaction is a quote or order. The price point definition describes how it should be calculated and whether it is treated specially (for example, identifying a particular price point as the Net Price point for integration with downstream systems such as billing.)

Pricing administrators or CPQ administrators define price points and configure price pipelines. Price Pipeline definition is further used in the Price Waterfall chart to determine

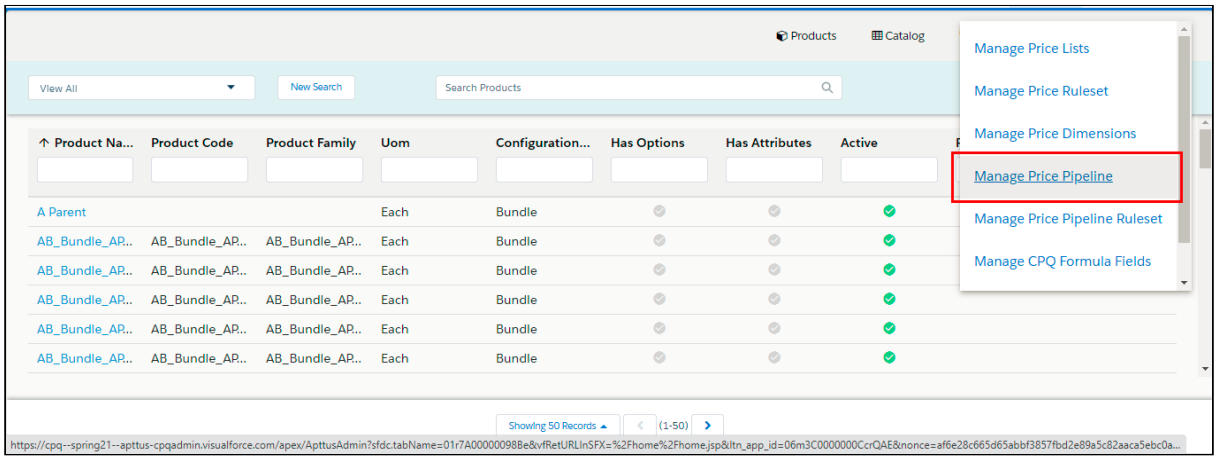
whether certain features (such as manual adjustment to any price point) are allowed to the end-user or not. For more information, refer to Configuring the Pricing Engine.

To add a price pipeline:

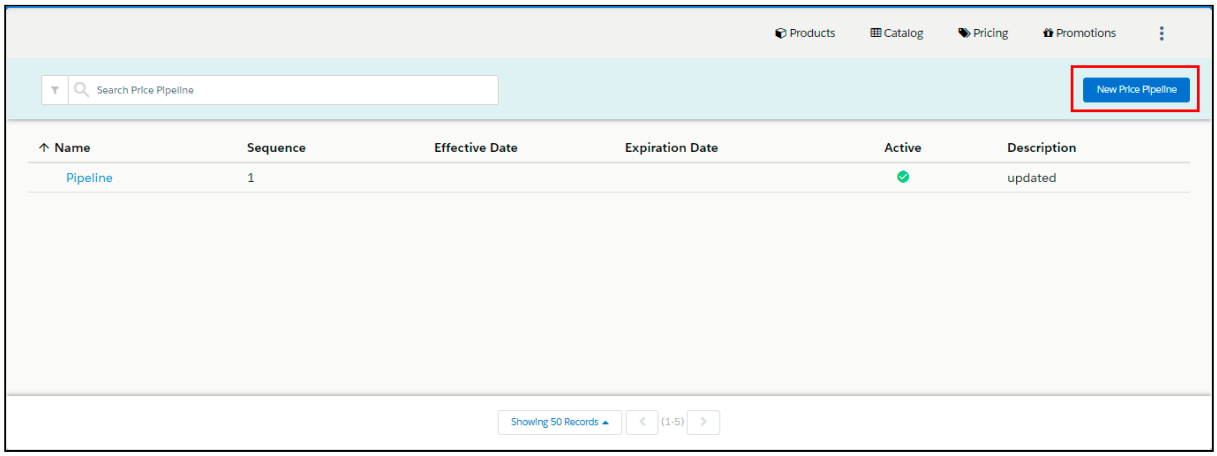
1. On **App Menu**, click **Apttus CPQ Admin** and then click the **Admin UI** tab. The new admin console is launched.



2. Select **Pricing** > **Manage Price Pipeline**.



The Manage Price Pipeline page is displayed.



3. Click **New Price Pipeline** and update the following fields:

Field Name	Description
Price Pipeline Name	Name of the price pipeline
Sequence	The sequence allows you to determine the correct price pipeline to apply if multiple price pipelines match the criteria for a line item. The lowest matching sequence will have the highest priority.
Is Active	Slide the slider to enable the price pipeline
Description	Description of the price pipeline
Effective Date	Enter a date to associate an effective start date with the price pipeline
Expiration Date	Enter a date to associate an effective end date with the price pipeline

Field Name	Description
Pipeline Criteria	Define the criteria for the price pipeline. The price pipeline is used based on the criteria defined by the user. If the given criteria are not met for any line item, then the price waterfall chart is not created. Define the criteria for Line Items and Product Attributes as follows. • For Line Items, click the icon > Line Items > Add New Criteria > enter the desired Field (Select the field that must be used for the adjustment), Operator, and Value > Save. • For Product Attributes, click the icon > Product Attributes > Add New Criteria > enter the desired Field (Select the field that must be used for the adjustment), Operator, and Value > Save.
Define Price Points:	Define price points as follows. List price and Base Price are set by default. a. Click the text box in the name column and select the desired price point from the list. If the price point is not available, create a new price point. b. Is Modifiable: This allows you to modify the price point on the Waterfall page. For example, If you enable the Is Modifiable option for a price point such as *Invoice Price*, the sales representative can only modify the *Invoice Price* of the product by adding manual adjustment to it on the Price Waterfall chart. c. Is this Net Price: This allows you to make a price point as a net price. Move the slider to enable the option. Any manual adjustments that the user performs on the cart appear on the waterfall chart against this price point (which is marked as the net price). d. Click the '+' icon to add new price points.

4. Click **Save.**

ⓘ You must set at least one Price Point as a Net Price.

To Modify a Price Pipeline

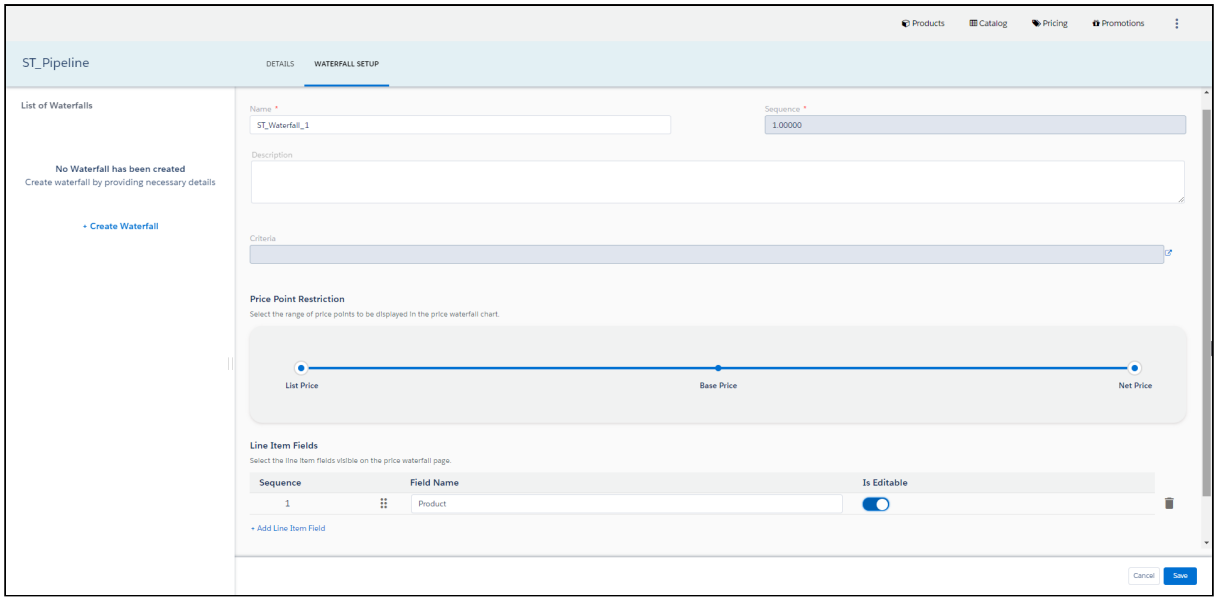
1. On **App Menu**, click **Apttus CPQ Admin** and then click the **Admin UI** tab. The new admin console is launched.
2. Select **Pricing** > **Manage Price Pipeline**.
3. Click the required Price Pipeline > Modify the details and click **Save.**

Configuring Price Waterfall to a Price Pipeline

After configuring the price pipeline, you must configure the price waterfall to a specific price pipeline.

To configure a price waterfall:

1. In the **App Menu**, click **Apttus CPQ Admin** and then click the **Admin UI** tab. The new admin console is launched.
2. Select **Pricing** > **Manage Price Pipeline**.
3. Click the required Price Pipeline > Click the **WATERFALL SETUP** tab.



4. Update the following details and click Save.

Field	Description
Name	Name of the price waterfall.
Description	Provide a description.
Criteria	Allows you to set criteria for a line item rule or a bundle, depending on the Application level you have selected, such that the ruleset only applies when it satisfies a line-level field value or a product attribute value. Click the icon to fill in your criteria.

Field	Description
Price Point Restriction	Set the visibility restriction on price points. This option allows you to control the visibility of price points for the user. 
Cart Line Items Field	a. Click the text box in the name column and select the desired Line Item field from the list. b. Is Editable: This enables you to modify the price point on the *Waterfall* page. For example, if you enable this option for the cart line item *Quantity,* then the sales representative can modify the quantity of the product on the waterfall chart and apply it to the price. Move the slider to enable the Is Editable option. c. Click the '+' icon to add new field names. d. Click **Save.**

Configuring Price Pipeline Ruleset

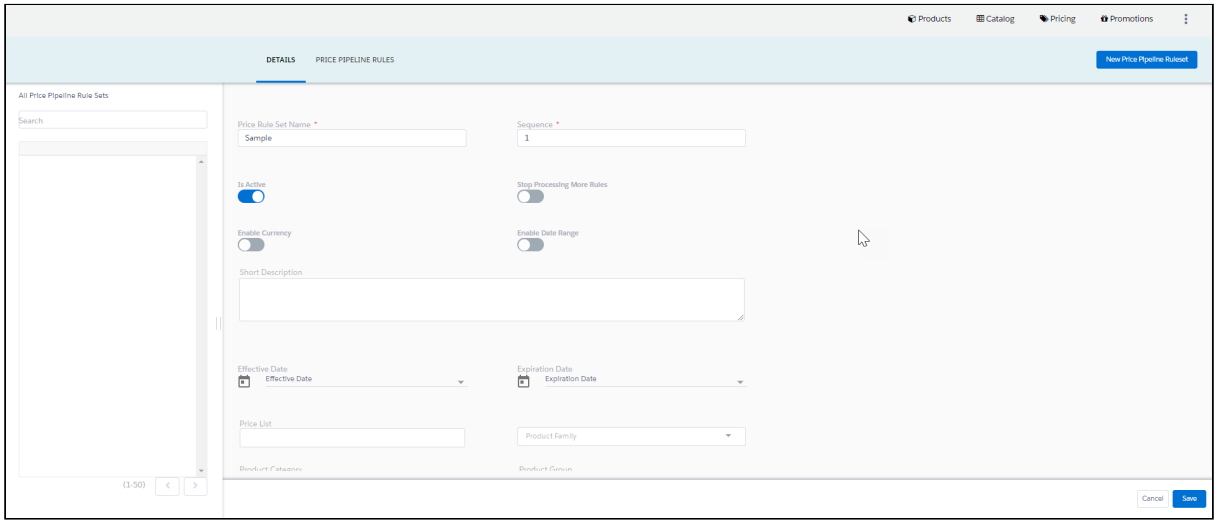
Price Pipeline Ruleset allows pricing adjustments across a range of products through price rules. Price Pipeline Ruleset manages adjustments to specific price points, based on particular line item criteria or product attribute criteria. Price Pipeline rulesets are very similar to standard price rulesets, except that these contain price pipeline rules instead of the standard price rules. A price pipeline ruleset can contain one or more price pipeline rule entries and it can be dimensional, which employs a similar concept as Price Matrices. You can apply line item pricing adjustments as well as summary pricing adjustments.

To create a price pipeline ruleset

You must have an existing price list.

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin UI** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Pipeline Ruleset**.

3. Click **New Price Pipeline Ruleset**.



4. Enter details in one or more of the following fields, as required:

Field	Description
Price Pipeline Ruleset Name	Enter a mandatory price pipeline ruleset name and a mandatory sequence in which the system will evaluate multiple rulesets. Typically, you will perform line item adjustments first and then any summary adjustments
Is Active	Select this to set the ruleset as active.
Enable Currency	Select this to enable currency for the Price Pipeline Ruleset.
Currency	Select the currency for the Price Pipeline Ruleset. ⓘ If the Price Pipeline Ruleset and a line item have different currencies, CPQ applies the currency conversion rate (defined at **Setup** > **Company Profile** > **Manage Currencies**) when applying the rule adjustment.
Short Description	Enter a description for the ruleset.
Effective Date	Not required as long as the Price Pipeline Ruleset is Active, but can be used for promotional rules, and more.

Field	Description
Expiration Date	Select an expiration date.
Price List	Select a price list. The ruleset will only source product prices with the selected price list and then apply adjustment criteria.
Product Category	Select a category. The ruleset will only source product prices within the selected category.
Category	Select All, Agreement, or Proposal. This indicates if the ruleset is relevant to Agreements, Proposals, or both.
Application Level	Select a level to which this ruleset will be applied. The supported values are: • *Line Item*: Selecting this will apply the line item adjustment within the line item net price in the shopping cart. • *Bundle*: Selecting this will apply the adjustment to a bundle and its options and the adjustment is displayed in the Totals tab in the shopping cart. • *Aggregate:* Selecting this enables the **Application Method** field.
Application Method	This field is enabled if you selected *Aggregate* from the **Application Level** drop-down list. This indicates that you want the adjustment to select products in aggregate but apply an adjustment as a summary line in the Totals tab or spread the adjustment over numerous products. The supported values are: • *Apply to Line Items*: Applies the adjustment to line items on the cart. • *Create Summary Lines*: Applies the adjustment as a summary line in the Totals tab on the cart.
Enable Date Range	Selecting this enables you to set the effective date and expiration date on the rule entry.
Product Family	Select the Product Family. This is the Product Family field on the products object. The ruleset will only source product prices with the selected Product Family.

Field	Description
Product Group	Click to search and select a custom product group to the ruleset will apply the pricing adjustments. These custom product groups have no relation to a category, a Product Family, or any other product designation.
Charge Type	Select a charge type to which the ruleset will apply adjustments.
Ruleset Criteria	Allows you to set criteria for a line item rule or a bundle, depending on the Application level you have selected, such that the ruleset only applies when it satisfies a line-level field value or a product attribute value. Click the  icon to fill in your criteria.

5. Click **Save**.

By filling out these criteria, the source products and prices are then designated and the ruleset can apply adjustments through price rules.

Creating Price Pipeline Rules

A *Price Pipeline Rule* enables you to provide target adjustments to specific price points in the price waterfall chart. Although the standard price rules are applied to the *Base Price*, you can now target any custom price point that you want by using the price pipeline rule. When defining any price pipeline rule, you must associate at least one price point to a price pipeline rule. These rules are will be executed (subject to fulfilling other criteria) and shown as adjustments specific to only those price points in the price waterfall chart

You can associate multiple pipeline rules to a price pipeline ruleset or through to a product. Price matrices can be defined in price rules as well as conditional price pipeline rules (rules that use Boolean logic such as AND/OR). The price pipeline rule can be dimensional, which uses a similar concept to Price Matrices or Conditional, that relates to fields for pricing adjustments.

One ruleset can have multiple rules and is evaluated in order of the rule sequence. A rule can be dimensional or conditional. A price pipeline rule determines the actual price adjustment made.

To create a price pipeline rule

You must have an existing Price Ruleset.

1. In the **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.

2. On the **Pricing** menu, click **Manage Price Ruleset**.
3. Select a price ruleset.
4. On the **PRICE PIPELINE RULES** tab click **New Price Price Pipeline Rule**.
5. On the **DETAILS** tab, enter the following details.
    a. In **Rule Name**, enter a mandatory rule name.
    b. In **Short Description**, enter a description for the rule.
    c. Click **Target Price Points** and select the price point from the list to associate with the rule.
    d. From **Rule Type**, select a type. The supported values are *Dimension* and *Condition*.
    e. From **Adjustment Applies To**, select an option to which the adjustment is applied. This is for line-item adjustments only. The supported values are List Price, Previous, and Prior Price Point.

Value	Description
List price	The source price for adjustment calculation is the List Price.
Previous	The source price for adjustment calculation is the current running price.
Prior Price Point	The source price for adjustment calculation is the prior price point.

    f. By default, the price pipeline rule inherits the value of **Currency**, from the price ruleset.
    g. The **Sequence** is auto-generated.
    h. To make the rule active, select **Active**.
    i. From **Adjustment Charge Type**, select an option to set adjustment as a charge type. You can change this in the price pipeline rule object.
6. Select the **DIMENSIONS** tab, enter the following details.
    a. From **Dimension**, select a price dimension.

> You can enter up to six dimensions, which bring in attributes from the line items or headers of any data type within Salesforce. For example, you can select a dimension on quantity.

    b. From **Dimension Value Type**, select **Discrete, Range, Cumulative Range,** or **Cumulative Range-Line Item.**
7. Click **Save**. The price pipeline rule is created and listed under the price pipeline rule list.
8. Click **Save**.

Managing Quotes or Proposals

In this section,

Enabling Quote Collaboration in the Org

Quote collaboration feature empowers your sales reps to work together on the same quote. A sales rep starts the configuration process, and assigns configuration request to other sales rep (called collaborator) where he needs help. The sales rep can assign multiple line items to a collaborator. A sales rep can also assign different line items to different collaborators.

The Quote Collaboration feature is useful for a large industrial equipment manufacturer, which has a complex quoting needs. The people who understand the products are located in different factories around the world. When a client requests a quote from a sales rep, the sales rep has to contact different factories and get configuration information. The problem is that the sales rep does not always know who is the right person for the job and does not have all the necessary support and collaboration. This burdens the quoting process by:

- Slowing it down.
- Making it more complex and expensive.
- Making it less accurate.

In such a scenario, quote collaboration enables the sales rep to collaborate while working on a single quote by assigning configuration tasks to different people and orchestrating

that process. Then, the sales rep can merge the collaboration data to generate one final quote and send it to the customer.

For detailed flow and use case around Quote Collaboration, see About Quote Lifecycle Collaboration.

To enable Quote Collaboration in your org

1. Add the **Collaboration** column by navigating to **All Tabs > Config Settings > Display Columns settings.** Choose the **Display Type** as *Cart Line Item* and **Flow** as *ngFlow*. Apart from the desired flow (e.g. ngFlow), the **Collaboration** column must be added to the settings for *Default* flow as well.
2. Specify the following collaborator fields API names in the **Setup > Develop > Custom Settings > Config System Properties > Manage > System Properties > View Cart Custom Fields.**
    - Apttus_Config2__CollaborationRequestId__c
    - Apttus_Config2__CollaborationRequestId__r.Apttus_Config2__Status__c
    - Apttus_Config2__CollaborationRequestId__r.Owner.Name
3. Navigate to **CPQ Console > Maintenance Console > Query Field Metadata** and click **Refresh Field Metadata**.
4. Create Lookup Field Settings for User and Queue assignment. Also, for Queue assignment, ensure that you have listed *Product Configuration* and *Collaboration Request* in the **Supported Objects** list.
5. Ensure that the Salesforce queue which needs to participate in the collaboration process supports **Product Configuration** and **Collaboration Request** objects. This is required to assign a collaboration request to the queue.

- Ensure that you assign appropriate field level permissions for Collaboration Request object and Line Item object.
- If the collaboration icon is not displayed on the Cart, ensure that **Collaboration** field's **Is Editable** property is set to *False* under Display Column Settings > Cart Line Item.

To define Permissions to use Quote Collaboration Feature

The administrator must define proper access permissions for user profiles to use Quote Collaboration. Otherwise, the user cannot see the collaboration feature on the Cart page. Follow the steps below to define access permissions for the user.

1. Go to **Setup> Administrator Setup> Manage Users> Profiles**.
2. Find the profile for which you want to set permissions and click **Edit**.
3. Find **Collaboration Request** under the section **Custom Object Permissions**.
4. Select one or more permission from the list below to define access for the profiles.

Name	Impact on the Cart Page
Read	A disabled collaboration icon is displayed on the Cart page.
Create	The collaboration icon is enabled. The actions fields in the collaboration pop-ups are enabled and the user can assign a line item to a collaborator.
Edit	The **Update** and **Merge** actions are enabled on collaboration pop-up and the user can edit a configuration assigned to them.
Delete	**Cancel** action is enabled on the collaboration pop-up.
Modify All	The **Read**, **Delete**, and **Edit** permissions are selected automatically. The **Update**, **Merge**, and **Cancel** actions are enabled on the collaboration pop-up.

   Actions performed on the collaboration request with the **Modify All** permission have a different impact on the collaboration status as opposed to when the **Modify All** permission is not selected. If **Modify All** is selected, when the user abandons the cart, cancels the collaboration request, or deletes the line item, the collaboration record is deleted. Otherwise, the collaboration status changes. If you do not select any permission, the collaboration icon remains disabled on the Cart page.
   For more information on collaboration status, see About Quote Lifecycle Collaboration.
5. Click **Save**.

To enable Chatter on Collaboration Request

A chatter feed is enabled on each collaboration pop up. The feed is in the context of a single collaboration request and it can be accessed through a new tab on the collaboration pop up.

Ensure that you configure the following steps:

1. Go to **Setup > Build > Customize > Chatter > Feed Tracking**.
2. In the Object section, search for *Collaboration Request*.
3. In the Fields section, select **Enable Feed Tracking**.
4. Click **Save**.

Ensure that you have Quick actions such as Post, File, Link, and Poll on the Chatter Layout. You can verify these actions from **Setup > Build > Chatter > Publisher Layouts**.

To define field set settings

You must define **Collab Create Fields** and **Collab Merge Fields** in Field Set Settings. For more information, see Configuring Field Set Settings.

To setup the Email Templates for Quote Collaboration

In order to send emails to the Sales rep and Collaborator for different actions involved in Quote Collaboration Lifecycle, you must setup the following email templates in your org.

Template Name	Description
Collaboration Task Assigned	When sales rep assigns the configuration to a collaborator, collaborator receives an email. The content of the email is controlled from Collaboration Task Assigned template.
Collaboration Task Completed	When collaborator submits his configuration, sales rep receives an email. The content of the email is controlled from Collaboration Task Completed template.
Collaboration Task Merged	When Sales rep accepts and merges the configuration submitted by the collaborator, collaborator receives an email. The content of the email is controlled from Collaboration Task Merged template.

Template Name	Description
Collaboration Task Updated	After assigning a configuration to the collaborator, when sales rep updates such configuration request, the collaborator receives an updated email. The content of the email is controlled from Collaboration Task Updated template.
Collaboration Request Cancelled	When sales rep updates the configuration request owner and re-submits the same configuration to someone else, the previous collaborator receives a cancellation notification email. The content of the email is controlled from Collaboration Request Cancelled template.

To setup the email templates, navigate to **Setup > Communication Templates > Email Templates** and create separate records for the above-mentioned templates.

Once you have created the above-mentioned templates, you need to create workflow rules for **Collaboration Request** object from **Setup > Build > Create > Workflow & Approvals > Workflow Rules**, which uses the appropriate template to send out the email notification.

Workflow Rule Name	Object	Evaluation Criteria	Rule Criteria
QuoteCollabNotifyOnCreate	Collaboration Request	Evaluate the rule when a record is created, and every time it's edited	`OR (AND (ISCHANGED(Apttus_Config2__Status__c), ISPICKVAL(Apttus_Config2__Status__c, 'Submitted')), AND (ISCHANGED(OwnerId), ISPICKVAL(Apttus_Config2__Status__c, 'Submitted')))`
QuoteCollabNotifyOnUpdate	Collaboration Request	Evaluate the rule when a record is created, and every time it's edited	`AND ( NOT(ISCHANGED(OwnerId)), NOT(ISBLANK(Apttus_Config2__ChildConfigurationId__c)), NOT(ISCHANGED(Apttus_Config2__ChildConfigurationId__c)), ISPICKVAL(Apttus_Config2__Status__c, 'Submitted'), ISCHANGED( LastModifiedDate ) )`

Workflow Rule Name	Object	Evaluation Criteria	Rule Criteria
QuoteCollabNotifyOnComplete	Collaboration Request	Evaluate the rule when a record is created, and every time it's edited	`AND (ISCHANGED(Apttus_Config2__Status__c), ISPICKVAL(Apttus_Config2__Status__c, 'Completed'))`
QuoteCollabNotifyOnAccept	Collaboration Request	Evaluate the rule when a record is created, and every time it's edited	`AND (ISCHANGED(Apttus_Config2__Status__c), ISPICKVAL(Apttus_Config2__Status__c, 'Accepted'))`
QuoteCollabCancelNotificationOnChangeOfOwner	Collaboration Request	Evaluate the rule when a record is created, and every time it's edited	`AND ( ISCHANGED(OwnerId), ISCHANGED( LastModifiedDate ), ISPICKVAL(PRIORVALUE( Apttus_Config2__Status__c ), 'Submitted' ) )`

Workflow Rule Name	Object	Evaluation Criteria	Rule Criteria
QuoteCollabNotifyOnUpdateAfterCompletion	Collaboration Request	Evaluate the rule when a record is created, and every time it's edited	`AND ( NOT(ISCHANGED(OwnerId)), AND( ISCHANGED(Apttus_Config2__Status__c ), ISPICKVAL(Apttus_Config2__Status__c , "Submitted"), OR( ISPICKVAL(PRIORVALUE(Apttus_Config2__Status__c ), "Completed"), ISPICKVAL(PRIORVALUE(Apttus_Config2__Status__c ), "Accepted") ) ), ISCHANGED( LastModifiedDate ) )`

For the corresponding Workflow Rules, create the appropriate email alert action (email template, and the recipient) from the **Workflow Actions** section.

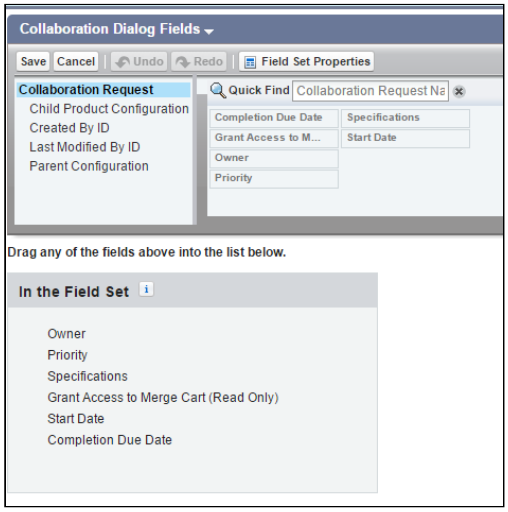
Configuring the Collaboration Dialog Box

You can specify the fields to be displayed on the Collaboration Pop Up dialog box.
This configuration enhancement allows you to configure your own set of relevant fields on the assignment pop-up in order to capture adequate information before the request is sent or assigned to the collaborator to work on.

To configure the collaboration pop up Assignment Details:

1. Navigate to **Setup > Create > Objects > Collaboration Request**. This custom object persists the information for quote collaboration tasks and it is used to relate the parent configurations and its related child configurations.
2. Click **Edit** for **Collaboration Dialog Fields** Field Sets. You can control the fields that are displayed on the Quote Collaboration pop up **Assignment Details** tab.

3. Drag and drop the required fields in the **In the Field Set**.



4. Click **Save**.

You can configure the fields that you see on **Assigned Products** and **Completed Configuration** tabs from **Config Settings > Display Column Settings > Collaboration Parent Line Item** and **Config Settings > Display Column Settings > Collaboration Child Line Item** respectively. These two tabs are enabled after you have initiated the collaboration request with the collaborator.

Assigned Products tab shows the information of the collaboration request details as assigned by the sales rep. Completed Configuration tab shows the information of the collaboration request details as submitted by the collaborator. When you navigate to **Completed Configuration** tab, and click **More details**, you can see the submitted configuration in a read-only mode.

Configuring the Default User or Queue Setup

By configuring default user or queue to be displayed on the collaboration popup dialog box, you can achieve a robust as well as accurate collaboration process, as the request assignment becomes faster and intuitive. The purpose of this feature is to enable enterprise organizations having a plethora of users and queues to filter and default their queues based on the product group, product family, account location and so on. In order for this feature to fulfill its goals and function properly, the right setup has to take place and link the desired user or queue to the filtering logic you want to use.

Ensure that following setup is configured successfully.

1. Set up a user or queue.

2. Set up Location on the quote: This quote location should carry over to the line items on the configuration. you can setup location or any other parameter. make sure that the line items have the location, if you want to filter based on the location.

> **Note**
>
> The field that you wish to base your filtering on, ensure that it is defined (whether it is a product family, product group, account location).

3. Set up Junction object - lookup junction
   - Include all fields that should be referred to. For example Queue ID (you can grab the ID from the URL), account location, product, product family, users, and so on.
   - You need to set this up for each user or queue separately.
   - After this setup, your user or queue is now tied to a location or a product. In order to expose this newly formed link, you need to configure the lookup filter.

> **Note**
>
> Default flag determines if this user or queue is default or not. Specify rest of the fields, you wish to link to your user or queue.

4. Set up **Lookup field settings**. This filter uses the information from the line item and matches it to the lookup junction object information. For example, if the line item's location is France, then all the queues that are defined with a junction object as located in France appears in the collaboration popup dialog box. You can default one of the French queues. In case if there are multiple queues are defaulted, the system only chooses the first one that gets defaulted. Other French queues which do not default are also displayed in the queue drop down and the sales rep can change the selection. Note that the default queue will be pre-selected.

Field	Description
Object Name	Specify the object name from which the lookup relationship is created.

Field	Description
Lookup Field Name	Specify the field name that shows the filtered results.
Display Columns	Specify the API names of fields to be displayed on the lookup dialog box, separated by a comma.
Filter Criteria	Specify the valid expression to further filter down the search results for a lookup.
Default Flag Field	Specify the API name of the specific default flag field on the junction object that stores the information whether the user or queue is default or not.
Source Field	Specify the API name of the field that contains the ID of the field that you are trying to lookup and display on the collaboration popup dialog box.
Source Object	Specify the junction object API name.

Configuring Real-Time Cart Notifications

Sales rep sees a notification popup on the top right side of the screen when collaboration request is submitted, completed, accepted and merged in the parent cart.

To enable this enhancement, after you upgrade to the November 2016 release, you must enter the following piece of code in the Developer Console/Workbench:

```
Apttus_Config2.NotificationFeedService.createPushTopic();
```

ⓘ You need to execute the above code only once and it will apply on all subsequent upgrades, unless you manually delete the PushTopic.

Enabling Cart Locking for Concurrent Access

You can use this feature to lock the cart for all the Sales reps with concurrent access except the one currently editing the cart. The other Sales reps can only view the cart in read-only mode. The cart is unlocked when the Sales rep currently editing the cart saves, finalizes, abandons, or closes the cart, the Sales reps in read-only mode can gain edit access only after these operations are performed. Cart locking helps prevent data loss while a Sales rep is working on a configuration. You can use **Enable Cart Locking for Concurrent Access** in Config System Properties to enable cart locking.

If the cart is left idle by the Sales rep currently in edit mode for a predefined time limit, the cart is unlocked automatically. The cart is considered idle if the Sales rep is not performing changes to the line items, the changes include updating a value, deleting line items, or adding line items. Other actions on the cart such as updating cart views, viewing the totals, or applying filters are not considered modifications to the line items, and time spent is counted as idle time. The time limit is defined in **Cart Edit Access Idle Timeout in Minutes** Config System Properties. After the time limit is exceeded the cart is unlocked allowing any Sales rep can open the cart in edit mode, the cart is locked again for the other Sales reps. If you do not define a time limit the cart remains locked until the Sales rep in edit mode does not save, finalize, abandon, or close the cart.

To enable cart locking for concurrent access and setting a time limit for idle cart

1. Go to **Setup** > **App Setup** > **Develop** > **Custom Settings** > **Config System Properties** > **Manage**
2. Select **Enable Cart Locking for Concurrent Access**.
3. Enter a number of minutes in **Cart Edit Access Idle Timeout in Minutes**.
4. Click **Save**.

The field **Current User** in the Product Configuration object contains the ID of the user accessing the cart in the edit mode.

Enabling PDF Security for Generated Proposal Documents

When you generate a quote in a pdf format, you can Enable PDF Security for the generated proposal document by following the configuration steps below:

To enable pdf security on the generated Quote/ Proposal document

- You must set up *Proposal System Properties* for this feature. Go to **Setup > Develop > Custom Settings**. Click **Manage** next to *Proposal System Properties* list, then click **Edit** next to *System Properties* and do the following:
    a. Select the **Enable PDF Security** check box.
    b. Enter a desired value in the **PDF Owner Password**.
    Refer Custom Settings > Proposal System Properties for detailed information.

1. Click  and click **Admin.**
2. Do any one of the following:
    - Click **New** to create a new Admin entry.
    - Click **Edit** for an existing admin property *APTS_ProposalConfig*.
3. In the **Name** field, type *APTS_ProposalConfig* and in the **Value** field, type *XML*.
4. In the **Code** field, type the following code:

```
<ProposalConfig>
<PDFSecurityDefault>
<CanPrint>true</CanPrint>
<CanCopy>true</CanCopy>
<CanChange>false</CanChange>
<CanAddNotes>true</CanAddNotes>
<CanFillFields>true</CanFillFields>
<CanAssemble>false</CanAssemble>
</PDFSecurityDefault>
</ProposalConfig>
```

5. All the tags defined over here are available in the *ProposalConfigSchema* static resource. To view this static resource, you may navigate to **Setup** > **Develop** > **Installed Packages** > **Apttus Proposal Management (Managed)** > **View Components** > **ProposalConfigSchema** > **View file**.
6. Click **Save**.

The proposal document generated in PDF format will now be secured with the system password.

Configuring Quick Quote Mode

The Quick Quote feature allows Sales reps to create a simple quote with fewer clicks from opportunity. The Sales rep can directly go to the Catalog page from Opportunity, configure products, and finalize the cart without having to create a proposal in turn reducing the number of clicks. Quick Quote feature is especially meant for creating budgetary quotes where Sales Reps can begin the quoting process by populating the quote header with limited values.

To enable the Quick Quote feature, you must create and add a custom button to the opportunity layout. CPQ populates the critical header fields in the new quote based on the parameters passed in the formula of the button. You can define variable or static parameters. In the case of variable parameters, CPQ retrieves the values from opportunity. You can use the following parameters in the URL as described:

Fields	Required	URL Parameter	Description
Opportunity ID	Yes	*businessObjectId*	Add the ID of the opportunity to the URL.
Price List	Yes	*priceListId*	Add the ID of the price list to the URL. You must use one of the following ways to pass the price list. • Add a static value of the price list ID the URL. This is applicable when the quotes are always created with a fixed price list. • Create a Price List lookup field on the Opportunity object and a formula field to store the ID of the selected price list. Pass this formula field value in the URL.
Account	Yes	*NA*	This field is copied by default.

Fields	Required	URL Parameter	Description
Expected Start Date	No	*expectedStartDate*	You must create the field on the Opportunity object with datatype as *Date* and add the field in the URL.
Expected End Date	No	*expectedEndDate*	You must create the field on the Opportunity object with datatype as *Date* and add the field in the URL.
Existing Configure Products URL parameters	No	• *flow* • *useAdvancedApproval* • *asyncFinalize* • *deferPricingUntilCart*	Add the parameters based on your business needs.

The custom fields with the same API name, that are common between both opportunity and quote are also populated.

To create a quick quote custom button on Opportunity

1. Go to **Setup** > **App Setup** > **Customize** > **Opportunities** > Buttons, Links, and Actions.
2. Click **New Button or Link**.
3. Fill in the details as explained below.

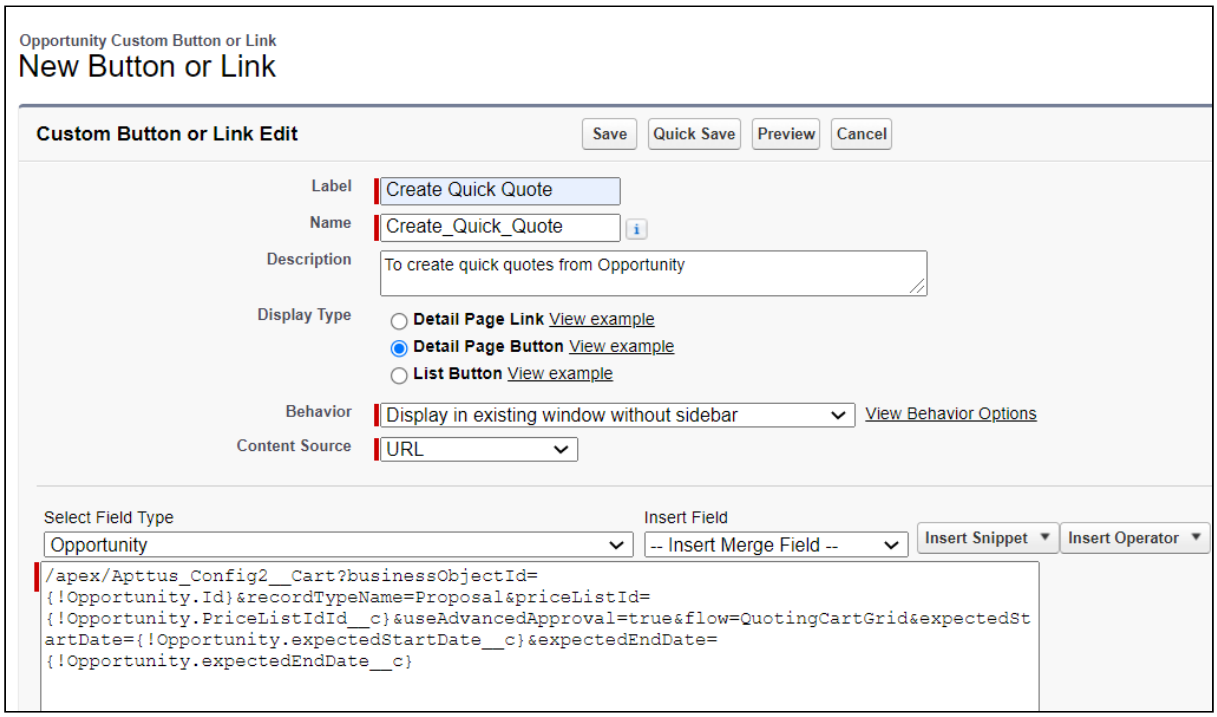
Fields	Description
Label	Enter the name of the button. For example, *Create Quick Quote*.
Name	This field is populated automatically when you click the text box.
Description	Enter a description for the button.
Display Type	Select **Detail Page Button**.

Fields	Description
Behavior	Select the behavior based on where you want to display the button
Content Source	Select **URL**.

4. In the Code Snippet block, type the following:

```
/apex/Apttus_Config2__Cart?businessObjectId={!Opportunity.Id}
&recordTypeName=Proposal&priceListId={!Opportunity.PriceListIdId__c}
&useAdvancedApproval=true&flow=QuotingCartGrid&expectedStartDate={!
Opportunity.expectedStartDate__c}&expectedEndDate={!
Opportunity.expectedEndDate__c}
```

> Ensure that you enter the name of your Flow setting in the *flow* parameter in the above formula. In our example, it is *QuotingCartGrid*.



5. Click **Save**.

To add the quick quote button

After you create the custom button, add the Quick Quote custom button to the layout of the opportunity.

1. Click **Page Layouts** and click **Edit** for the page layout to which you want to add the **Create Quick Quote** button.
2. From **Buttons** in the Layout configuration window, select **Create Quick Quote** and drag and drop it onto the page layout.
3. Click **Save** in the Layout configuration window.

Create Quick Quote button is added and saved onto the page layout.

For more information on creating a quick quote, see Creating Quick Quotes.

Customizing Visualforce Pages

You can configure which standard columns and actions are available in the catalog, attribute, bundle, and shopping cart pages, the style they use, and the order in which they are displayed. Additionally, you can create your own custom fields and actions and display them in these Visualforce pages.

The Visualforce pages are used for product selection, select attributes, options, and in the cart page change quantities, apply discounts, and more for your quote/proposal. If there are default fields and actions that are not required, or new fields you want to add, you can make those changes through the CPQ *Config Settings* tab.

- Configuring Display Columns Settings
  The Display Column Settings enable you to display only those key fields needed to quickly and easily configure your quote/proposal.
- Configuring Display Actions Settings
  The *Display Action Settings* page provides a simple interface for selecting actions, applying styles, and reordering them. You can even create and use your own custom action.
- Configuring Custom Actions for Auto-Synch of Cart Lines
  The Custom Actions feature has been enhanced to auto-synch cart lines with proposal lines and navigate to other pages.

Configuring Display Columns Settings

The Display Column Settings enable you to display only those key fields needed to quickly and easily configure your quote/proposal.

The *Display Column Settings* page provides a simple interface for selecting fields, making fields editable, applying styles, and reordering them.

To display different number type values as expected, you can add one of the following styles to cart line item fields:

aptCurrency	This will prepend the appropriate currency symbol for the org's locale and retain two digits after the decimal place for the field value. You can change the number of digits using the Currency Field Precision in Config System Properties.
aptPercentage	This will retain two digits after the decimal place (if present) and append the percentage symbol to the field value. You can change the number of digits using the Percentage Field Precision in Config System Properties.
aptQuantity	This will retain two digits after the decimal place (if present). You can change the number of digits using the Quantity Field Precision in Config System Properties.

The following style classes have to be applied in pair to the dynamically selected adjustment column and its value column. i.e., Adjustment Type and Adjustment Amount columns.

- aptAdjustmentType
- aptAdjustment

The formatting for the **Adjustment Amount** field will change dynamically based on the **Adjustment Type**. This is achieved by applying available style classes to the adjustment based on the value in the pick list of the adjustment type.
Note: For changes to the shopping cart beyond those described here, you would need to create custom pages of the Shopping Cart (*CartView* and *CartFinalize* Visualforce pages) and write your own Apex code to further control what fields are displayed and how the page is organized.

> ⓘ **In order to use PULL promotions on the Cart, you must follow the procedure mentioned below with Display Type = *Apply Promotions*.**

To customize a display column

1. Click ✚ (All tabs) and click **Config Settings**.
2. Click **Display Column Settings**.
3. To create or modify columns on one or more of the following, select the view from the **Display Type** picklist.
   - Cart Line Item
   - Cost Line Item

- Cart Total Item
- Installed Product
- Related Purchases
- Agreement Price Rule
- Agreement Price Tier
- Adjustment Line Item
- Usage Price Tiers
- Apply Promotions
- Option Line Items
- Price Ramp
- Tiered Price
- Related Price
- Mass Update
- Mass Edit (Assets)

  > Mass Edit (Assets) is an invalid selection for Display Type.

- Collaboration Parent Line Item
- Collaboration Child Line Item
- Line Item Key Metrics
- Line Item Waterfall
- Summary Group Key Metrics
- Summary Group Waterfall
- Optional Cost and Profitability Fields
- Cart Summary
- Asset Termination

  > If you add only the **Termination Date** field, CPQ displays:
  >
  > - The Confirm Swap pop-up with the **Effective Date** field instead of the Confirm Installed Products Swap page during asset swap.
  > - The Confirm Termination pop-up with the **Termination Date** field instead of the Confirm Termination page during asset termination.
  >
  > When the Conga Billing package is not installed in your org, it is recommended to display the Confirm Swap or Confirm Termination pop-up. Otherwise, all billing fields will be blank on the Confirm Installed Products Swap or Confirm Termination page.

4. Select your desired flow from the **Flow** picklist.
5. The **Sequence** column is system-generated and based on your **Display Type** selection in Step 3, the **Display Type** column displays the line item name.

6. In the **Field Name** column, select a field you want to display as a column.

   > You should not add Option as a column as Product and Option columns are identical in the CartDetailView page.

   The supported values are:
   - Termination Date
   - Start Date
   - End Date
   - Original Start Date
   - Net Price
   - Billed Through Date
   - Total Billing
   - Invoiced
   - Pending Billing
   - Estimated Credit
   - Asset TCV
   - Current Contract Start Date
   - Current Contract Value
   - Current Contract Billing (Invoiced)
   - Current Asset Billing (Invoiced)
   - Current Contract Pending Billing
   - Current Asset Pending Billing
   - Custom asset line item fields

     > You can allow Sales Representatives to edit the custom asset line item field values on the Confirm Termination page instead of waiting to go to the Cart page to edit these values. For example, they may want to capture the reason for terminating assets. However, CPQ does not support text area type of fields. CPQ carries these edited field values from the Confirm Termination page to the Cart page where Sales Representatives can make further changes to those field values, if required. You can select these fields only if you have added the same to the **Editable Fields for Cancelled Lines** setting. After selecting one of these fields here, if you remove the same field from the **Editable Fields for Cancelled Lines** setting, CPQ automatically removes that field from **Display Column Settings**. You must execute the maintenance job. For more information, see Configuring Installed Products Settings.

7. To make the field name column editable, select **Is Editable**. Clearing the **Is Editable** check box, makes the field read-only.
   Note: If you select **Display Type** as *Installed Product*, the Is Sortable check box is

displayed. To make the field name column sortable, select the Is Sortable check box.Note: For **Selling Term**, and **Selling Frequency** columns to be editable, apart from selecting the **Is Editable** check box, ensure that **PriceType** for the corresponding product is set as either *Recurring*, *Usage*, or *Included Usage*. Additionally, for **Selling Frequency**, ensure that **Allow Proration** check box is selected on the price list item.

8. In the **Style** column, type values separated with a colon to change the default appearance of a field. Example 1: If you want to resize all the columns in the cart, type *width: 30px;text-align:left*. Example 2: If a product name is lengthy and is not displayed properly on the shopping cart page, type *text-align: left; width: 300px; max-width: 300px; display: block;white-space:0px;*. The product name is wrapped and displayed completely on the cart page.
9. In the **Style Class** column, type one of the following to change the way dynamic columns are displayed:
    - *aptCurrency*
    - *aptQuantity*
    - *aptPercentage*
    - *aptAdjustmentType*
    - *aptAdjustment*

    Note: These style classes apply two decimal points to the fields accordingly. You can change the number of decimal places from **Config Settings > System Properties** using the *Currency Field Precision*, *Percentage Field Precision*, and *Quantity Field Precision*.
10. In the **Header Style** column, type values separated with a colon to change the default appearance of the header of a column. For Example: *width:200px;*
11. You can also do one or more of the following:
    - To add or delete a row, use the  and icons.
    - To re-order the rows, use the  and  icons.
12. Click **Save.**

The columns are displayed as configured in the Display Columns Settings.

Configuring Display Actions Settings

The Display Action Settings page provides an interface for selecting actions, applying styles, and reordering them. You can even create and use your own custom action.

In order to create custom actions, you must first create a custom label.

To create a custom label

1. Go to **Setup > Create > Custom Labels** and click **New Custom Label**.
2. Type a mandatory **Short Description** and type a mandatory **Name.**
3. In **Categories**, type comma-separated values that can be used in filter criteria when creating custom label list views. For example Pages, Label, Components, and more.
4. In **Value**, type mandatory text in the form of a brief description or copy and paste the field name from Step 1.
5. Click **Save.**

A custom label is created and saved.

To change a custom label

1. Go to **Setup > Create > Custom Labels** and search for **Change**" label with Installed Products Change Action as the short description.
2. Click the Change label. The system displays the Custom Label Detail page.
3. Click the button, **New Local Translations/Overrides** in the Local Translations/ Overrides section. The system displays the New Translation page.
4. Select the appropriate language from the drop-down.
5. Enter the Translation Text as an Add-on in the text box.
6. Click **Save**.

The custom label is changed.

To customize display actions

If you want to create and use custom actions, you must have an existing custom label for the action.

1. Click (All tabs) and click **Config Settings**.
2. Click **Display Action Settings**.
3. From **Display Type**, select the page where you want to place the custom action.
4. From the **Flow** drop-down, select the flow you want to configure.
5. The **Sequence** column is system-generated and based on your Display Type selection in Step 3, the Display Type column displays the line item name. The Action and Action Label Name display the default actions.
6. You can create a custom Action button, where the **Action** column name is CustomAction <number> and in **Action Label Name** column, type the field name of the custom label you created. For example: Action = CustomAction1 and Action Label Name = Edit_Quote.
7. From the **Display As** column, select one of the following:

Value	Description
Action	Select this to set the action as an action button
Task	Select this to set the action as Task menu item in the Status bar.
Action and Task	Select this to set the action as both an action button and a Task menu item.
Sidebar Action	Select this to set the action as an action button on the sidebar of the Configuration page. You can define a maximum of 3 actions in the sidebar. Only the first 3 sidebar actions in the list are valid, CPQ does not display the remaining sidebar actions on the Configuration page at all.
Nav Link	Select this to set the action as a hyperlink in the Navigation bar.
Help	Select this to set the action as a Help menu icon.

For the new UI, the behavior is as follows:

Display As	Behavior
Action	Listed as a primary button. Depending on the sequence that you have defined, only one button is placed on the top and the rest of the buttons are placed in the drop-down list.
Task	Listed as a secondary button. A maximum of 4 secondary buttons are allowed on any page.

For example, on the catalog page displayed below, **Go to Pricing, Installed Products**, **Save,** and **Compare Products** have **Display As** = *Task,* whereas **Abandon**, **Update Price**, **Quick Save** and **Finalize** have **Display As** = *Action.*

8. From the **Action Area** column, select one of the followings to position the action:

Value	Description
Center	Select this for center alignment of the button

Value	Description
Left	Select this for left alignment of the button
Right	Select this for right alignment of the button
More	Select this to add the button as an item under the More action button

> **Note**
>
> Note that the alignment settings are not applicable for Nova and Grid UI.

9. In the **Action Style Class** column, type one of the following available style class name for the action.

Style	Description
apt-selected-btn	Type this to display the button in the drop-down menu with a white background and black font color. This also indicates the step that an end-user is on.
apt-left-btn	Type this to display an arrow pointing left on the button.
apt-right-btn	Type this to display an arrow pointing right on the button.

> **Note**
>
> If you want to use your own style, create a CSS file including all of the custom styles you want in the shopping cart and upload it as a static resource. Type the name of the CSS file in **Config Settings > System Properties > CSS Override**. This enables you to use the styles defined in the CSS file as opposed to out-of-the-box styles.

10. In the **Action Page** column, in **Action Page**, type the custom Visualforce page name and in **Action Params**, type the parameter that you want to pass for your custom page. For example: {!Apttus_QPProposal__ProposalID__c.ID} If you leave the **Action Params** field blank, by default, the system includes the ID, Business Object ID, and Config Request ID as the dynamic parameter. Example URL: .....CustomVFPage?

id=a1Oe00000018DLUEA2&retId=a0ee0000003GUCfAAO&businessObjectId=a0ee0000003GUCfAAO&configRequestId=a1ie0000000kbjTAAQ

11. From the **Behavior** column, to specify the action page behavior, select one of the following:
    - To open within the same window, select *Self*.
    - To open in a new window, select *New Window*.
    - To open in a dialog, select *Dialog Window*.
12. From the **Action Type** column, select one of the following:
    - To redirect to another page using the custom button and not show the spinner, select *Quick Redirect*. This option ignores the unsaved changes made by the customer before clicking the custom button.
    - The other options, such as *Save, Quick Save*, *Submit for Approval, and Generate* perform the same function as the out-of-the-box action buttons.
13. To enable or disable actions, select or clear the **Is Enabled** check box.
14. To enable permanent display of the action irrespective of the setting in Step 12, select **Always Display** check box.

Scenario	Display Action
Is Enabled = True and Always Display = True	Action is displayed.
Is Enabled = False and Always Display = True	Action is displayed but inactive. You cannot click the action.

15. Click **Save** to save the changes you made on this page.

The action is added to the list of actions to be displayed on the Visualforce pages. At any given time, if you want to disable the custom action, go to the Display Action Settings page and clear the **Is Enabled** check box.

Use Case

You can configure action buttons displayed on the Summary panel on the Configuration page. You can add a maximum of 3 buttons on the Summary panel. If you configure more than 3 buttons, only the first 3 buttons are displayed on the Summary panel. The following action buttons are displayed on the Summary panel by default:

- Go to Pricing
- Update Price
- Validate

To configure action buttons on the Summary panel on the Configuration page

1. Go to **All Tabs** > **Config Settings** > **Display Action Settings.**
2. From the **Display Type** drop-down, select **Attribute Page**.

   > ⓘ You can select **Display Type** as **Bundle Page** to configure actions on the sidebar for bundle page in case of sub-bundles.

3. From the **Flow** drop-down, select the flow you want to configure.
4. Find the buttons you want to display on the Summary panel and define the following fields as described:

Field	Value
Display As	*Sidebar Action*
Action Area	*Center*

5. Select the following checkboxes:
   - **Is Enabled**
   - **Always Display**
6. Click **Save**.

To hide default action buttons on the Configuration page

1. Go to **All Tabs** > **Config Settings** > **Display Action Settings.**
2. From the **Display Type** drop-down, select **Attribute Page**.
3. From the **Flow** drop-down, select the flow you want to configure
4. Find the buttons you want to hide and clear the following checkboxes:
   - **Is Enabled**
   - **Always Display**
5. Click **Save**.

Configuring Custom Actions for Auto-Synch of Cart Lines

The Custom Actions feature has been enhanced to auto-synch cart lines with proposal lines and navigate to other pages. When Auto-synch is turned on, the system does validation for errors and auto-synchs the cart lines with the Proposal Lines. You can view

the synched line items in the Quote/Proposal detail page under the Line Items related list. Auto-sync is done when following conditions are satisfied:

- No error produced by validation callback
- No Constraint Rules error
- No line items in price pending state
- No Must Configure product is in configuration pending state

To configure custom actions for auto-synch

1. Click (All Tabs) and click **Config Settings**.
2. Click **Display Action Settings**, and scroll down to add a custom action button.
3. Type an **Action Label Name**, and from the **Display As** list, select *Action*.
4. Select an **Action Area**, and in **Action Style Class**, specify a style class.
5. In Action Page, type the name of the Visualforce page that appears when the action button is clicked.
6. In **Action Params**, type *autoSync=true*.
7. In **Behavior**, select an option of how you want the action page to open.

   > The auto-synch custom actions should not have action type (Save, Quick Save, Generate) specification.

8. To enable the custom Action button, select **Is Enabled**.
9. To always display the custom Action button whether enabled or disabled, select **Always Display**.

A custom action is created with auto-synch capability.

Disabling the Clone Icon on the Cart Page

You can now choose to disable the clone icon on the Cart page. Select the **Hide Copy Action** check box in **Config System Properties** to disable the clone icon for in the cart

To disable the clone icon for on the cart page

1. Go to **Setup** > **Develop** > **Custom Settings** > **Config System Properties** > **Manage** > **System Properties**.
2. Click **Edit**.
3. Select the **Hide Copy Action** check box.

4. Click **Save**.

Configuring Flows

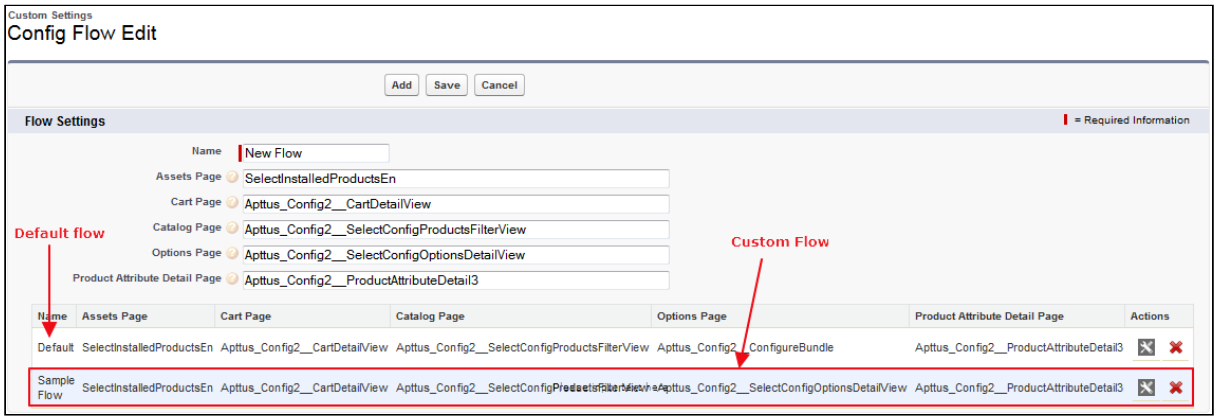
As an administrator, you can now vary the Catalog, Options, Attributes, Installed Products and Cart pages that an end user interacts with as well as the data and actions within those pages based on the end user profile and other business rules. This is made possible by introducing the concept of Flows. Flows are groups of pages that are assigned to each step in the CPQ process. Earlier the only option to set up the Visualforce pages was through custom settings.

You can now use Flow Settings to assign custom or out-of-the-box pages to each step and configure the fields and actions that are visible within each page. You can then create criteria that would determine the Flow which in turn determines the pages that are displayed when an end user clicks the Configure Products button. These flow settings once setup, override the Visualforce pages setup in custom settings.

To create flows using flow settings

1. Click ✚ and click **Config Settings**.
2. Click **Flow Settings**.
3. In **Name**, type a name for the flow. For example: Sample Flow
4. Type the Visualforce pages in the following fields:
    - Assets Page
    - Cart Page
    - Catalog Page
    - Options Page
    - Product Attribute Detail Page

   Hover the mouse over ? to see valid values for each field.
5. To add the new flow, click **Add**. An entry is added to the list below.



Default - When there is no flow created, the system uses the Default flow.
Custom Flow - In this example, Sample Flow is the custom flow created.

6. Click **Save**.

A new flow is added and saved. Use the flow name you just created in the parameter to pass on for the Configure Products button.

To set up the configure products button to use the flow

You must have an existing flow.

1. Go to **Setup** > **App Setup** > **Create** > **Objects**.
2. Scroll down to select the Quote/Proposal object and from the **Custom Fields & Relationships** related list, select **Configure Products**.
3. Click **Edit** and insert the flow name in the parameter as below.

```
IF ( LEN(Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/
Apttus_QPConfig__ProposalConfiguration?id=" &Id+'&flow=Sample Flow', IMAGE("/
resource/Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"),
NULL)where flow=<name of the flow you created>
```

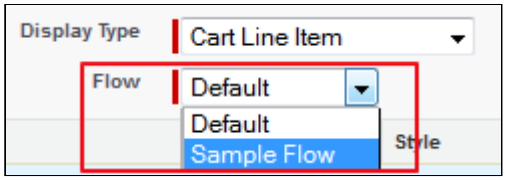
4. Click **Save**.

The parameter is saved for the Configure Products button. On the Quote/Proposal page when you click the Configure Products button, all the pages appear as defined in the flow. You can modify the columns and actions that you want to use with the flow you created.

To set up columns and actions for flows

You must have an existing flow.

1. Click ✚ and click **Config Settings**.
2. For setting up columns or actions or both for the custom flow, do one or more of the following:.
    - For columns, click **Display Column Settings**
    - For Actions, click **Display Action Settings**
3. Select a **Display Type** and from **Flow**, select the flow you created.



4. Click **Save**.

The custom display columns and actions are set for the flow you created. All the Visualforce pages that you set up in the flow are displayed with the columns and actions set up in the config settings. If you select Default flow, the system displays the columns and actions set up in the Default flow. Clicking the Load Default Settings button will override all the columns and action settings to out-of-the-box settings in the Default flow.

Enabling Enterprise as a QTC Profile Value

When you have enabled TurboPricing in the org to process large-sized carts, Conga recommends you to create such quotes with the **QTC Profile** value as *Enterprise*. In a quote with *Enterprise* QTC Profile, CPQ processes downstream operations asynchronously for a maximum of 2000 lines. When the Sales Representative selects the **QTC Profile** as *Enterprise* during quote creation, CPQ performs the following downstream operations in async mode:

- Finalizing the cart
- Cloning a quote with Line items
- Creating an agreement with line items (single agreement)
- Creating an order (single order)
- Creating assets

You must manually add the picklist value of *Enterprise* in the **QTC Profile** field of the Quote/Proposal, Product Configuration, and Agreement objects.

To add Enterprise as a QTC Profile value

You must add the value *Enterprise* in the **QTC Profile** field of the Quote/Proposal, Product Configuration, and Agreement objects so that CPQ displays the value *Enterprise* in the **QTC Profile** drop-down on the quote for the Sales Representative.

1. Go to **Setup** > **App Setup** > **Create** > **Object**.
2. Click **Quote/Proposal**.
3. In **Custom Fields & Relationships**, Click **QTC Profile**.
4. On **QTC Profile(Managed)** page, click **New** on the **Values** section.
5. Enter *Enterprise*.
6. Click **Save**.

> Repeat steps from 3 to 6 for the **Product Configuration** and **Agreement** objects. For the **Product Configuration** object, in Step 3 you must select the **Business Object Profile** field instead of **QTC Profile**.

Configuring Progress Tracker for Async Operation

When you enable this feature from the Quote/Proposal or Agreement object, CPQ displays notifications on the Quote Detail and Agreement Detail page tracking different stages of async operations. The tracker informs the Sales Rep of the progress in real-time. It also immediately informs the Sales Rep in case of errors and provides detailed information of the error, helpful for debugging. For example, when a Sales Rep finalizes a cart asynchronously and returns to the Quote Detail page, a progress bar displays the progress of the finalize operation.

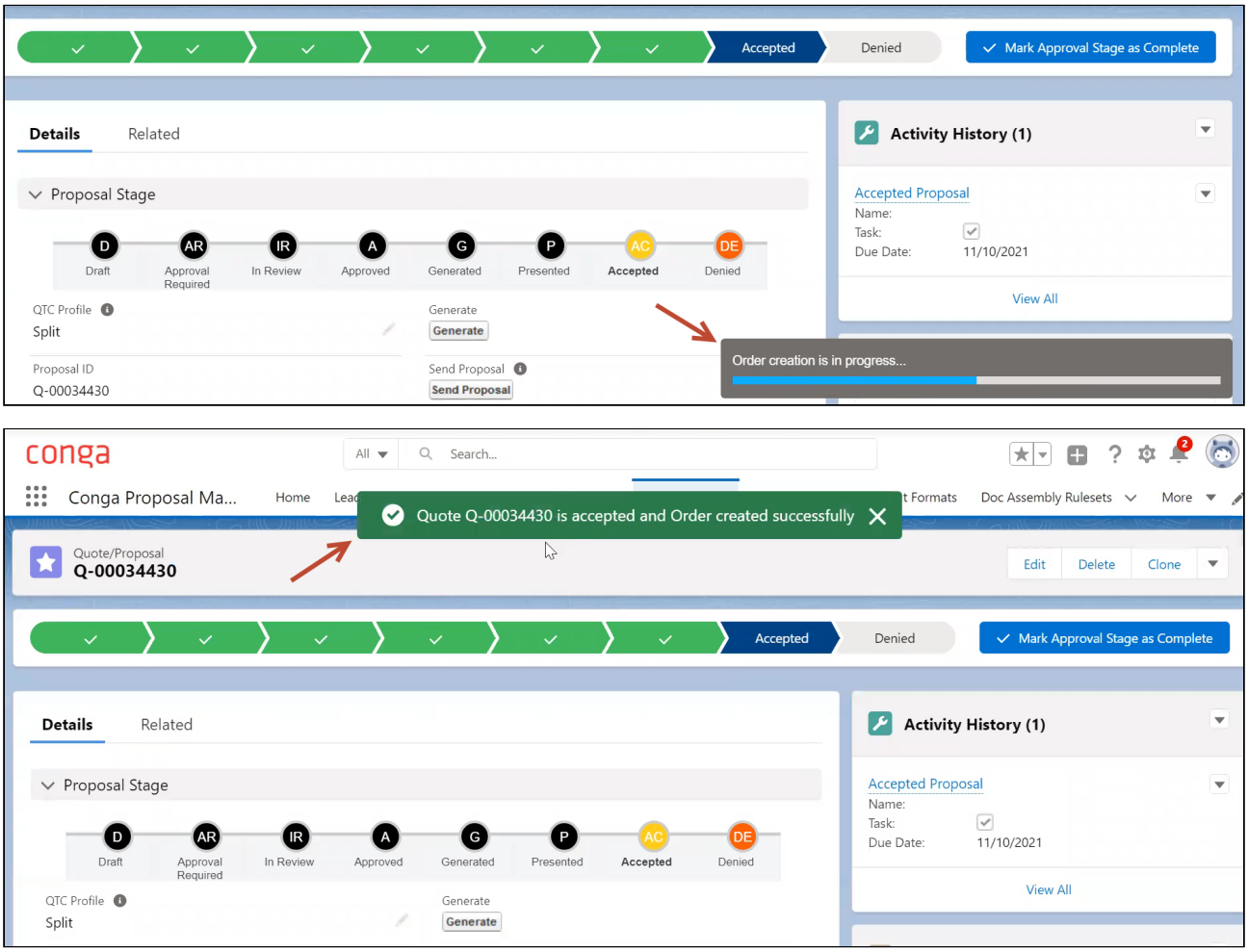
You can use this feature to track the progress of the following tasks that CPQ processes asynchronously:

- Finalizing a cart
- Cloning a quote or agreement
- Accepting quote to create order
- Creating an agreement with line items from Split and Enterprise quotes.
- Activating agreement to create order

The feature tracks the following stages of the async operation:

- The progress of the async operation
- The errors encountered while processing the operation
- Completion of the async operation. Upon completion, the Quote page is refreshed automatically
- Success notification after the completion of the operation(only supported for Lightning Experience)

The Sales rep can use this feature for async operations for quotes of all profiles namely Regular, Split, and Enterprise.



Follow the steps below to enable the tracker.

To enable the tracker for Salesforce Classic UI

1. Go to **Setup** > **App Setup** > **Create** > **Object**.
2. Click **Quote/Proposal** or **Agreement** object based on the page you want to enable the tracker for.
3. In the **Buttons, Links, and Actions** section, click **Edit** next to the **View** record.
4. Define the fields as described in the below table:

Field	Description
Label	The label is populated automatically.
Name	The name is populated automatically.
Default	The default has already been selected.

Field	Description
Salesforce Classic Override	Select the **Visualforce page** radio button. From the picklist next to the **Visualforce page** radio button, select • For Quote/Proposal object: **ClassicProposalRecordDetail [Apttus_QPConfig__ClassicProposalRecordDetail]** • For Agreement object: **ClassicAgreementRecordDetail [Apttus_CMConfig__ClassicAgreementRecordDetail]**
Lightning Experience Override	Select the **Lightning Page** radio button.
Mobile Override	Leave the field as-is.
Comment	Add comments with a description of the action.

5. Click **Save**.

To enable the tracker in Lightning Experience

1. Go to **Lightning Experience**.
2. Click the wrench icon ( ). Click **Edit Page**. Lightning App Builder page is displayed.
3. From the **Pages** dropdown, select the Proposal or Agreement record page based on the page you want enable the tracker for.
4. Under the **Components** tab, find and expand the **Custom - Managed** list.
5. Drag and drop the following components from the **Custom - Managed** list to the proposal record page:

Component	Description
proposalBanner	Add this component to the Proposal record page.
agreementBanner	Add this component to the Agreement record page.

Component	Description
BannerRefreshView	You must hover the cursor over the name and select the correct component as described below: • For Proposal record page: *BannerRefreshView (Apttus_QPConfig).* • For Agreement record page: *BannerRefreshView (Apttus_CMConfig).*

> Conga recommends placing the above components between the **Highlights Panel** and **Path** components.

6. Click **Save**.

You can also translate or override the labels and messages in the progress bar based on your Locale and Language. For more information, see Managing Translation of Fields and Values in Different Language.

Turning off Async Operation Email Notification

You can use the Admin Setting **APTS_DisableCartAsyncNotification** to turn off the email notifications you receive after completion of any async operation such as cart finalization and pricing in every type of quote or agreement. In the Split quote or agreement, the notification for cart splitting is also disabled along with finalization and pricing. If you enabled the Progress Tracker on the quote or agreement page to monitor the progress of all the async operations, email notifications are redundant and you can use this feature to avoid them.

To turn off async operation email notification

1. Click  and click **Admin**.
2. To create a new Admin Setting, click **New**.
3. Enter **Name**, **Value**, and **Code** for the Admin Setting as described in the below table:

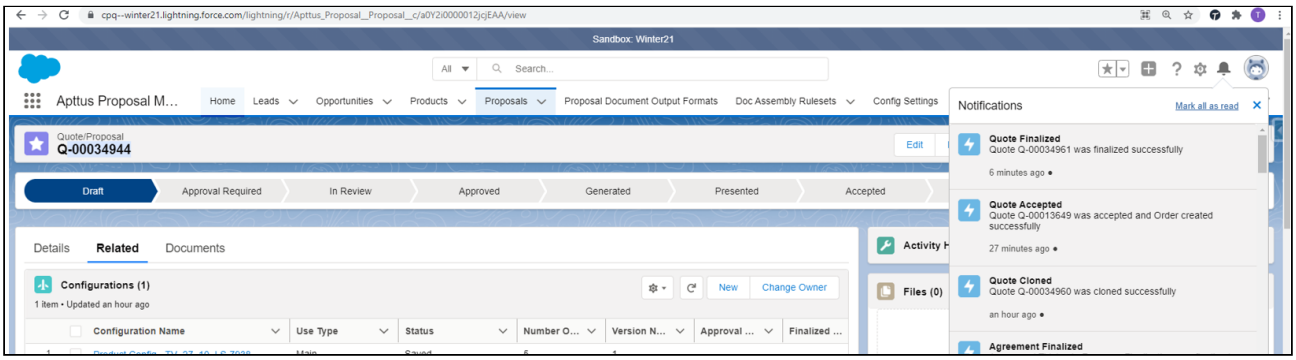
Name	*APTS_DisableCartAsyncNotification*
Value	*true*
Code	Leave the field blank

4. Click **Save**.

Configuring Push Alerts

You can enable Salesforce Push Alerts to track async operation status when you are not on the Quote or the Agreement page. You must select **Enable Notification Feed** in Config System Properties. If you enable push alerts, the Sales rep receives notification for the following asynchronous CPQ operations:

- Finalizing a cart
- Cloning a quote or agreement
- Accepting quote to create order
- Creating an agreement with line items from Split and Enterprise quotes.
- Activating an agreement to create an order



To enable push alerts

1. Go to **Setup > Develop > Custom Settings > Config System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Select **Enable Notification Feed** checkbox.
4. Click **Save**.

Configuring Document Generation for Quote

You can you Document Generation(Doc Gen) feature to generate a document for the line items in the quote. You can use this functionality by clicking **Send Proposal** button on the Quote/Proposal Detail or Cart page. After clicking the button you are redirected to Doc Gen Wizard page. The Sales Rep can generate in a particular format, merge documents, preview, and send the documents to customers on the Doc Gen Wizard page. To use the feature, you must configure the following settings.

To configure output format for the profile

You must define an output format for the profile, in case you want to generate documents automatically.

1. Click the All Tabs icon () and click **Proposal Document Output Formats**.
2. Select the profile you want to update.
3. Select one of the following:
    - Prompt
    - Doc
    - Docx
    - PDF
    - RTF
4. Click **Save**.

To add endpoint URL

You must add the endpoint for the document URL that you want to list on the Proposal Doc Gen. Follow the steps below:

1. Go to **Setup** > **Security Controls** > **Remote Site Settings**.
2. Add a New Remote Site.
3. Click **Save**.

To enable Salesforce Files

1. Go to **Setup > Develop > Custom Settings > Proposal System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Select **Enable File**.
4. Click **Save**.

To configure preview of File documents on doc gen page

When you are using Files in your org, CPQ does not automatically enable previews for files. You must disable **Enable clickjack protection for customer Visualforce pages with headers disabled** fields, to enable file document preview on the Doc Gen page. Follow the steps below to enable preview on the Doc Gen page:

1. Go to **Setup > Administrator Setup> Security Controls > Session Settings> Manage**.
2. Find **Clickjack Protection**.
3. Disable **Enable clickjack protection for customer Visualforce pages with headers disabled**.

To add Send Proposal button on the quote page

1. On the Quote/Proposal Detail page, click **Edit Layout**.
2. Select **Fields** on the left-hand side menu.
3. Search **Send Proposal** using Quick Find.
4. Drag and drop the field on the quote.
5. Click **Save**.

Configuring Fast Doc Gen

You can use this feature to generate the default document automatically. When the Sales Rep clicks on **Send Proposal** button, a document is generated automatically using a queried or default template. The generated document is in output format defined and is displayed on the Doc Gen page. You can set up default document generation for the overall org or for specific scenarios as the document generation depends upon the templates that are defined.

> ⓘ In the case of large quotes, Conga recommends that you must not use Fast Doc Gen as CPQ generates default documents in sync mode.

CPQ generates documents automatically based on the template that is defined in **Query Template Filter** and the **Default Template Name** setting. The automatic generation of documents using a particular template is based on the below-mentioned scenarios:

Scenario	
When a template is defined in Default Template Name but not in Query Template Filter	The document is not generated.
When the same template is defined in both Default Template Name and Query Template Filter	The document is generated using that template.

Scenario	
When a single template is defined in Query Template Filter but no template is defined in Default Template Name	The document is generated using that one template.
When multiple templates are defined in Query Template Filter but no template is defined in Default Template Name	the document is not generated.

Prerequisite

An output format must be selected for the profile. Refer to section ***To configure output format for the profile***.

To enable fast doc gen

1. Go to **Setup > Develop > Custom Settings > Proposal System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Define the fields mentioned in the table:

Field	Description
Enable Fast Doc Gen	Select the checkbox to enable fast document generation.
Default Email Template	Enter the name of the template to set it as default.

4. Click **Save**.

Configuring Large Document Generation for Quote

You can use the Large Document Generation feature to generate documents for quotes with a large number of line items. The document generation for a large quote is processed asynchronously in small chunks based on the threshold and batch size and the resultant document is attached to **Notes and Attachments**. This feature is available in the **Generate** button and not available in the DocGen, **Send Proposal** button.

You must define the following settings to allow the Sales rep to generate a document for a quote with a large number of line items. You need to define a threshold and a batch size for the number of line items in the quote. When the number of line items in the quote exceeds

the threshold, the quote is considered a large quote. The line items in the large quote are processed in smaller chunks based on the batch size you defined.

You must take into consideration the template that the Sales rep uses to generate the document before defining the batch size. For example, if the template includes information about only line item fields is considered a simple template and can have large batch size. Similarly, a template that has too many line item fields, attribute information, adjustment line details would be considered a complex template and should have a small batch size.

To define the threshold and batch size.

1. Go to **Setup > Develop > Custom Settings > Proposal System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Define the fields mentioned in the table

Feild	Description
Enable File	Select the checkbox to enable files in the quote
Enable Submit Merge Call	Select the checkbox to enable the **Submit** button to identify a quote as large doc quote based in the **Large Doc Threshold**.
Large Doc Threshold	Enter a number in the field. ⓘ The default and the maximum value of the **Large Doc Threshold** is *10000*. You can define a value lower than 10000. Setting a value of *-1* for the Large Doc Threshold invalidates the setting.
Large Doc Process Batch Size	Enter a number in the field. ⓘ The default value of the **Large Doc Process Batch Size** is *1000* and a maximum of *2000*. You can define a value lower than 2000, based on template complexity.

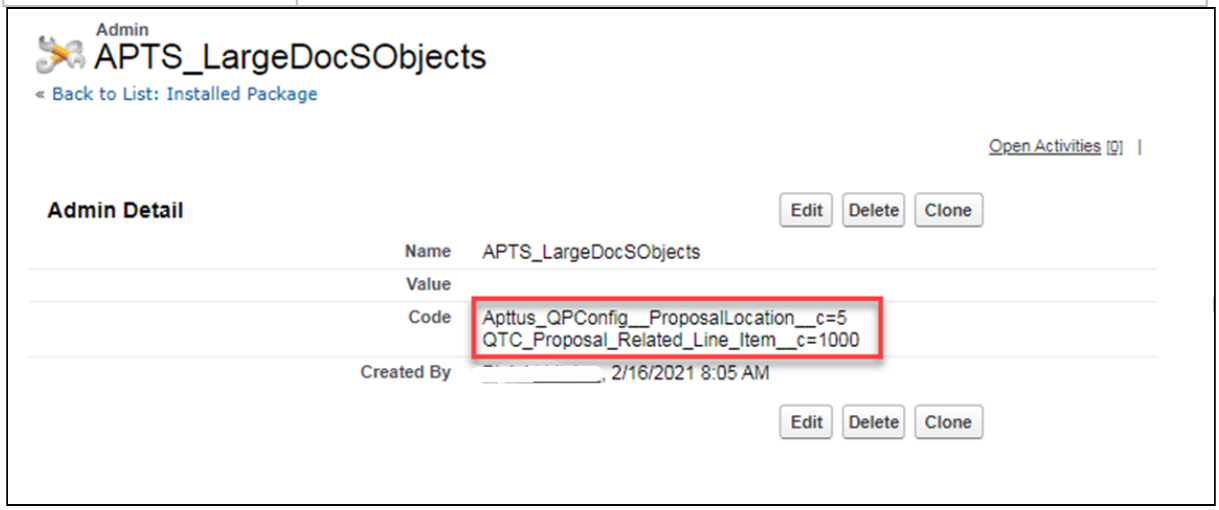
4. Click **Save**.

To define APTS_LargeDocSObjects Admin Setting

By default, CPQ considers Proposal Line Item and Agreement Line Item as Large Objects for document generation. In case you used any other object in the template, you must use this setting to define such objects as Large objects. You can specify multiple objects as Large Objects and also provide batch size for each of these objects. You must define the Admin Setting **APTS_LargeDocSObjects** by following the steps below:

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record.
3. Fill the following details

Field	Value
Name	APTS_LargeDocSObjects
Value	Leave the field blank
Code	*Apttus_QPConfig__ProposalLocation__c=5* *QTC_Proposal_Related_Line_Item_c=1000* ⓘ Use the format Auto Applied *<Object_API_name=<Batch Size>>* to defined the objects.You can add more objects separated by a new line. Decide the batch size based on the complexity of cart and template.



4. Click **Save**

Configuring Conga Sign for eSignature

You can integrate CPQ with eSignature solutions like Conga Sign. Conga Sign provides the ability to send the quote or proposal to the customer to gather signatures.

To Install Conga Sign

You must install the Conga Sign package in your org to use the Conga Sign features. For more information, see Installing Conga Sign Package.

To configure Conga Sign

Once you install the package in the org, you must complete the admin configuration in Conga Sign and set up integration with CPQ Proposal object. For more information, see Configuring Conga Sign.

To edit the layout of the Quote Detail page

After you configure Conga Sign, the **Send with Conga Sign** button is displayed on the Quote Details page. If the button is not displayed, you must edit the layout of the Quote Details page, and add the button on the header. You can also add Conga Sign Recipient and Conga Sign Transaction History related lists on the Quote Details page.

For more information, Sending Documents for eSignature Using Conga Sign.

Creating Promotional Banners

The banner settings feature has been deprecated.

Configuring Product Footnotes

Product Footnote object and two other objects: Proposal Footnote and Agreement Footnote enable you to dynamically include static *footnote* content in your quote/proposal documents, based on the products that are included in the quote as Proposal Line Items.

When a quote is finalized, the system creates corresponding Proposal Footnotes records. Merge Fields for this related object must be used in the proposal type templates. Similarly,

for the agreement documents and agreement type templates merge fields from the Agreement Footnotes object must be used for the system to bring footnotes for the included Agreement Line Items.

Currently, the system supports multiple Footnotes for the same product; each being a text for up to 4000 characters on separate lines. Only unique footnote records are brought by the system to the context object level on finalization. Footnotes for the line items of the type: Product/Service, i.e., standalone products or bundles, are supported.

To create product footnotes

1. Select a product and from the Product Detail section, click **Product Console**.
2. Scroll to Additional Data and click **Terms** to display the New Footnote page.
3. Type a **Footnote Name** and type the text of the footnote in **Body**.
4. Type a mandatory **Sequence** and click **Save New Footnote**.

The footnote record is saved and associated to with the product and automatically added to the Footnotes related list.

You can go to the New tab and add another footnote.

You can go the Associate tab and search for an existing footnote to add to this product's Footnotes related list. You can associate only one footnote per product, but a footnote can be associated to more than one product.

From the Edit tab you can make changes to the footnote or click Remove and choose to remove it from this product or any other products it has been associated with.

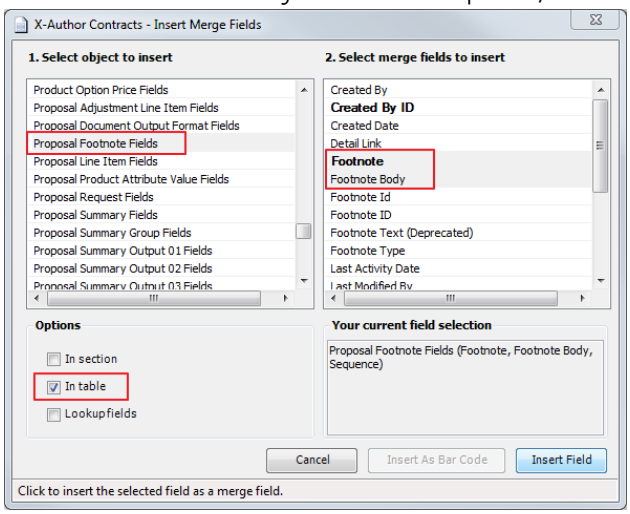
The newly created footnote can also be associated with other products.

To create a proposal type template

You must have existing product footnotes.

1. Login to X-Author Contracts, select the **X-Author Templates** tab and check out an existing proposal template (or create a new proposal type template).
2. Place the cursor in the template where you want the footnotes and click **Insert Merge Fields**.

3. From the *1. Select object to insert* pane, locate and select **Proposal Footnote Fields**.



4. From the *2. Select merge fields to insert* pane, select the fields that you want to insert.
5. From the *Options* section, select In Table and click **Insert Field**.

The merge fields are entered into the document template.

Proposal Footnotes in TABLE

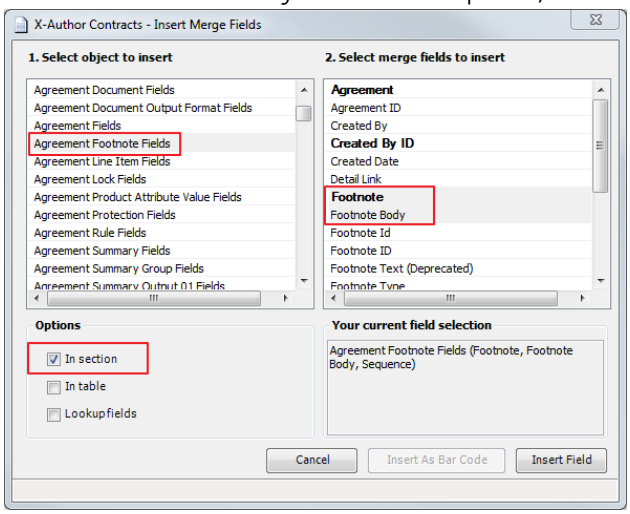
Sequence	Footnote	Footnote Body
{ MERGEFIELD Apttus_QPConfig__ProposalFootnote_start * MERGEFORMAT}{ MERGEFIELD ProposalFootnote_Sequence * MERGEFORMAT}	{ MERGEFIELD ProposalFootnote_FootnoteId * MERGEFORMAT}	{ MERGEFIELD ProposalFootnote_FootnoteBody * MERGEFORMAT}{ MERGEFIELD Apttus_QPConfig__ProposalFootnote_end * MERGEFORMAT}

To create an agreement type template

You must have existing product footnotes.

1. Login to X-Author Contracts, select the **X-Author Templates** tab and check out an existing proposal template (or create a new proposal type template).
2. Place the cursor in the template where you want the footnotes and click **Insert Merge Fields**.

3. From the *1. Select object to insert* pane, locate and select **Agreement Footnote Fields**.



4. From the *2. Select merge fields to insert* pane, select the fields that you want to insert.
5. From the *Options* section, select **In Section** and click **Insert Field**.

The merge fields are entered into the agreement template.

Agreement Footnotes in Section
{ MERGEFIELD Apttus_CMConfig__AgreementFootnote_section_start * MERGEFORMAT }
{ MERGEFIELD AgreementFootnote_FootnoteId * MERGEFORMAT }
{ MERGEFIELD AgreementFootnote_FootnoteBody * MERGEFORMAT }
{ MERGEFIELD AgreementFootnote_Sequence * MERGEFORMAT }
{ MERGEFIELD Apttus_CMConfig__AgreementFootnote_section_end * MERGEFORMAT }

Retaining the Order of Line Items in a Generated Proposal Document

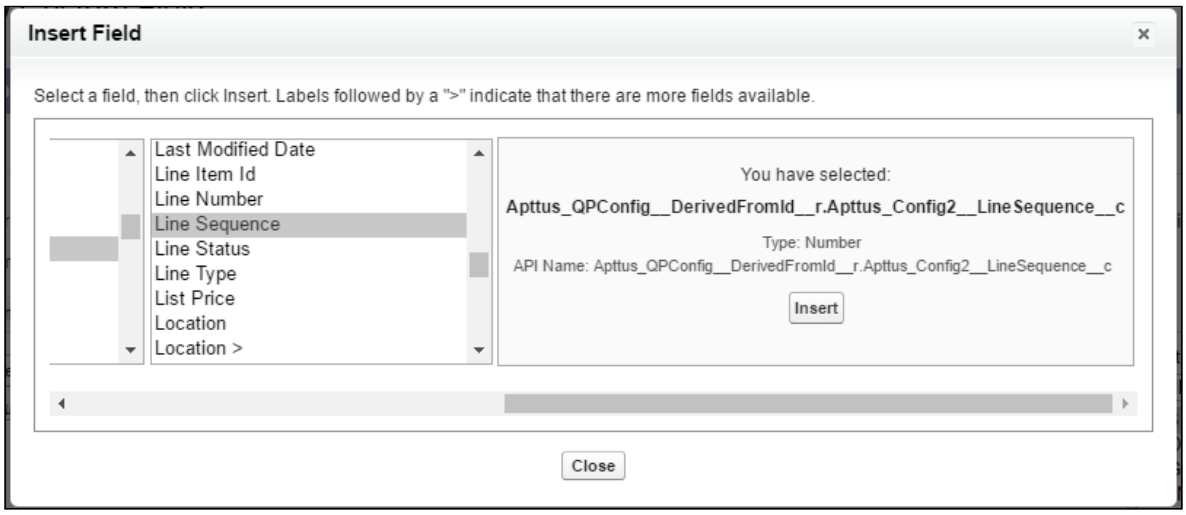
The sequence of the Line Items on the Cart depends on the addition and configuration of the products on the Catalog and Configuration pages. Using the drag and drop feature, a Sales Rep can rearrange the Line Items on the Cart and retain the rearranged sequence in a generated Quote/Proposal document.

To retain the order of the Line Items from the Cart to the Generated Quote/Proposal document, you must perform the following procedures:

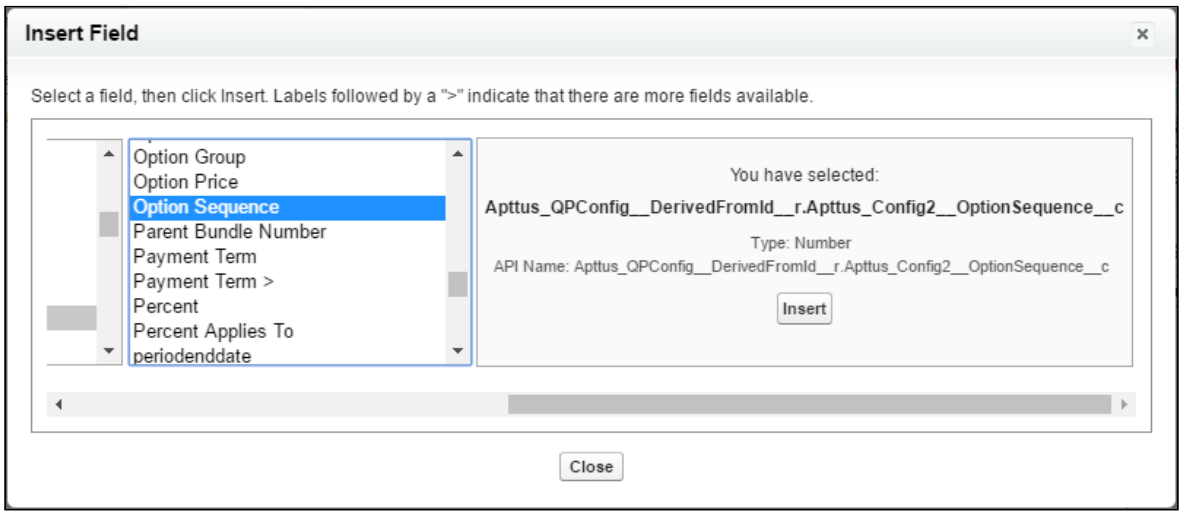
Creating new formula fields for Line and Option Sequence

You must create two new custom formula fields for Proposal Line Item object, which must be derived from **Line Sequence** and **Option Sequence** fields of Line Item object.

1. Go to **Setup > Create > Objects** and click **Proposal Line Item**.
2. In the Custom Fields & Relationships related list, click **New** and create a new Formula field.
3. Enter the field Label as *Sequence Line* and choose the **Formula Return Type** as *Number*.
4. Choose the Advanced Formula tab and build the formula field using the flow- **Proposal Line Item > Derived From > Line Sequence**.



5. Grant the necessary Field Level Security, assign the new custom field to the Page Layouts and click **Save**.
6. Repeat step: 2 to 5 for creating a new custom formula field for *Sequence Option* field, which will be derived from the formula- **Proposal Line Item > Derived From > Option Sequence**.



Creating an Admin entry for Line and Option Sequence

1. Click ⊞ and click **Admin.**
2. Do any one of the following:
   - Click **New** to create a new Admin entry.
   - Click **Edit** for an existing admin property *APTS_ProposalConfig*.
3. In the **Name** field, type *APTS_ProposalConfig* and in the **Value** field, type *XML*.
4. In the **Code** field, type the following code:

```
<ComplyConfig>
<SortSpec>
<SortObjects>
<SortObject>
<Name>Apttus_Proposal__Proposal_Line_Item__c</Name>
<SortFields>
<SortField>
<Name>SequenceLine__c</Name>
</SortField>
<SortField>
<Name>SequenceOption__c</Name>
</SortField>
</SortFields>
</SortObject>
</SortObjects>
</SortSpec>
</ComplyConfig>
```

Depending upon your business object (Quote/Proposal or Agreement), you can enable this for ProposlConfig or ComplyConfig respectively.

5. All the tags defined over here are available in the *ProposalConfigSchema* static resource. To view this static resource, you may navigate to **Setup > Develop > Installed Packages > Apttus Proposal Management (Managed) > View Components > ProposalConfigSchema > View file**.
6. Click **Save.**

Now when you generate a Quote/Proposal document, the rearranged sequence of the Line Items is reflected.

Updating the sequence of options in bundles that are reordered might cause CPU time out error. Conga recommends refraining from updating the option sequence after reordering the bundle. You must add an admin entry to disable the sequencing of options.

Updating the sequence of options in bundles that are reordered might cause CPU time out error. Conga recommends refraining from updating the option sequence after reordering the bundle. You must add an admin entry to disable the re-sequencing of options by following the steps below:

1. Click All Tabs icon ( ) and click **Admin.**
2. Click **New** to create a new Admin entry.
3. Fill the following details

Field	Value
Name	*APTS_IncludeBundleOptionsForResequencing*
Value	*false*
Code	Leave the field blank

4. Click **Save**.

Managing Guided Selling

The Guided Selling feature has been deprecated since the CPQ Spring '21 release. The chapters related to the features have been removed as well.

Managing Promotions

The Conga Promotions package is available as an add-on with the CPQ license and package.

With the Conga Promotions package, you can incentivize the sales of products in your inventory. Promotions add value to product through discount pricing and add reward for purchase. Promotions provide value for money for your customers and contribute to better sales for your organization.

If you are a marketing manager or are responsible for promoting some products over others, you use historical transaction and sales data to create promotions. You can use promotions to:

- build traffic to the storefront
- create extra interest in product offering
- introduce new product, generate trial, or increase penetration in a specific market or segment
- increase sales of products during down-period
- provide competitive pricing to products during heavy competition
- increase revenue margins
- enhance customer retention
- clear stocks of a product that is about to reach end-of-life
- sell excess Inventory
- motivate staff

A promotion is a marketing technique that you apply to reduce the list price of a product or a service. A promotion consists of all non-tech activities that marketers use to try to increase sales over a specific period of time. You can create such promotions and restrict their scope, limit, and benefits so that your sales representatives can apply promotions to specific products, for specific customers, and for a limited period.

For more information on how promotions work, see the Promotions Workflow.

Usage of Promotions

In the Managing Promotions section, you can quickly find out how Promotions Management works and how you can manage your organization's business requirements with respect to Promotions.

If you are a Marketing Manager, you will learn to

- Create and edit promotion plans

- Submit promotions for approval
- Approve or reject requests for approval
- Analyse the impact and effectiveness of promotions on quotes and orders

If you are a Marketing Executive, you will learn to

- Analyze the Impact of promotion on quotes and orders
- Analyze the effectiveness of promotions

If you are a sales representative, see Applying Promotions on Line Items in the Cart, to learn how to

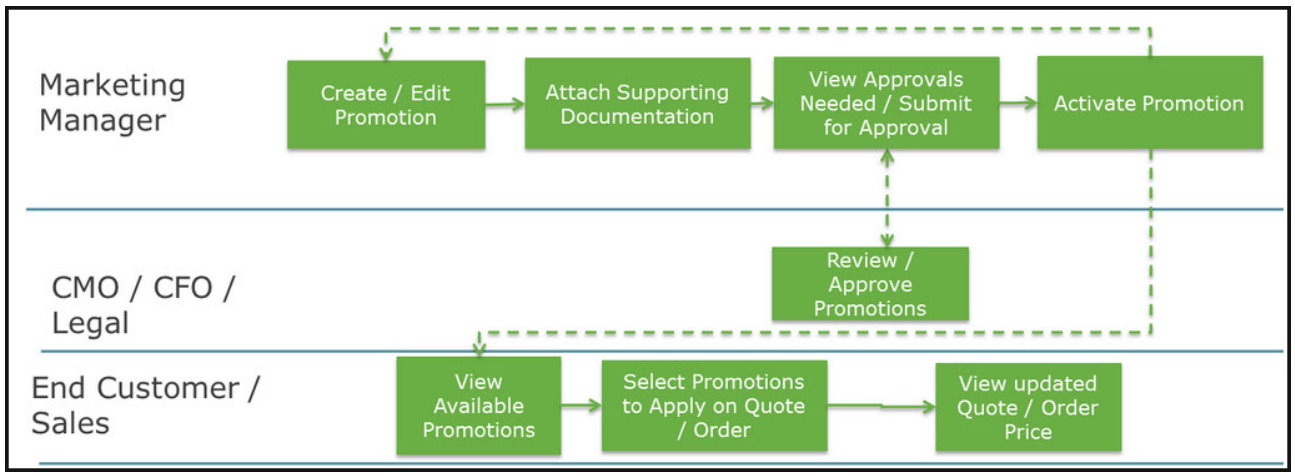
- View the available Pull promotions
- Add a promotion code and apply promotions

For more information on Promotion Management, see About Promotions Workflow.

About Promotions Workflow

For your organization to have a promotion event, the marketing team must first select the products to promote, define promotional pricing, and activate the promotion.

Promotional pricing may require approvals from your marketing manager and sometimes the finance and legal team.



Your sales representatives then can view the activated and available promotion to apply it on a quote or an order.

You can select from one or more different types of promotions that you want to create. For more information, see Types of Promotions.

Types of Promotions

The following are the major categories of promotions that you see on the Incentives page:

- Normal promotion: This is the default promotion that CPQ allows you to create. Promotions are applicable on the Cart.
- Order-level promotions: Order-level promotions are called rebates. For more information, see *Rebates Management on Salesforce Administrator Guide*.
- Sales promotion: A marketing manager or pricing manager can define promotions that can be grouped with the adjustments defined by the Sales Representative. A sales promotion allows the Sales Representative to modify the promotion in the shopping cart and structure discounts for deals. A sales promotion can be bucketed with the adjustments defined by the Sales Representative. Adjustments that are made to the Cart for a sales promotion are listed in the adjustment bucket.

To create a sales promotion

Perform the following steps to create a sales promotion:

1. Go to **Setup** > **Create** > **Objects** > **Incentives**.
2. In the **Buttons, Links, and Actions** related list, click **New Button or Link**. The New Button or Link page is displayed.
3. Enter the following details:
    a. **Label**: Enter a label for the button, for example, *New Sales Promotion*.
    b. **Name**: Enter a name for the button. This field is auto-populated based on the Label when you click inside the field.
    c. **Description**: Enter a description for the button.
    d. **Display Type**: Select a display type for the button, for example, *List Button*.
    e. **Behavior**: Select what should happen when the user clicks the button or link.
    f. **Content Source**: Select a content source, for example, *URL*.
    g. In formula editor, enter the following formula for the button:

```
/apex/Apttus_Config2__IncentiveManager?
DisplaySetting=Promotion&UseType=Promotion&SubUseType=Sales&DataSource=Apt
tus_Config2__OrderLineItem__c&ApplicationMethod=Buy X Get X
```

4. Click **Save**.
5. Add the new button to the layout of the Incentives page.

An action button is created on the Incentives page. You can now create a new sales promotion.

Furthermore, you can create the following types of promotion for each major category:

> ⓘ You can configure how the benefit must be applied on a selected product. For example, a promotion can be a discount in price or an increased quantity of product.

Promotion Type	Description	Example
Buy X Get X	When the customer purchases a product, you offer some benefit on the same product.	As a marketing manager of a mobile phone company, you need to promote a new smart phone. You create a promotion where you offer your customers 20% discount on the price of the phone that they buy.
Buy X Get Y	When the customer purchases a product, you offer some benefit on a different product.	As a marketing manager of an e-commerce company, you need to increase the sales of a refrigerator. You create a promotion where you offer your customers a free voltage stabilizer when they buy the refrigerator.
For every X Get X	When the customer purchases a certain quantity of a product, you offer some benefit on the same product.	As a marketing manager of a telecommunications company, you need to retain existing enterprise customers. You create a promotion where you offer your customers 10% discount on the price of each subscription for every 10 subscriptions that they buy.
For every X Get Y	When the customer purchases a certain quantity of a product, you offer some benefit on a different product.	As a marketing manager of a software company, you need to provide competitive pricing to a software as a service product during heavy competition. You create a promotion where you offer your customers free training for every 10 licenses that they buy.

Configuring Promotions

With the Promotions Management application, you can now manage, execute, and analyze promotions using the CPQ product line. With the Promotions Management application,

marketing managers can create new promotions, get internal approvals for such promotions, and roll these promotions to their sales channels.

You can use Conga Promotions to create and manage promotions like,

- buy X get X promotions
- tiered promotions
- automatically applied promotions
- coupon-based promotions

You can apply promotions on line item and on order levels. If you have applied promotion on order level, the promotion type is restricted to Buy X get X only.

⚠
- Line item promotions (Allow User Override = true) should not display incentives with Context Type = Summary Group.
- If the promotions is “push” promotion, the promotions are auto applied to the matching quote total or total summary group.
- If the promotions is “pull” promotion, the user can select the promotion and have it applied to the matching grant group total or total summary group.
- When the user override is enabled, only display Promotions, of which the context type is "line item.

For more information on how promotions work, see About Promotions Workflow and Types of Promotions

Preparing Data for Promotions

If you are a Marketing Manager, you will want to use as much data as you can get to strategize and create an effective and efficient promotion plan that guarantees fulfilling every business requirement. While you are already aware of several strategies to create an effective Incentive, you will use data and information such as:

- Price dimensions for a promotion
- Formula fields for lookup objects that you can reference in a promotion
- Child filters for the formula fields you define
- Approval workflows for a promotion
- Groups of promotions
- Create data roll-up

This data must already be available before you can implement or roll-out your promotion. This section explains how you can access data and then include or reference this data in the Incentives that you create.

Defining User Profiles and Permissions

If you are a CPQ administrator, you must first define who can access the Promotions configuration page and what actions they can perform on this page. You must create users and define permissions for each user of each profile as listed in the following table:

User	Profile	Permissions
Marketing Manager	• Create or edit promotion programs • Submit promotions for approval • Analyze impact of promotion on quotes or orders • Analyze promotion effectiveness	• Read or Write access on all objects used for promotions management • Read access on adjustment line items, Limit Data table, rollup data, and other related objects
Marketing Executives	• Analyze impact of promotion on quotes or orders • Analyze promotion effectiveness	• View access on Promotion objects, quotes, and proposals • Read access on adjustment line items, Limit Data table, rollup data, and other related objects
Promotion Approvers	• Approve or reject promotions • View or analyse promotions	• Read or Write access on promotion objects • View access on all objects used for promotions
Sales Representatives	Using promotions on quotes • View available Pull promotions • Add promotion code and apply promotion • View promotion benefit	View access on promotion objects, quotes, and proposals, adjustment line item objects, and Limit Data table

User	Profile	Permissions
Customers	Using promotions on the e-commerce orders • View available Pull promotions • Add promotion code and apply promotion • View promotion benefit	View access on promotion objects, quotes, and proposals, adjustment line item objects, and Limit Data table

For more information of user profiles and permissions, see About Permission Sets.

Configuring Incentive Records

With the help of custom settings, you can show or hide sections and fields of incentives that are not applicable for your business.

Hide or show sections in an Incentive record

For sections displayed in an Incentive record and more configurations, configure the following custom setting:

1. Go to **Setup > Develop > Custom Settings > Incentive Custom Display Setting**.
2. Click **Manage** next to **Incentive Custom Display Setting**.
3. Click **Edit** next to Promotions record. If there is no existing record, click **New**.
4. Enter the requisite details as follows:

Field	Description
Name	The mandatory name of the custom setting record. Enter *Promotions* here.

Field	Description
Navigation Pages	The names of the sections which will appear in an incentive record on the left pane of the page. The sequence and name of the sections are defined here. The Information section is defaulted for every Incentive record. S: This is for the Scope section. C: This is for the Criteria section. B: This is for the Benefit section. D: This is for the Deduction section. L: This is for the Limit section. P: This is for the Coupons section. For example, if you specify *SCBL*, only the Scope, Criteria, Benefit, and Limit section will be displayed in the given order.
Show Exclusion Criteria	This check box indicates if you want to include Exclusion Criteria along with the default Inclusion Criteria under Criteria section for an incentive record.
Application Type	The name of the incentive type where this custom setting record will be applied. Enter *Promotion*.

5. Click **Save**.

Configuring the fields displayed in an Incentive record

You can also control which standard and custom fields will be visible under different sections on your Incentive record using the following configuration.

1. Go to **Setup > Create > Objects > Incentive.**
2. Go to the **Field Sets** section and click **Edit** next to the name of the fieldset for which you want to modify the fields.
   For example, if you want to edit the fields displayed in the Information Section, click Edit next to **Information Section Fields**.
3. Search for your field, add or remove it from the fieldset.
4. Click **Save**.

Creating Price Dimensions for a Promotion

When you define the scope of your promotion, you must include or exclude fields such as Region, Country, and Account, and Account Type, that impact the pricing of the purchase. You must define a Price Dimension of one or more of these fields in Salesforce that you want to include as a parameter to define the scope of your promotion.

For example, you are a marketing manager with an Online University. You want to roll out a promotion for prospective subscribers based only in the United States. Before you can define the Geographical scope of this promotion, you must create a Price Dimension for the **Country** object.

To create a price dimension for the Country object

1. On **App Menu**, click **Apttus CPQ Admin** and then click the **CPQ Admin** tab. The new admin console is launched.
2. On the **Pricing** menu, click **Manage Price Dimensions**. A list of existing price dimensions is displayed.
3. Click **New Price Dimension**.
4. Enter the values for fields described in the following table.

Field	Description
Name	Enter a unique name for this Price Dimension.
Context Type	Select one of the following options. • Line Item - if this price dimension is only for the line item. • Formula Field - if this price dimension is based on the Field and Type you define below. • Child Filter - if this price dimension is based on filter criteria you define.
Business Object	Line Item (default)
Field	The field you select depends on the Context type. In this example, we selected Formula field and the objective is to create a price dimension for the Country field. The Country formula field for "Country" Price Dimension should point to the "CountryCode" field from the address as that is the field that has the pick-list associated with it.

Field	Description
Type	Select from one of the following options: The field **Type** is related to promotions. Based on the selected value, CPQ creates a dimension type for promotion. For example, you can create a quantity-based dimension by selecting *Quantity* from the **Type** drop-down. To set up values for the **Type** field, go to **Setup** > **Create** > **Objects** > **Price Dimension** (Apttus_Config2__PriceDimension__c) > **Type** (Apttus_Config2__Type__c) field. • **Standard** - for a standard promotion • **Quantity** - if the scope of your promotion is based on quantity • **Term** - if the scope of your promotion is based on the term of the subscription • **Amount** - if the if the scope of your promotion is based on the purchase amount • **Equipment** - if the if the scope of your promotion is based on equipment (Service CPQ)
Description	Describe the purpose of this price dimension so others in your organization can use it effectively.

5. Click **Save**.

For more information on Price Dimensions, see Creating Price Dimensions.

Creating Formula Fields for Lookup Objects

When you define the criteria of your promotions, you will require one or more lookup fields to fetch values from objects such as Quote, Order, Account, and Contact. For this purpose, you must create Formula fields that can reference the values you have defined for the fields in these objects.

Note

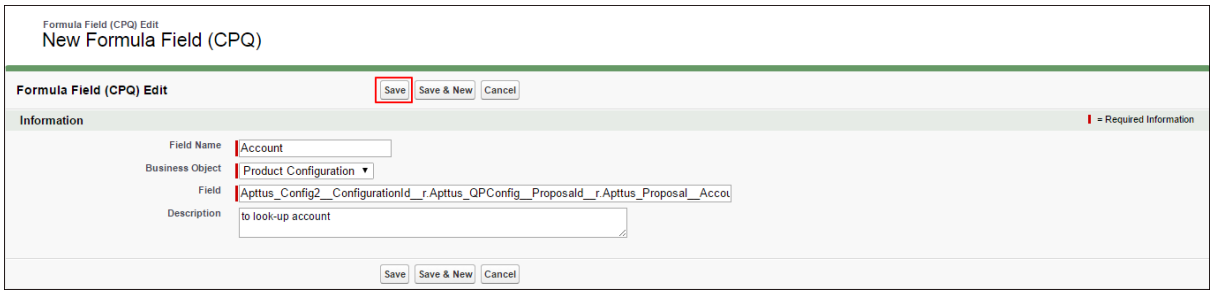
- Formula fields are not physical fields on the line item object.

- Once you setup a new Formula Field, be sure to run the **Criteria Maintenance > Update Pricing Fields**.

For example, you are a marketing manager who must define a promotion. You want to create and roll out a promotion that is applicable only to the healthcare industry. You must create a Formula field that references the Industry on the Criteria page. Here, we will create a formula field to create a look-up object.

To create a new Formula field

1. From the force.com app menu, select **Apttus CPQ**.
2. Click the **Formula fields** tab.



3. Enter values for the fields described in the following table.

Field	Description
Field Name	Enter a unique name for this new Formula Field.
Business Object	Select the business object you want to reference.
Field	Enter the API Name of the field that you want to look-up.
Description	Describe this formula field for the benefit of others in your organization.

4. Click **Save**.

Creating Child Filters

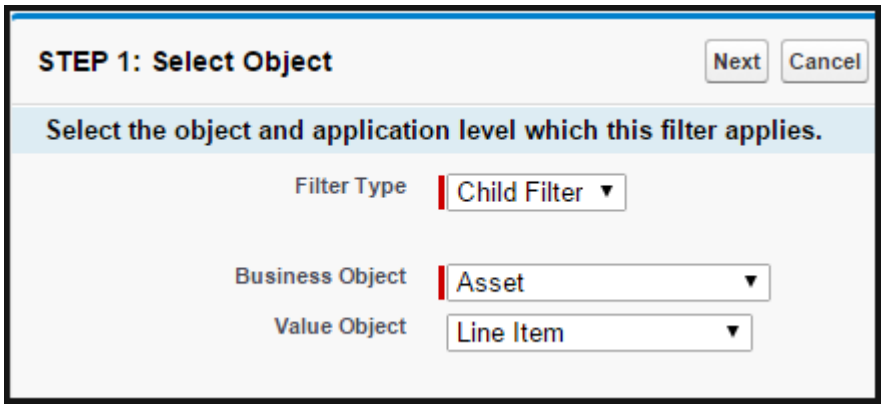
When you define the criteria of you promotion, you will require reference to Order Line item, Asset Line Item, Related Line Item, historical purchase summary, or other child objects of a look up object. These look-up objects can be Account, Contact, or User in the Promotion

Criteria. You will need the **Search Filter (CPQ)** of type Child Filter. Using these child filters, you can also create the Price Dimensions to use in Benefit Matrix.

For example, you are a marketing manager who must define a promotion. You want to create and roll out a promotion that is applicable only to one specific asset. You must create a filter so you can reference the specific asset and map it to a field and specify a definite value.

To create a Search Filter,

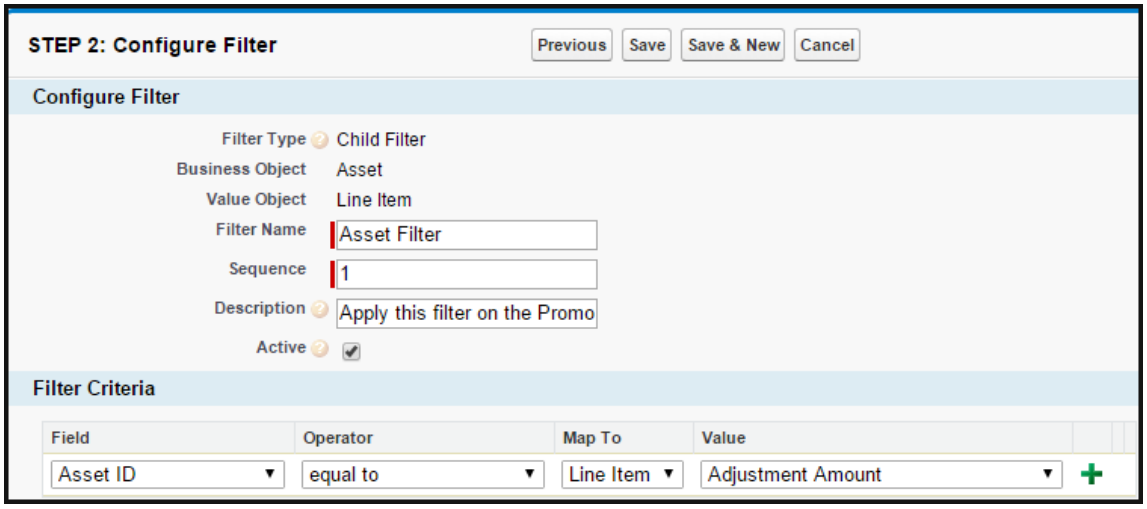
1. Click Search Filter (CPQ) > New.



2. Enter values for the fields described in the following table.

Field	Description
Filter Type	Select Child Filter.
Business Object	Select that Object you want to create a filter for.
Value Object	Select the object whose value you want to search for.

3. Click **Next**.



4. Enter values for the fields described in the following table.

Field	Description
Filter Name	Select Child Filter.
Sequence	Select that Object you want to create a filter for.
Description	Select the object whose value you want to search for.
Active	Select the check-box to activate this filter.
Filter Criteria	
Field	Select the field you want to reference.
Operator	Select the appropriate operator.
Map To	Select the field that you want to map your search filter to.
Value	Select the field whose value you want to populate in the promotions criteria.

5. Click **Save**.

Creating Approval Workflows for Promotions

You can define a hierarchy so that Promotions beyond a certain dollar amount or percentage require a manager's approval. CPQ allows you to selectively turn on or off approvals at a flow level. You must set the **isCartApprovalDisabled** parameter to True in the URL that you have configured on the **Configure Products** field of the respective quote. For more information, see Creating Custom Buttons for Different Flows.

To create an Approval workflow for Promotions,

1. From the force.com app menu, select **Apttus Approvals Management**.
2. Click **Approval Processes**.
3. For the **Manage Approval Process for** field, select Incentive.
4. Click **Create new Process**.
5. Enter values for the fields described in the following table.

Field	Description
Process Name	Enter a unique name to identify this specific Approval Process.
Description	Describe this process for others in your organization.
Entry Criteria	
Field	Select the field that you want to apply the approval trigger for.
Operator	Select the appropriate option.
Value	Enter the value that must trigger the request for approval.

6. Click **Next**.
7. According to your Business requirements, select any one or more of the following check boxes.
    - **Consolidate Approvals**
    - **Consolidate Notifications**
    - **Continue Pending approvals on a Reject**
8. Assign an Email Template.
9. Assign an Approvals page.
10. Assign a Back-up Admin user.
11. Click **Save**.

For more information, see Approval Functional End User Workflow.

Configuring Data Rollups for Promotions

If you are a Marketing Manager, you will require historical purchase data of customers and what products they have purchased. This information helps you effectively strategise and plan specific details for your promotion plans.

The purpose of a data roll-up is to convert different categories of historical purchase information into variables. You can use a data roll-up to generate a result that aggregates a hierarchy of values for one or more parameters.

For example, you are a Marketing Manager for a global electronics and consumer durables company. One of the products you want to promote is televisions. You must consider

purchase history patterns of your customers from different geographical locations before you begin to strategize your promotion plan for each location. The purchase, pricing, and applied discount patterns for TVs varies greatly from one region to another. Notably, customers in the United States are more inclined to purchase TVs and other consumer durables for Thanksgiving which is in November. Your customers in China on the other hand, are inclined to make such purchases during the Chinese New Year which is celebrated in February. There are more subtle differences such as your customers' brand loyalties that you can learn about using a data roll-up of historical purchases.

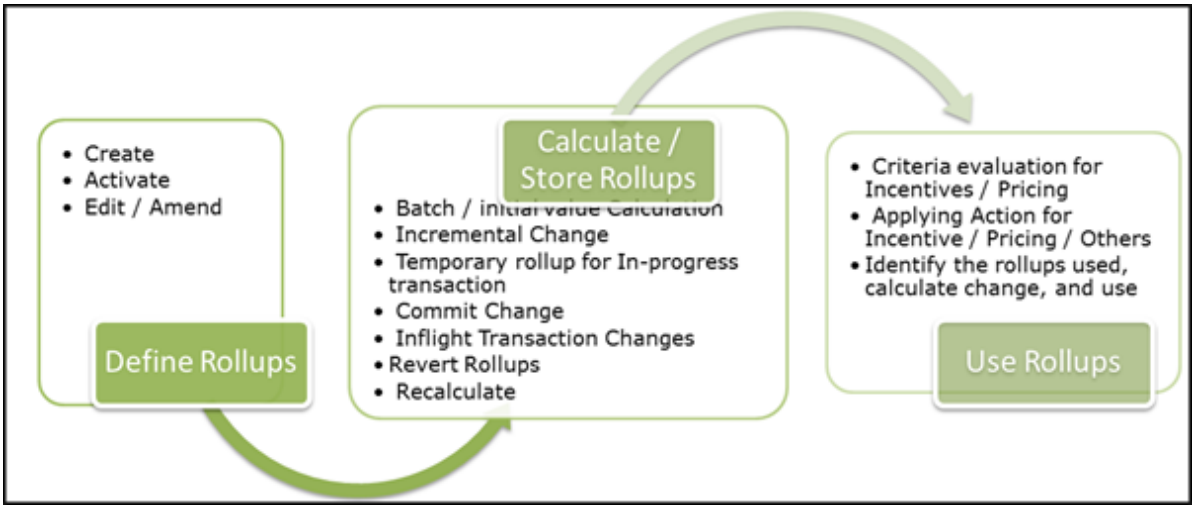
With Apttus Incentive Management, you can base your promotions on historical information such as how much your customers spent on specific products in a specific period.

The following table lists the factors you must consider and define to create a data roll-up.

Component	Description
Data Source	The **Order line items** and the **Cart line items** that your customers purchased are the source of all your historical purchase information.
Roll-up period	The period for which you want to measure the sales of a specific product or service. You must define this period with a specific Start Date and an End Date.
Roll-up function	A function is the operations you can perform with the data roll-up. At this time, **SUM** is the only function you can use.
Roll-up metrics	The metrics are factors that determine your customers' purchase behaviour. The two metrics you can use are • **Quantity** – the number of each product sold. You can use the Quantity metric to plan for inventory stocks, forecast projected sales and strategise an effective promotion. • **Amount** – the amount for which you sold each product. You can also use the Amount metric to measure the revenue you earned from each product sale and forecast revenues for the new promotion plan you want to define.
Roll-up criteria	The criteria the basis on which you roll-up historical purchase data. You can use data directly or reference it from the **Account**, **Product**, and **Contact** fields or their child records.
Group by	A groups is the parameter that defines how you categorize information in your data roll-up.

Roll-up scope	The scope defines the limits of your data roll-up. The scope is a mechanism to filter the list of the data roll-up that you must evaluate for a specific order.

Data Rollup Workflow



You must create a data roll-up to define the criteria, benefits, and the dimensions of a promotion plan.

Creating a Data Rollup

To create a new data roll-up,

1. Go to the Salesforce App Menu and select **Apttus Incentive Setup**.
2. Click **Incentive Rollups**. The Incentive Rollups page with existing rollups is displayed.
3. Click **New**. The Rollup Rules page is displayed.
4. In the **Rollup Information** section, enter values for the fields described in the following table.

Field	Description
Filter Name	Enter a unique name for this filter. Consult your administrator for the naming conventions you must follow.
Description	For others in your organization to learn and use this data roll-up, describe the purpose and function of this data roll-up.
Active	Select the **Yes** radio button to activate this data roll-up and make it available for reference in your Promotions.

Rollup Effective Date	Select a date from when the rollup is effective.
Rollup Expiration Date	Select a date when the rollup will expire.

5. In the **Rollup Value** section, enter values for the fields described in the following table.

Field	Description
Rollup Type	Select **SUM** from the drop-down menu.
Business Object	Select from one of the following options. • Asset • Asset Line Item • Product • Product Attribute Value • Product Configuration
Rollup Metric Type	Select from one of the following options. • **Quantity** - the number of each product sold. • **Amount**- the amount for which you sold each product.
Rollup Field	Select from one of the following options. • Order Line Item • Cart Line Item

6. In the **Rollup Duration** section, enter values for the fields described in the following table.

Field	Description
Rollup Frequency	Select from one of the following options. • One Time – for purchases that require only a one-time payment • Recurring – for purchases that are subscriptions or usage-based.
Rollup Period Source	The source of the period depends is the **Business Object** you defined in the Rollup Value section.

<table>
<tr>
<td>Rollup Start Date Type</td>
<td>If you select Constant, only the Rollup Start Date is visible. Enter a start date for the rollup.

If you select Reference Value Source, the following fields are visible. Enter values for those fields.
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Data Source</td><td>The source of the data you want to roll-up is the Order Line Item or a child filter for which the value you define must be the Order Line Item.</td></tr>
<tr><td>Field</td><td>The field is the parameter that determines the period for which you want to generate the data roll-up.

Select from one of the following options.
• Created Date
• Effective Date
• Expiration Date
• Last Modified Date
• Last Referenced Date
• Last Viewed Date
• System Modstamp</td></tr>
<tr><td>Offset Type</td><td>Offset values that you define determine the start date of the period for which you are creating a data roll-up.

Select from one of the following options.
• Offset (+) Days
• Offset (-) Days
• Offset (+) Months
• Offset (-) Months
By default, the Value is constant.</td></tr>
<tr><td>Offset Value</td><td>Enter the number of days or months by which you want to offset the start date of the data roll-up.</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>Rollup End Date Type</td>
<td>If you select Constant, only the Rollup End Date is visible. Enter an end date for the rollup.

If you select Reference Value Source, the following fields are visible. Enter values for those fields.
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Data Source</td><td>The source of the data you want to roll-up is the Order Line Item or a child filter for which the value you define must be the Order Line Item.</td></tr>
<tr><td>Field</td><td>Select from one of the following options.

• Created Date
• Effective Date
• Expiration Date
• Last Modified Date
• Last Referenced Date
• Last Viewed Date
• System Modstamp</td></tr>
<tr><td>Offset Type</td><td>Offset values that you define determine the end date of the period for which you are creating a data roll-up.

Select from one of the following options.

• Offset (+) Days
• Offset (-) Days
• Offset (+) Months
• Offset (-) Months
By default, the Value is constant.</td></tr>
<tr><td>Offset Value</td><td>Enter the number of days or months by which you want to offset the end date of the data roll-up.</td></tr>
</table>
</td>
</tr>
</table>

7. In the **Rollup Group By Parameters** section, enter values for the fields described in the following table.

Field	Description

Group by Parameter	This parameter defines how you will categorize information from the data source in your data roll-up. Lookup and search the field that you want to select.
Related field on Rollup Data	Lookup and search the related field that you want to select for your data roll-up.

8. In the **Scope** section, enter values for the fields described in the following table.

Field	Description
Note: To include the following fields in the scope of your data roll-up, select the **Include** Radio button.	
Product Family	Lookup and search for the product families that you want to include or exclude from the scope of your data roll-up.
Products	Lookup and search for the products that you want to include or exclude from the scope of your data roll-up.
Region	Lookup and search for the regions that you want to include or exclude from the scope of your data roll-up.
Country	Lookup and search for the countries that you want to include or exclude from the scope of your data roll-up.
Account Type	Lookup and search for the Account Types that you want to include or exclude from the scope of your data roll-up.

9. In the **Inclusion Criteria** section, enter values for the fields described in the following table.

Field	Description
Field	Select an option from this list of fields to which the promotion must be applied. The fields available depend on what **Reference Data Source** you selected while defining the information.

Operator	If the field you defined is a pick-list or a lookup field, you can select one of the following operators. • equal to • not equal to • in • not in For any other field you defined, select one of the following operators. • equal to • not equal to • less than • less than or equal to • greater than • greater than or equal to • contains (displayed only if you selected a field of type text or area such as Description)
Value Type	If you select *Constant* (default), only **Value** is displayed. If you select *Field Value*, **Data Source** and **Field** are displayed. If the field you selected is date, you can select from one of the following options: • Offset (+) Days • Offset (-) Days • Offset (+) Months • Offset (-) Months
Value	Enter a simple constant value or select single or multiple values using search widget depending on the field type and operator you defined for the primary data source. If you selected *Product* from the **Field** drop-down, you can search a product by its name or code in the **Value** field.
Data Source	Select one from the list of Search Filter CPQ data sources linked to Line Item you selected as the primary data source. With this data source, you can use the information from child objects to create evaluation criteria that can be based on previous purchase history such as order or asset line items, or roll-up data.
Field	Select one from the list of fields from the child filter object.
Criteria Expression	Apttus Incentive Management populates this field when you enter values for the fields that defines the scope.

10. Click **Save**.

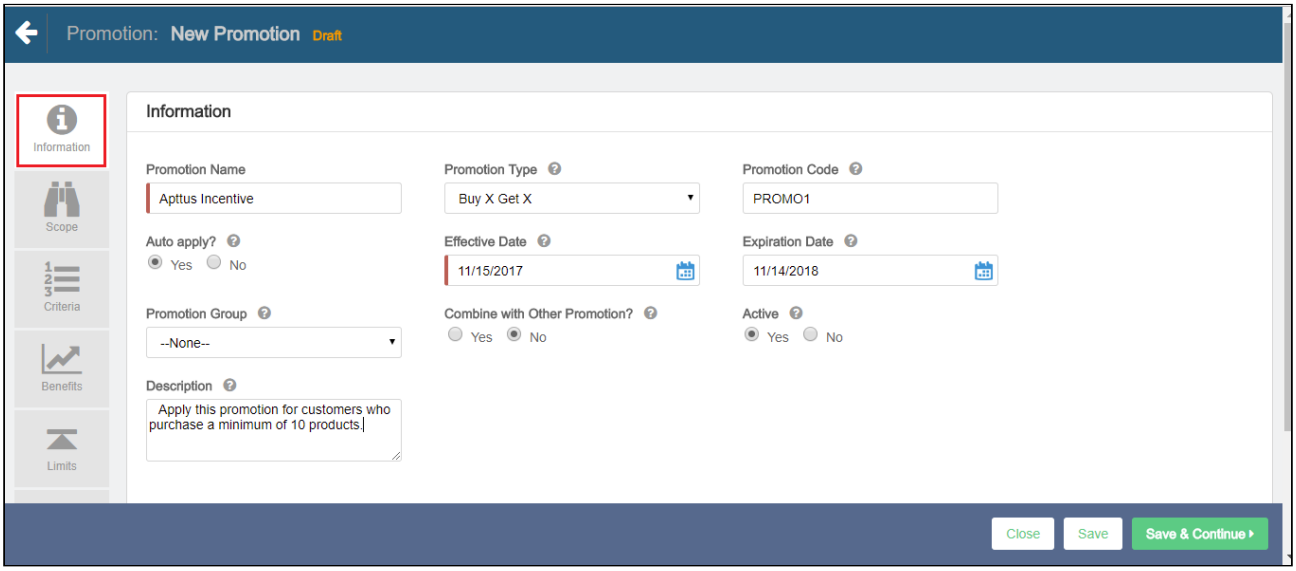
Defining Promotions

Every promotion you create has the following aspects that you must define:

- Promotion Information
- Promotion Scope
- Promotion Criteria
- Promotion Benefits
- Promotion Limits

Each aspect of a promotion is dependent on certain objects and fields that you must configure or define before you can create your first promotion. You must complete the prerequisite tasks before creating your first promotion plan.

Defining the Information of a Promotion



To define a new promotion,

1. Go to the Salesforce App Menu and select **Apttus Incentive Setup**.
2. Click **Incentives** > **New**.
3. On the Incentives information page, define values for the fields described in the following table.

Field	Description
Promotion Name	User-specific name for the promotion. This name is displayed to the user at any place the promotion information is displayed. For example, whenever a user looks-up the promotion information applicable to a specific product, the promotion name you provide here is displayed.
Promotion Type	Different types of promotions. Supported types are: • Buy X Get X • Buy X Get Y • For Every X Get X • For Every X Get Y • Own Every X Get Y To configure what options to be displayed in this drop-down, go to **Setup** > **App Setup** > **Create** > **Objects** > **Incentive** > **Application Method** (API name: Apttus_Config2__ApplicationMethod__c) > **Values** section.
Reference Data Source	You can apply promotions on line item and on order levels. Select from the following options: • Line Item: To apply a promotion on line item level. This line item is the primary object to which your promotion is applied. • Summary Group: To apply promotion on the order level. Note: For Summary Group, the promotion type is Buy X get X only.
Promotion Code	This is the code that is used for applying promotion. This code is captured on the order or proposal line items when a promotion is applied to the line item. This can also be used by the marketing team for marketing campaigns. When a promotion is not auto-applied, that is when the **Auto Apply?** checkbox is cleared, the user can use the incentive code to avail a promotional offer.
Auto Apply?	Can be set to Yes or No. Set it to *Yes* when you want a promotion to be automatically applied when a promotion criteria is met. Set it to *No* when you want the user to manually enter an Incentive Code to apply promotions.
Effective Date	Date from which the promotion is applicable. Cannot be blank. By default, it is set to current date (date when you create the promotion record).
Expiration Date	Date on which the promotion expires. Can be blank. Needs to be greater than Start Date. By default, it is set to 1 year from the Effective Date.

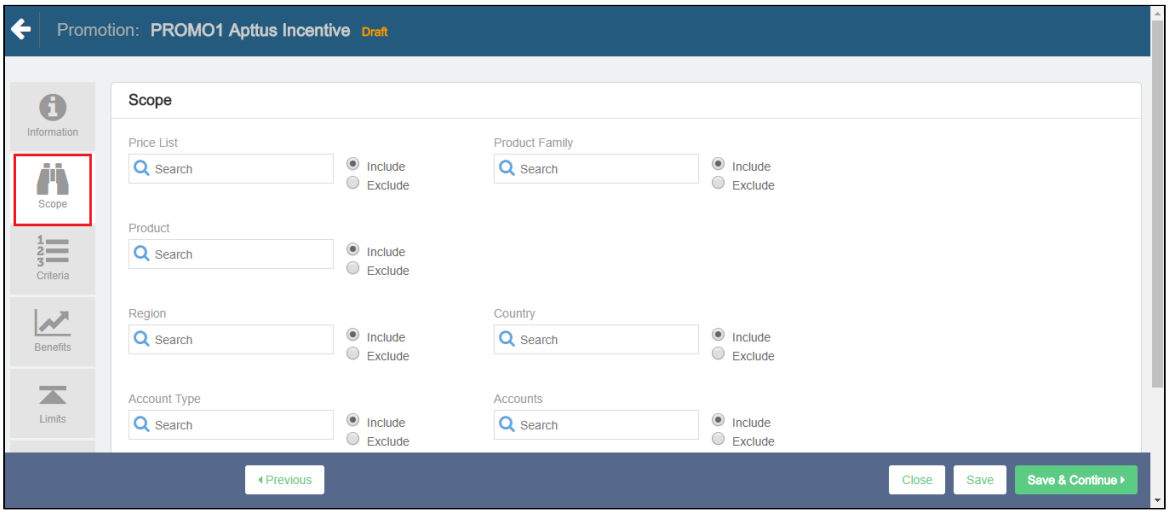
Field	Description
Promotion Group	This is a generic grouping of the Incentives for analyzing and tracking the incentives. The user can select a promotion group for each promotion. The list of Promotion Groups needs to be defined separately in the Incentive Group object and should be active. Marketing managers can create reports and track the incentive groups that are more successful than others and drive the business strategy accordingly.
Combine with Other Promotion?	Enables you to combine the new promotion with an existing promotion.
Active?	Can be set to *Yes* or *No*. Set it to *Yes* when you want a promotion to be applied and active for the criteria you specify. Active flag on the promotions will be set using a workflow linked to approvals similar to what you would do for any other custom object approval. For example, you can write a workflow rule to set the active status to true once the promotion is Approved (if approval is set up). Approvals set up would be similar to custom object approvals for any object.
Description	User-entered description for the promotion.

4. Click **Save** to save the changes or click **Save & Continue** to move to the next tab.

After you have described the promotion, you must now define the scope of this new promotion.

Defining the Scope of a Promotion

You must define the scope of a promotion to determine the types of accounts, the regions, and the products that this promotion will be available to.



To define the scope of your new promotion,

1. On the left side work pane, click **Scope**.
2. Define values for the fields described in the following table.

> **Note**
>
> The default value for each field is **All**. If you do not select a value, CPQ performs evaluation on all items for each field and that can impact the performance.
> By default, the fields you define are set to be included in the scope of your promotion. Select the **Exclude** radio button for any parameter if you want to exclude that specific object from the scope of your promotion.

Field	Description
Price List	Select the price list that you want to use for your promotion.
Product Family	Select the Product Family that you want to apply the promotion to.

Field	Description
Product	Select the specific products that you want to apply the promotion to. You can search a product by its name or code. If you selected **Promotion Type** as *Buy X Get X* on the Information page, you will see the following options: • **Include (OR)**: CPQ applies promotion on all products (in scope), which are available in the cart. • **Include (AND)**: CPQ applies promotion on all products (in scope) only if the products are added to the cart. • **Include Bundle Options (AND)**: CPQ applies promotion to a bundle and options only if the bundle is configured with specified options. • **Exclude**: The promotion will not be applied to the products specified. ⓘ To see these options on the UI, you must update **Setup** > **App Setup** > **Create** > **Objects** > **Price Rule** object > **Product Criteria Oper** field. • To use **Include (OR)** scope, update the **Product Criteria Oper** field as "in". • To use **Include (AND)** scope, update the **Product Criteria Oper** field as "all". • To use **Include Bundle Option (AND)** scope, update the **Product Criteria Oper** field as "all in bundle". If you selected **Promotion Type** as *Buy X Get Y* on the Information page, you will see the following options: • **Include (OR)**: If one of the specified products is included in the cart, CPQ applies the promotion to the benefit product. • **Include (AND)**: If all specified products are included in the cart, CPQ applies the promotion to the benefit product.

Field	Description
	• **Include Bundle Options (AND)**: If the bundle and all specified option products are included in the cart, CPQ applies the promotion to the benefit product. • **Exclude**: CPQ does not apply the promotion to the products specified.
Region	Select geographical regions of the customers that you want to make this promotion available to.
Product Group	Select the Product Group that you want to apply the promotion to.
Country	Select countries where you want to make this promotion available.
Account Type	Select the types of accounts for which you want to make this promotion available.
Accounts	Select the accounts for which you want to make this promotion available.

3. Click **Save** to save the changes or click **Save & Continue** to move to the next tab.

After the scope, you must now define the criteria for this promotion.

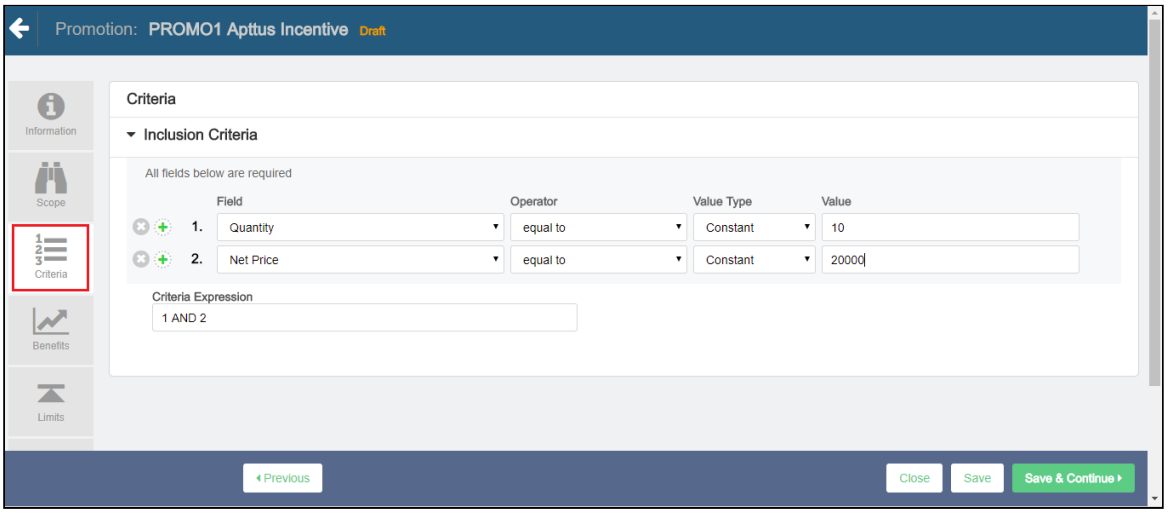
Defining the Criteria of a Promotion

You must define the criteria for a promotion before making it available to customers. If a sales representative wants to apply this promotion to a customer's purchase, the criteria you define here will determine if that purchase qualifies for the promotion.

You can define two types of criteria based on:

- any the past purchases of customers or rolled up metrics of past purchases.
- data from the line item, header, lookup objects, and child objects of the lookup objects for a quote or an order.

You can also define additional filter criteria to determine which products this promotion applies to.



To define the scope of your new promotion,

1. On the left side work pane, click **Criteria**.
2. Define values for the fields described in the following table.

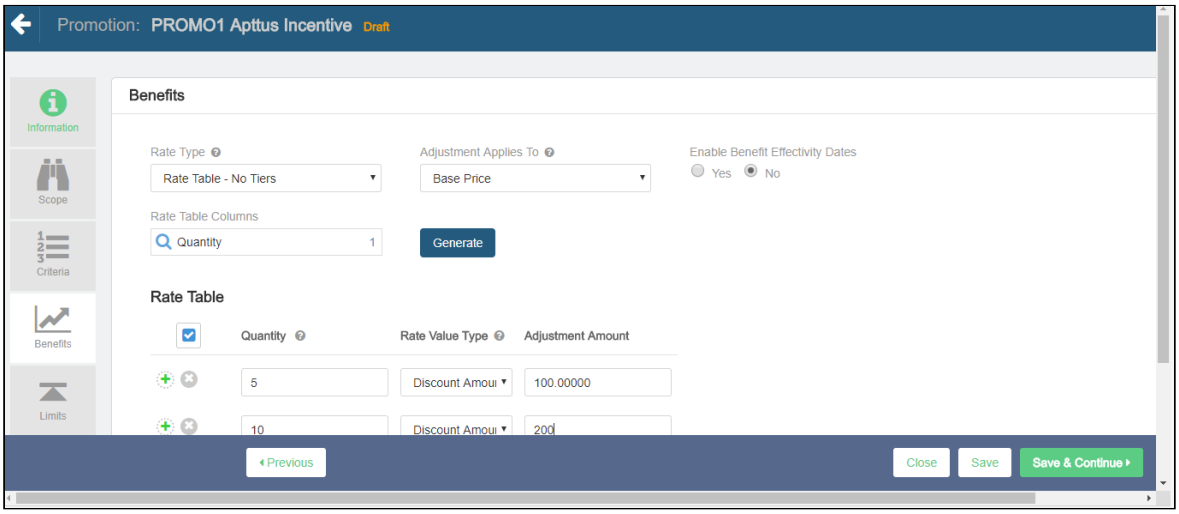
Field	Description
Field	Select an option from this list of fields to which the promotion must be applied. The fields available depend on what **Reference Data Source** you selected while defining the information.
Operator	If the field you defined is a picklist or a lookup field, you can select one of the following operators. • equal to • not equal to • in • not in For any other field you defined, select one of the following operators. • equal to • not equal to • less than • less than or equal to • greater than • greater than or equal to • contains (displayed only if you selected a field of type text or area such as Description)

Field	Description
Value Type	If you select *Constant* (default), only **Value** is displayed. If you select *Field Value*, **Data Source** and **Field** are displayed. If the field you selected is date, you can select from one of the following options: • Offset (+) Days • Offset (-) Days • Offset (+) Months • Offset (-) Months
Value	This field is visible only if you select *Constant* from the **Value Type** drop-down. Enter a simple constant value or select single or multiple values using search widget depending on the field type and operator you defined for the primary data source. If you selected *Product* from the **Field** drop-down, you can search a product by its name or code in the **Value** field. The search results display the product name and code wherever applicable. If a product has a code defined, the search result displays name and code; else the search result displays only the name.
Data Source	This field is visible only if you select *Field Value* from the **Value Type** drop-down. Select one from the list of Search Filter CPQ data sources linked to Line Item you selected as the primary data source. With this data source, you can use the information from child objects to create evaluation criteria that can be based on previous purchase history such as order or asset line items, or roll-up data.
Field	This field is visible only if you select *Field Value* from the **Value Type** drop-down. Select one from the list of fields from the child filter object.

3. Click **Save** to save the changes or click **Save & Continue** to move to the next tab.

You must now define the benefits of this promotion.

Defining the Benefits of a Promotion



To define the benefits of a promotion,

1. On the left work pane, click **Benefits**.
2. From the **Benefits** section, enter the following details:

> In the Benefits section, the **Rule Name**, **Product Family**, and **Product** fields are available only if you selected the *Buy X Get Y* promotion.
> In the Benefits section, the **For Every X Criteria**, **Benefit Product**, and **Benefit Type** sub-sections are available only if you selected the *For Every X Get X* and *For Every X Get Y* promotions.

   a. In the **Rule Name** field, enter a name for the benefit rule.
   b. From the **Product Family** lookup, search and select the required product families.
   c. From the **Products** lookup, search and select the required products. You can search a product by its name or code. The search results also display product name and code wherever applicable. If a product has code defined, the search result displays name and code; else the search result displays only the name.

> For Buy X Get Y promotions, if the charge types of scope and benefit products are different in price list items, you must select a charge type for the benefit product in the **Charge Type** multiselect box. Otherwise, CPQ does not apply the promotion on the benefit product. CPQ applies

the Buy X Get Y promotion to the benefit product regardless of the charge type present on the product you have selected in the Scope and Benefit sections.

Use the following operators to define the benefit for multiple products:

- **Or**: CPQ applies promotion only to the first product that matches the criteria (selected by default).
- **And**: CPQ applies promotion to all products on the benefit criteria. All benefit products must exist on the cart to apply the promotion.

d. In the **Charge Type** multiselect box, select the required charge type for the benefit products if the charge types of scope and benefit products are different in price list items.

> The **Charge Type** multiselect box is applicable only for Buy X Get Y type of promotions. The charge type of the product you selected from the **Product** drop-down in the Scope section does not have any impact on the charge type you select for benefit product in the **Charge Type** multiselect box in the Benefit section.

- If you select multiple benefit products from the **Products** lookup, you must select multiple charge types in the **Charge Type** multiselect box depending on their charge types in price list item.
- If you select only one benefit product from the **Products** lookup and its charge type is same of the charge type of the scope product, CPQ applies promotion on the benefit product even if you do not select a charge type in the **Charge Type** multiselect box.
- If you select only one benefit product from the **Products** lookup and its charge type is different from the charge type of the scope product, you must select a charge type in the **Charge Type** multiselect box. Otherwise, CPQ does not apply promotion on the benefit product. For example, the charge type of the scope product is Standard Price and the charge type of the benefit product is License Fee, you must select License Fee for the benefit product in the **Charge Type** multiselect box.
- If the benefit product has two charge types in price list item, you must select only one of the charge types in the **Charge Type** multiselect box depending on which charge type you want CPQ to apply the promotion on the benefit product.

e. Enter the details in the following subsections:

> In the Benefits section, the **For Every X Criteria**, **Benefit Product**, and **Benefit Type** sub-sections are available only if you selected the *For Every X Get X* and *For Every X Get Y* promotions.
> You can create a promotion of type *For Every X Get X* where X can be multiple products or any product from a group. For example, Mobile Phones is a group of products consisting ABC 13, ABC 13 Pro, and ABC 13 Max phones. You create a *For Every X Get X* promotion for Mobile Phones as "for every 5 mobile phones get 20% discount on the 6th mobile phone". When the users add any product from Mobile Phones (that are defined in the criteria of the promotion) to the Cart; CPQ applies the defined *For Every X Get X* promotions to those products. For example, if the user adds 6 ABC 13 and 6 ABC 13 Max phones to the Cart, CPQ provides 20% on the 6th 6 ABC 13 and 6th ABC 13 Max phones. However, the users must fulfil the quantity criteria for each product to get the benefit for that product. For example, the users must add 5 ABC 13 mobile phones to get 20% discount on the 6th ABC 13. The users will not get any benefit if they add 4 ABC 13 and 2 ABC Pro phones to the Cart.
> From Conga CPQ May '22 release, CPQ does not add the criteria product on the Price Rule entry object. All products in scope are eligible for promotions. After upgrading Conga CPQ to May '22, the existing or previously defined *Every X Get X* promotions do not work with multiple products. You must create a new promotion to support multiple products. However, the existing *Every X Get X* promotions will work for a single product according to their current implementation.

   i. **Criteria Value**: Enter the criteria value on the scope product for the promotion to be applied on the benefit product. The criteria value determines how many quantities of the scope product the user must buy for the promotion to be applicable on the benefit product.
      - **Type**: Select Quantity.
      - **Metric**: Select Quantity.
      - **Value**: Enter a numerical value.
   ii. **Benefit Product**: Select the benefit product on which promotion must be applied.
   iii. **Benefit Type**: Select Price and Quantity.
   iv. **Benefit Quantity**: Enter a numerical value. The benefit quantity determines on how many benefit products the promotion is applicable.

3. From the **Rate Type** drop-down, select a value.

The values on the **Rate Type** drop-down are based on what you add in **Setup** > **App Setup** > **Create** > **Objects** > **Price Rule** > **Rule Type** > **Values**. The same values are available on the **Rule Type** drop-down during price rule creation. Some values that you add for **Rule Type** cannot be used for **Rate Type** on the Benefit section of promotion. Ensure that you select only the supported values while creating a price rule and while defining promotions. On the **Rate Type** drop-down, only Single Rate, Rate Table - No Tiers, and Rate Table - With Tiers are supported.

If you remove some values that are not supported for **Rate Type** from the **Price Rule** object > **Rule Type** field, you cannot see the removed values on the **Rule Type** drop-down while creating a price rule.

Rate Type	Description
Single Rate	Selecting this option allows you to define lump sum benefits which do not depend on transaction volume or amount. This is an ideal option when your promotion is not dependent on the amount or quantity of the line items. Select this value if your promotion calculation has no dependency on the amount or quantity ordered. Example, a seasonal promotion that offers 15% off for smartphones. If you select this Rate Type, enter the following values:

<table>
<tr><th>Rate Type</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Rate Value Type</td><td>Select a value to define how you want to apply the benefit. The supported values are:
% Discount: A percentage discount specified in the Adjustment Amount field is applied on the option selected in Adjustment Applies To.
Discount Amount: A discount amount specified in the Adjustment Amount field is applied on the option selected in Adjustment Applies To.
% Markup: A percentage markup specified in the Adjustment Amount field is applied on the option selected in Adjustment Applies To. For example, if you want to increase the price of laptops during the month of September, you can apply a percentage markup that adds to the List price.
Markup Amount: A markup amount specified in the Adjustment Amount field is applied on the option selected in Adjustment Applies To. For example, if you want to increase the price of laptops during the month of September, you can apply a markup amount that adds to the List price.
Price Override: The amount specified in the Adjustment Amount field is overrides the option selected in Adjustment Applies To.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Rate Type</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Adjustment Applies To</td><td>The value of this field controls to which price point the adjustments are applied. For example, whether the price discounts are applied to the total net price or the starting price.</td></tr>
<tr><td>Adjustment Amount</td><td>Enter the adjustment amount.</td></tr>
<tr><td>Benefit Uom</td><td>Select the benefit unit of measure for the product.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Rate Type</th><th>Description</th></tr>
<tr><td>Rate Table - No Tiers</td><td>Allows you to conditionally define promotional amount. Examples of conditions are products, regions, account types.

If you select this Rate Type, enter the following values:

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Benefit Uom</td><td>Select the benefit unit of measure for the product.</td></tr>
<tr><td>Adjustment Applies To</td><td>The value of this field controls to which price point the adjustments are applied. For example, whether the price discounts are applied to the total net price or the starting price.</td></tr>
<tr><td>Enable Benefit Effectivity Dates</td><td>Select Yes to define the start and end dates for the benefit. Select the dates in the Effective Date and Expiration Date lookup fields.</td></tr>
<tr><td>Rate Table Columns</td><td>This field is used to define the conditions to be used with the rate table.

If you need to define different promotions for different products, make sure the product column is added. If you would like to add another condition, make sure to select the value from the picklist and click the Generate button.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Rate Type</th><th>Description</th></tr>
<tr><td>Rate Table - With Tiers</td><td>Selecting this option allows you to define promotional amount that depends on transaction volume or amount. If you select this Rate Type, enter the following values:
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Rate Table - Tier Type</td><td>The value of this field controls how the promotion is calculated.

Rate Per Each Tier - Promotion is calculated at different pricing levels. The promotion amount for each unit depends on which tier the unit falls under. This is the most common scenarios for customer promotion calculation.

Highest Tier Rate - Promotion is calculated based on the total volume or revenue. Each unit receives the same promotional amount.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Rate Type</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Tier Metric Type</td><td>The value of this field controls which metric will be used to qualify for tiers.

Volume - Volume or quantities are used to qualify for tiers. Select this value if your promotion varies based on the purchase quantities but has no dependencies on the dates.

Volume with Dates - Volume and dates are used to qualify for tiers. Select this value if your promotion varies based on the purchase quantities as well as the when they purchased.

Revenue - Total quote amount is used to qualify for tiers. Select this value if your promotion varies based on the revenue amount but not on the dates.

Revenue with Dates - Total quote amount and dates are both used to qualify for tiers. Select this value if your promotion varies based on the purchase amount as well as the when they are purchased.</td></tr>
<tr><td>Benefit Uom</td><td>Select the benefit unit of measure for the product.</td></tr>
<tr><td>Metric Calculation Period</td><td>Select a period over which the tier metric is aggregated. The supported value is No Rollup, which means the tier metric is based on the value from individual transaction only.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Rate Type</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Metric Value Source</td><td>The value from this field determines the source based on which the volume or revenue is calculated. The calculated volume or revenue amount is used to determine the tier.

Select this value if the Tier Metric Type is Volume. Once Quantity is selected, you will see that in the Rate Table Structure, the “from” and “to” columns become “From Quantity” and “To Quantity”.</td></tr>
<tr><td>Adjustment Applies To</td><td>The value of this field controls to which price point the adjustments are applied. For example, whether the price discounts are applied to the total net price or the starting price.</td></tr>
<tr><td>Rate Table Columns</td><td>This field is used to define the conditions to be used with the rate table.

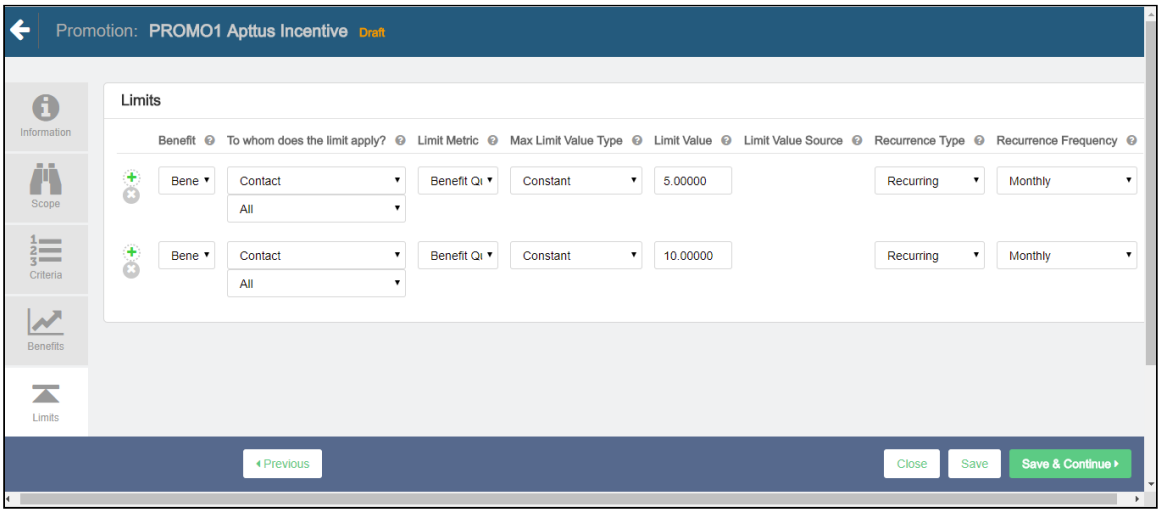
If you need to define different promotions for different products, make sure the product column is added. If you would like to add another condition, make sure to select the value from the picklist and click the Generate button.</td></tr>
</table>
</td></tr>
</table>

4. Click **Save** to save the changes or click **Save & Continue** to move to the next tab.

With the basic information, scope, criteria, and benefits defined, you must now define the limits of this promotion.

Defining the Limits of a Promotion

You must define promotion limits to determine the number of times you customer can apply a promotion for different types of the transactions.



To define the limit of a promotion,

1. On the left work pane, click **Limits**.
2. Click **Add** to add a new limit and **Delete** to delete a limit.
3. Enter values for the fields described in the following table:

Field Name	Description
To Whom Does the Limit Apply?	Enables you specify the object on which the limit is applied. The objects on which limit is applicable on are *Account*, *User*, and *Contact*.
Limit Metric	Specifies what will be the parameter on which the limit is applicable. *Benefit Quantity*: Specifies that the discount would be applied on a fixed quantity.
Max Limit Value Type	Specifies whether a limit would be a constant value or derived from some formula field.

Field Name	Description
Limit Value	Specifies the value of the Limit Metric. For example, if you specify the limit value as 10 and the limit metric as Benefit Quantity, the user can use the same promo code when the Quantity of the product is 10. If you enable **Setup** > **Build** > **Develop** > **Custom Settings** > **Incentive System Properties** > **Manage** > **Enforce Limits**, when a quote is finalized, CPQ applies the incentive limit across quotes in the same account. For example, if **Limit Value** is *5* and there are two quotes with three cart line items each, CPQ applies the incentive limit on three items in the first quote and two items in the second quote. If you disable **Setup** > **Build** > **Develop** > **Custom Settings** > **Incentive System Properties** > **Manage** > **Enforce Limits**, only when a quote is accepted, CPQ applies the incentive limit on each quote of an account. For example, if **Limit Value** is *5* and there are two quotes with five cart line items each, CPQ applies the incentive limit on five items in the first quote and five items in the second quote. CPQ resets the incentive limit for each quote. ⓘ The incentive limit is applicable when users finalize the cart. If they abandon the cart where promotion is applied, CPQ resets the incentive limit.
Recurrence Type	Specifies whether the limit applied is one time or on a recurring basis.
Recurrence Frequency	Specifies the frequency at which the applied incentives are limited. Available frequencies are *Monthly*, *Quarterly*, *Half Yearly*, *Yearly*, *Order*.

4. Click **Save** to save the changes or click **Save & Continue** to move to the next tab.

Multiple promotion Limits

- A given promotion can have more than one limits and all these limits needs to be enforced for the promotions. E.g. Promotion limits on a single promotion can be as follows.

Promotion Limit Parameters

- Overall Promotion Limit - Across customers
    - Irrespective of duration -Total number of promotions applied across customers.
    - By specific duration - Total number of promotions applied across customers during a given period.
- Limit by Customer
    - Irrespective of duration -Total number of promotions applied for a given customer.
    - By specific duration - Total number of promotions applied for a given customer during a given period.
- Limit by Customer and Product
    - Irrespective of duration -Total number of promotions applied for a given customer and product.
    - By specific duration - Total number of promotions applied for a given customer and product during a given period.
- Order level Limits: This means that customer can only use certain number of promotions within a given order.
    - By promotion
    - By required purchase product level
    - Order level promotion limits can be dynamically applied and does not need to be tracked separately. However, account level and promotion level needs to be tracked.

For example, all the eligible doctors within a DSO (Days Sales Outstanding) will receive 1 case (Invisalign Teen, Full or Assist) priced at $990, if that case is submitted within 90 days of their re-engagement training. Else, they are not eligible for receiving cases worth $990.

Expected Behavior

Options Selected	Expected behavior
Auto Apply deselected, Combine with Other incentives selected	You can build a User Interface to display the "Show Available Promotions" list using the Promotions API, which displays non-auto applied promotions. The user can then select one or multiple promotions to be applied.

Options Selected	Expected behavior
Auto Apply deselected, combine with other incentives deselected	If a user provides multiple promo codes as a comma-separated list, the highest priority promotion is selected and applied from the list of promotions provided in the comma-separated list.
Auto-Apply selected, combine with other incentives deselected	If two or more promotions are applicable, the system picks the promotion with the highest priority and applies it.
Auto-apply selected, combine with other incentives selected	All promo codes fulfilling the criteria will be applied. If one or more promo codes have priority 1 which is applicable in a scenario all the promotions are applied. Suppose for a given scenario there are 4 promo codes which are applicable having priority 1,1,2,2 all four are applied. Pricing is not stacked, so if all the four qualify and allow others to be applied then all four are applied. If some of the promotions do not allow others to be applied then those promotions are filtered out.

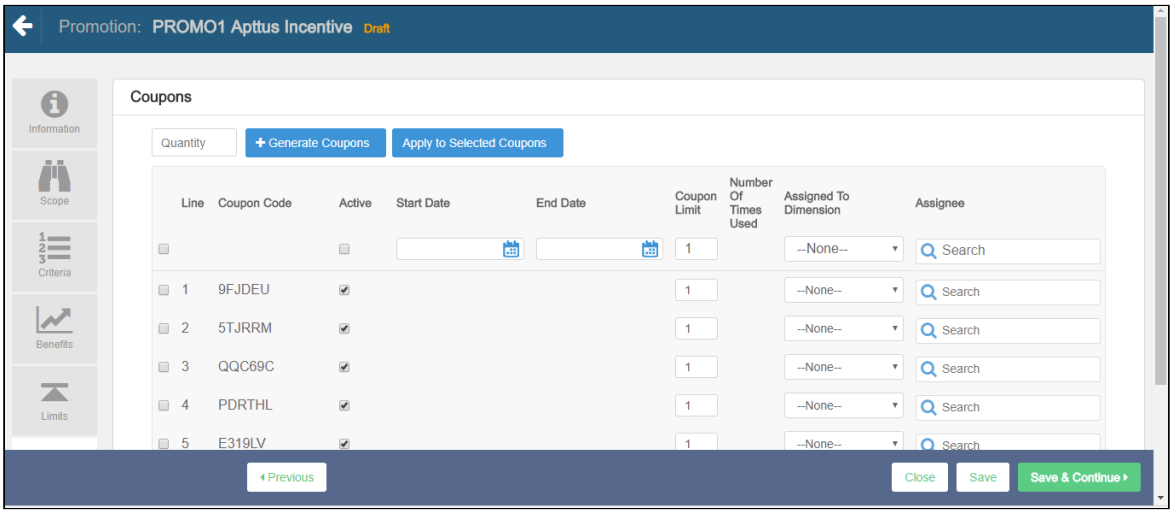
After defining promotions, ensure that you run **Criteria Maintenance > Update Pricing Fields** to apply the defined promotions.

Generating Coupons for a Promotion

Coupons are generally alphanumeric codes that correspond to a promotion. With Conga Promotions, you can now create multiple coupon codes for potential customers to promote your business and specific products in your inventory.

After you identify potential end customers in a target market, you can use coupons to track how customers use incentives, how you can optimize your promotions, and how you can re-target your customers. You can generate multiple coupon codes to share the range with your marketing team.

> ⓘ You can generate coupons only if you defined **Auto apply?** = **No** on the Information page. If **Auto apply?** = **Yes**, the **Generate Coupons** button is disabled.



To generate coupon codes for a promotion that you have created,

1. On the left work pane, click **Coupons**.
2. In the **Quantity** field, enter the number of coupons you want to generate and click **Generate Coupons**.
3. Enter values for the fields described in the following table:

> These fields act as filter criteria as well. For example, the coupon assign-to account or user needs to match the account field value that is specified on the quote or the logged-in user respectively. If the Account field value on the quote is missing, the coupon assign-to account needs to match the account linked to the opportunity.

Field	Description
Active	Select the checkbox to activate the coupon.
Start Date	The start date for the coupon.
End Date	The last date this coupon is valid.
Coupon Limit	The number of times you can use this coupon.
Assigned To Dimension	The level at which the coupon is valid. Select from one of the following options. • Incentive User • Account

Field	Description
Assignee	Select the user or account to which the coupon is applicable.

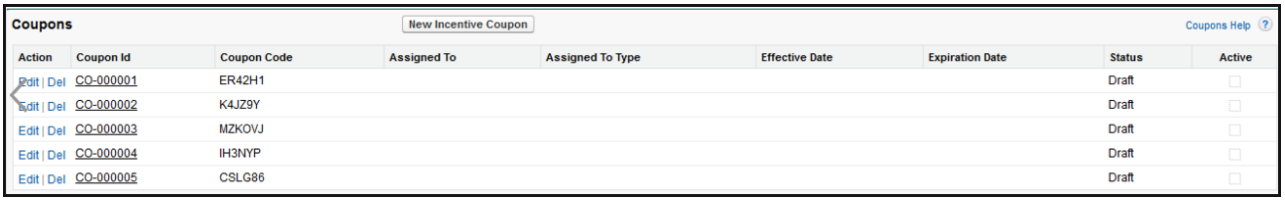
4. Click **Save** to save the changes or click **Save & Continue** to move to the next tab.

> After coupons are generated for a promotion, the Apply Promotions dialog does not list the promotion. The user must enter the coupon code to apply coupons instead.

5. Select specific coupons in the list and click **Apply to Selected Coupons**. You can also mass edit coupons by selecting the checkbox at the header and clicking **Apply to Selected Coupons**.
6. To create multiple coupons for one incentive, use the following sample code:

```
List<IncentiveCoupon__c> coupons =
IncentiveCouponFactory.createSObjectsForIncentive(incentiveSO.Id, 5);
            System.assertEquals(coupons.size(), 5);
            for (IncentiveCoupon__c couponSO : coupons) {
                  System.debug('couponCode=' + couponSO.CouponCode__c);
            }
```

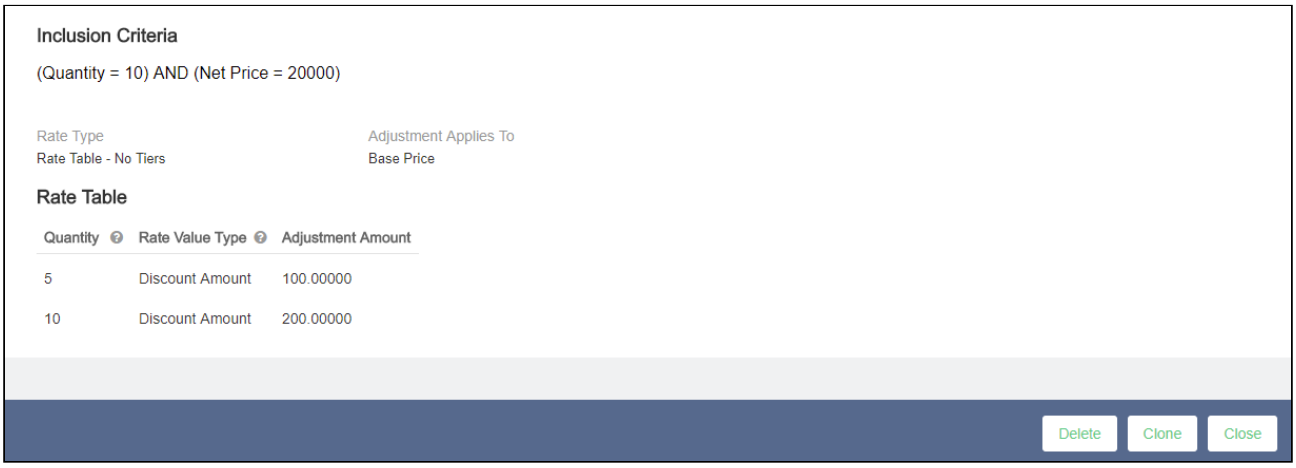
You can view the resulting list of coupons on the Accounts page.



> **The Coupon Code is displayed in the product details page. If there are multiple coupons, all coupons codes are saved as comma separated.**

Viewing the Summary of a Promotion

After you have configured the promotion record, you can use the Summary tab to view all the details. Something similar to the image below is displayed when you click Summary tab.



Defining Batch Size to Process Line Items with Promotions

During the pricing, you can process the promotions in batches for a cart with a large number of line items. You can add the **APTS_IncentivePricingBatchSize** Admin Setting to define the batch size of line items. The batch processing is only supported for **Buy X Get X** and **Buy X Get Y** promotions with **Auto Apply?** enabled. You can use this feature for both regular and async pricing.

To define batch size to process line items with promotions

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record. The New Admin page is displayed.
3. Fill in the following details

Field	Value
Name	*APTS_IncentivePricingBatchSize*
Value	Enter the number of line items to be processed in a batch. ⓘ You must define the batch size based on the complexity of the cart and the number of incentives applied on the line items.
Code	Leave the field blank

4. Click **Save**.

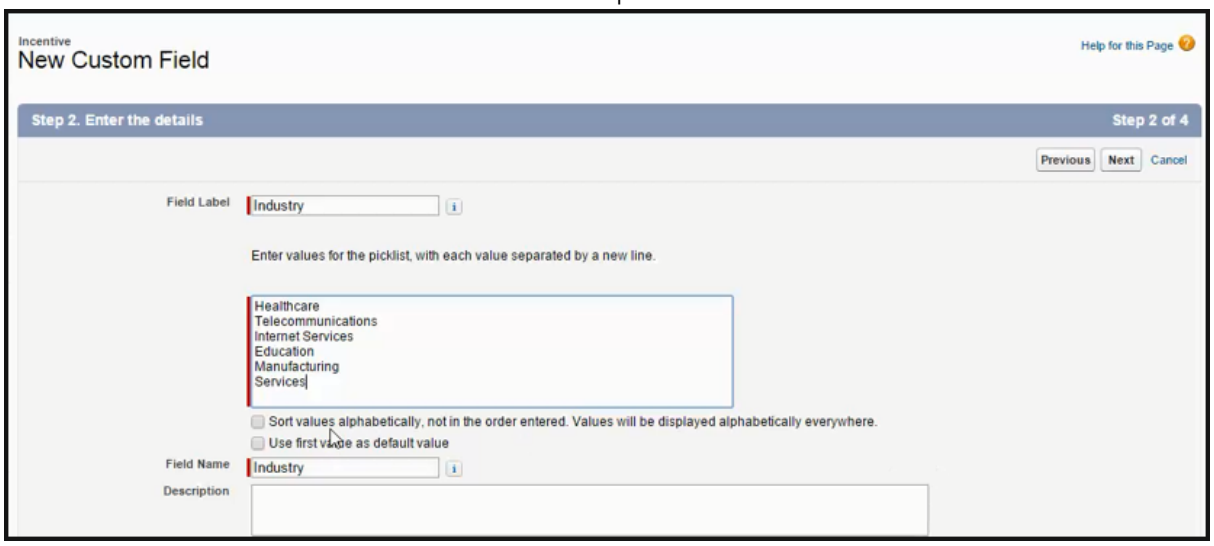
Adding Custom Fields to the Promotion Information Page

You can now add to the details that you want to capture on the Promotion Information page. You can include more fields such as check boxes, number, dates, texts, currency, pick lists, multi-pick lists, and other fields that Salesforce supports.

For example, you want to include a field to capture the industry for which you are rolling out this specific promotion. You can add such a custom field on the Promotion Information page.

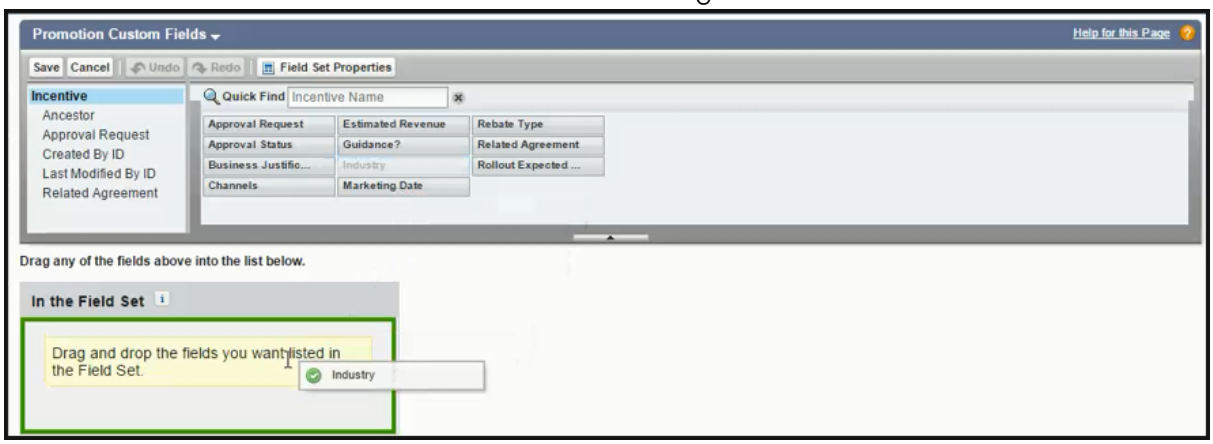
To add a custom field on the Promotion Information page

1. Add a custom field for the Incentive object. Let's call this field **Industry**, which is a pick list from which users can select a relevant option.



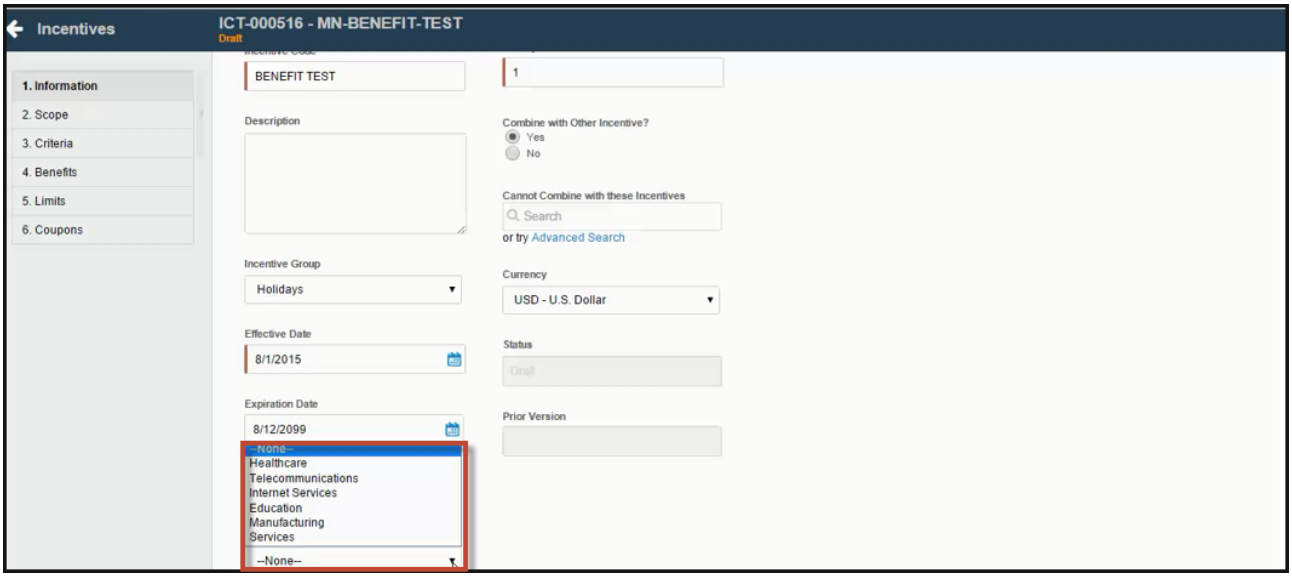
2. Add the pick list options you want to include. For example,
    - Healthcare
    - Industry
    - Manufacturing
    - Services
    - Telecommunication
    - Others

3. Now Edit the Incentive Field set to add the Industry field.



4. Click **Save**.

You have successfully added the Industry pick list to the Promotion Information page.



Configuring the Apply Promotion Window

The **Apply Promotion** window is displayed on the cart when the Sales Rep clicks **Apply Promotions** on the line items or the cart. You can configure the fields displayed on the window in the Config Settings.

To configure the Apply Promotion Window

1. Go to **All Tabs** > **Config Settings** > **Display Column Settings.**
2. From the **Display Type** drop-down, select **Apply Promotions**.
3. Select th flow you want to configure in **Flow**.

4. You can edit the existing fields or add new fields to the window. To add a new fields click add icon(✚).
5. The following fields are available to you:

Field	Description
Incentive Code	Displays the code of the incentive on the window
Incentive Name	Displays the name of the incentive on the window
Sequence	Displays the sequence of the incentive.
Expiration Date	Displays the date of the expiration of the incentive.
Description	Displays the description of the incentive that you added while creating the incentive.
Combine with Other Promotions	Indicates whether the promotion can be combined with other promotions.

   The fields **Incentive Code**, **Incentive Name**, **Sequence**, and **Expiration Date** are displayed by default.
6. Click **Save**.

Use Case: Configuring a Promotion for the Total Price on the Cart

This use case describes how to configure a promotion on an order level. You can configure a promotion such that the user gets a discount of 10% if the total price of the cart is greater than 500$.

This section provides an overview of how to configure promotion for the use case and list high-level steps of configuring the promotion for an order. For detailed steps about configuring a promotion, refer to Configuring Promotions.

Before you begin

Understand Promotions. Refer to Managing Promotions.

To Configure a Promotion for an order

1. Go to the Salesforce App Menu and select **Apttus Incentive Setup**.
2. Click **Incentives** > **New**.
3. On the Incentives information page, define values for the fields as described in Defining Promotions and **Save**. Ensure that you select **Summary Group** in the **Reference Data Source** Field.
4. In the **Criteria** tab, enter the line on which the promotion is applied. Select the following:
    - **Field**: Totals
    - **Operator:** Greater Than
    - **Value Type**: Constant
    - **Value:** 500

    > ⚠ You can also use Field as **Line Type** that has pre-populated values such as Total, Subtotal, Group Total, Category Total and Adjustment to configure the promotions for this use case.
5. In the **Benefits** tab, enter the following information:
    a. **Rate Type**: Single Rate
    b. **Rate Value Type**: % Discount
    c. **Adjustment Applies to**: Total price
    d. **Adjustment Amount**: 10

Result:

Click **Apply Promotions** on the Cart. The discount is visible on the Total Price.

Configuring Promotions on Benefit Products in Buy X Get Y

You can use the Admin Setting *APTS_EnableMultiBenefitItems* to control the ability of the Sales rep to apply promotions to multiple line items of the benefit product (Y). By default, this setting is disabled and CPQ applies the promotion only to the first line item of the

benefit product (Y). When you enable this setting, CPQ applies the promotion to all line items of the benefit product (Y).

To configure promotions on benefit products in a Buy X Get Y scenario

You must follow the steps below to configure this setting:

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record. The New Admin page is displayed.
3. Fill the following details

Field	Value
Name	*APTS_EnableMultiBenefitItems*
Value	*true* or *false*
Code	Leave the field blank.

4. Click **Save**.

Applying Promotions for eCommerce Orders

You can apply the promotions that you define, for orders placed on Conga eCommerce, using APIs.

The following table lists such APIs.

API	Description
getIncentivesForCart	To get a list of Incentives of type PULL (Auto Apply = False) for a given Cart for an order.

API	Description
updatePriceForCart	To apply one or more PULL promotions, set the comma separated list of PULL promotion codes (Incentive codes) on the Product Configuration object in the field "Apttus_Config2__CouponCodes__c". After setting the values, reprice the cart using the "updatePriceForCart" API. CPQ will auto-evaluate all promotions of type PUSH (auto Apply = True) and will apply qualified promotions as part of the pricing.
applyIncentiveLimit	API to apply commit incentive limit from work-in-progress orders to the final limit data. Ideally, this API will be called when the order is accepted or confirmed by the customer. However, individual implementations can decide when to call this API.

Analyzing Promotions

For analysis purpose, CPQ and Promotions Management captures the information of the applied promotions at the line item level in the promotions code and promotion benefit. As a Marketing Manager, once the promotions are applied on the sold products, you can perform different types of analysis by creating reports and dashboards for the applied promotions and forecast your business revenue.

The different promotions applied on your cart and the ones which are pending to be applied on the cart are captured in the form of **Incentive Limit Data**. This also helps in knowing limit details, such as Limit Start Date, Limit End Date, Limit Value and more.

Incentive Groups club all the incentives together under a single row, based on your business requirement, and help you analyse how the incentives applied from a certain Incentive Group are helpful in growing your business.

In case of multiple benefits applied on a single product or cart, the system captures the details of each benefit applied through a promotion in the Adjustment Line Items or Order Adjustment Line Items.

Configuring Assets

Quote-to-Cash is the end to end business process between your customer's interest in a purchase and the revenue that your business realizes from that purchase.

The Quote-to-Cash flow includes creating a Sales Quote, submitting the quote, negotiating and managing the contract, fulfilling the order, tracking payment, and managing your customers' purchases.

In the Quote-to-Cash flow, Asset Management is a set of mission-critical business processes that you can employ to manage your customer's purchased products with a variety of billing models to ensure efficient collections and accounting.

The following table lists and describes industry challenges with Asset Management and the solutions that Conga provides for each challenge.

Challenges with Asset Management	Conga-provided solutions
Asset Management can be a complex coordination of activities including custom design and configuration. Depending on your business requirements it could be, • scheduling milestone-based manufacturing processes • software provisioning • equipment installation • scheduling services teams. If customers update their orders, fulfillment becomes even harder to manage.	Asset Management is based on the order information that you have already defined in your sales quote and resulting contracts. Asset Management then, becomes very easy when you integrate Conga CPQ with Conga CLM. Historically, Assets have been managed using an Enterprise Resource Planning (ERP) application. Because it is important for you to capture on-the-fly changes to orders and integrate your Order Management processes with customer facing applications like Contract Management solutions, assets or purchased products are now also managed within a CRM system, or in multi-functional CPQ applications.
If a customer has multiple orders or changes, invoices must be coordinated so your customers do not receive multiple invoices.	Renewals, changes, swaps, terminations, and additions to an asset increase the complexity of invoices. You can use Asset Management to generate an accurate, up to date, and easy to understand invoice so your customers understand it easily, pay quickly, and have a positive experience.

Challenges with Asset Management	Conga-provided solutions
Your finance team must analyze transaction and revenue information to device the best revenue schedules that determine when revenue is recognized. Revenue recognition depends on when your customers receive their products, services and subscriptions.	Revenue recognition can be a challenge, but it's much easier when Finance can automatically generate the appropriate revenue schedules for products, services, and subscriptions based on contracts. With CPQ, you can verify if services are rendered or products delivered, and then see how those impact your customers' billing schedules and the overall revenue forecast.

You can define the Asset Management functions with different data objects to track Quote and Contract details until an order is fulfilled. With Asset Management you can

- make it easy for customers to add more products and services
- align all product changes to your customer's bill cycle
- drive charge pro-rations and pro-ration credits
- customize the renewal process that best suits your organizational needs

The following table lists the benefits Conga Asset Management has over ERPs.

Challenges with ERPs	Solutions by Asset Management
Do not understand recurring revenues or distinguish between monthly recurring revenues and annual recurring revenues.	Works with accounting solutions and meet the financial requirements unique to both one-time purchases and subscription businesses.
Do not handle pricing and packaging changes like upgrades, downgrades or increments.	Can quickly modify your pricing and packaging options irrespective of the number you increment or decrement.
Do not tell you how many active customers you have, or your rate of customer retention.	The reporting feature can detail your renewal rates and help you up-sell, cross-sell, and formulate strategies to increase customer retention.
Do not give you a unified view of how your business is doing.	Works seamlessly with CPQ, CLM, and Billing Management to bring awareness of information including renewal plans and usage statistics to every department in your organization.

Whether it is a one-time order, a subscription, or a usage-based service you sell to your customers, you can use Asset Management to manage any purchased product or service for your customers.

Asset-Based Ordering functionality is now built into CPQ. With the Asset-Based Ordering functionality, you can service and manage existing orders based on your customers' requirements.

With ABO, you can perform transactions for Purchased Products using cloud-enabled systems that operate on a massive scale and real-time speed. With Asset-Based Ordering, you can perform complex functions such as suspensions, mid-cycle upgrades, downgrades, add-ons, and cancellations, all on a single, user-friendly interface.

With Asset-Based Ordering you can

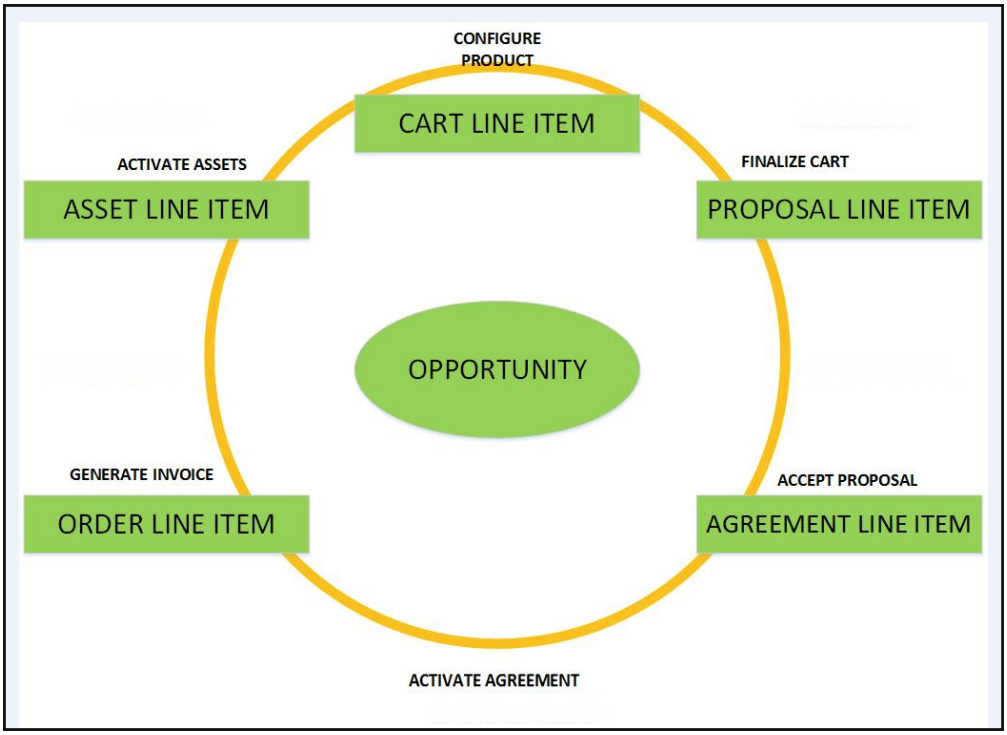
- work seamlessly with CRMs to bring subscription awareness to your front office
- unify your commerce, billing, and finance processes so customer relationships take center stage
- increase recurring revenue that you earn from existing customers by retaining them with renewals, incentives, rebates, and promotions

Asset-based Ordering adds to the robust CPQ functionality and the larger Quote-to-Cash experience by giving you a quick return on your investment and quickly increasing both, revenue and customer base.

About Asset and Asset Line Item

Any product or service that your customers buy, become that customer's assets. While your organization may have a unique and customized Order Fulfillment process, all the processes and procedures in Apttus Asset Management start with the Line Item in an order.

The following image describes the life-cycle of a Line Item in the Quote-to-Cash flow.



After your customers agree to purchase a product, the product changes from a Proposal Line Item to an Order Line Item. The status of the Quote/Proposal in this stage is **Accepted** and the order is now ready to be invoiced.

After you receive payments for an order is when the order line item changes to an **Asset** line item.

As an Administrator, you must configure CPQ to enable Customer Service and Sales Representatives in your organization to be able to perform the following actions to manage assets.

1. Generate asset records from orders
2. Renew the term for an asset
3. Change the **Term**, **Quantity**, **Options**, and **Attributes** of a product
4. Cancel products and subscriptions
5. Swap one purchased product with another
6. Create quotes for new products and services based on assets
7. Create quotes to modify existing products and services
8. Get visibility into the asset life cycle during customer interactions

Before you start to define the mechanism that Customer Service and Sales Representatives must use to manage a customer's assets, you must understand how they use Asset-Based Ordering.

Asset-Based Ordering Packages Required

For implementing Asset-based Ordering, follow the instructions of installing a package from the Install Center explained in the Installing CPQ section. Ensure that you have installed the following packages in your environment. Always check the latest Release Notes to get the latest package versions.

For the Quote/Proposal Flow,

Order	Package	Install Center tab to access the package
1	Conga Quote Management	CPQ
2	Conga Configuration & Pricing	CPQ
3	Conga CPQ Api	CPQ
4	Conga Quote Configuration Integration	Integrations
5	Conga Quote Asset Integration	Integrations

For the Contracts Flow,

Order	Package	Install Center tab to access the package
1	Conga Contract Lifecycle Management	Contract Management
2	Conga Quote CLM Integration	Integrations
3	Conga CLM Configuration Integration	Integrations

Configuring Asset-Based Ordering

As an administrator, you must enable Customer Service and Sales Representatives in your organization to **Change**, **Swap**, **Renew**, and **Terminate** a customer's assets. To this end, you must complete the following procedures in the listed order.

Mandatory Configuration

The following steps are the mandatory configuration steps which are required to implement ABO successfully.

1. Granting appropriate access and permissions to Managers, partners, other administrators, Customer Service, and Sales Representatives
2. Configuring Flow Management Settings
3. Configuring Installed Products Settings
4. Configuring Asset Management Flows
5. Customizing Display Actions for the Installed Products page
6. Defining the look and feel of the Installed Products page
7. Configuring Asset-based Pricing
8. Configuring Lookup Field Settings

Optional Configuration

The following configuration steps are optional. You may implement these steps based on your business requirements.

1. Defining the Actions you want to allow for each Asset Line Item
2. About Integration of Assets with Contracts
3. Creating an Approval Workflow for **Renew**, **Change**, **Swap**, and **Terminate**
4. Configuring Mass Update for Assets
5. Creating a Replacement Rule to enable the Swap feature
6. Configuring the Custom Attributes Page

The values you enter for the fields in the pages listed above will determine how Customer Service and Sales Reps in your organization manage a customer's assets.

About Asset Management Flows

Before you start configuring Asset-Based Ordering, you must understand how the Sales and Customer Service Representatives in your organization will use the ABO functionality.

Asset-based ordering is particularly useful when your business offers complex service products, such as phone services and equipment.

On the Installed Products page, you can Renew, Terminate, Swap, and Change an existing standalone, fixed, or a configurable bundle asset. For each transaction involving ABO actions, the system creates a new order while you are navigating on the Cart. The status of the Asset Line Item is not changed until you finalize the Cart and activate the Order containing that Asset Line Item.

Use case

For a sales-driven industry with products such as equipment (new purchase) and service (purchased for an equipment), you might want to achieve the following:

- As a manufacturing sales representative, for a service flow, you want to view the **Renew** and the **Terminate** button. For an equipment flow, you want to view the **Change** and the **Swap** button.
- As a subscription sales representative, you want to see the **Renew** button only when I make a purchase for a quote of type *renew*. For an add-on quote or an upgrade quote, you do not want to see the **Renew** button. For example, managing the renewal of a magazine subscription.

There are total 3 flows that you can consider before starting the configurations for using ABO. Ensure that you understand and choose one of the flows to implement error-free ABO.

1. Quote/Proposal Flow**:** This is one of the signature events in Asset-based Ordering involving Quote/Proposal. This is the normal flow of going through the Quote/Proposal to an Order and then to the Assets. Once you accept a Quote/Proposal, the Order is generated. Upon activation of the order, assets are created.
2. CSR Flow**:** This is the flow designed to enable the Sales Rep to skip the creation of a Quote/Proposal. No signature event, such as the creation of an Agreement or a Quote/Proposal is involved. A new button called **Asset Manager** is configured on the Accounts page to help the Sales Rep navigate directly to the Installed Products page.
3. Contract Flow**:** This is one of the signature events in Asset-based Ordering involving Contracts/Agreements. The assets are created through Contracts/Agreements using the normal Contract lifecycle flow. Once you finalize a Contract (containing Agreement Line Items), the corresponding Asset Line Items are created.

Quote/Proposal Flow

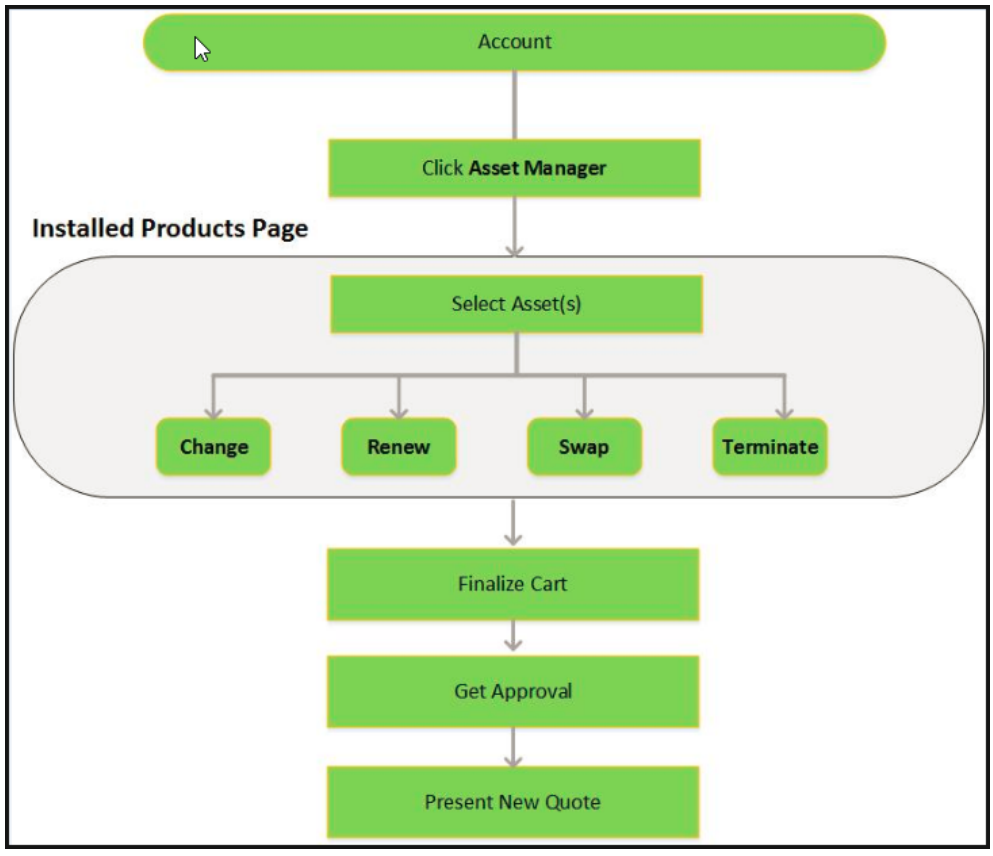
In this flow, the Sales Rep configures the Quote/Proposal and finalizes a configuration. Once the configuration is finalized, the Sales Rep presents the configuration to the customer. Upon acceptance from the customer, the Sales Rep clicks the **Accept** button on

the Quote/Proposal. An order is created for this Quote/Proposal, which upon activation, creates the Asset Line Items with the *New* status. The Sales Rep can configure the Quote/Proposal again and navigate to the Installed Products page to view all the active assets. No custom triggers are required here.

CSR Flow

There can be scenarios when your Sales Rep does not want to configure the Quote/Proposal again to land on the Installed Products page. For these type of scenarios, you can configure the **Asset Manager** button on your Account page. Clicking **Asset Manager** creates an Order and navigates you directly to the Installed Products page. In this flow, you skip the navigation of going to the Installed Products page through the **Quote/Proposal > Configure Products**. You just have to configure a custom formula button to achieve this, explained here.

The following image illustrates the CSR flow from the Sales and Customer Service perspective.



Contract Flow

A contract contains Agreement Line Items, Asset Line Items or Order Line Items. In this flow, you follow the normal Contract Lifecycle flow of creating a Contract, presenting it to your customer and finalizing the Contract. Here, the Asset Line Items are created when the Sales Rep finalizes and activates a Contract. CPQ now provides the ability to synchronize the changes in Assets with the changes in a Contract. The changes you make inside a Contract (such as a change in Contract End Date) will be reflected in the Assets contained in that Contract. Similarly, when you perform actions on your purchased Assets (such as changing the Quantity of an Asset), these changes will be reflected in the Contract.

See Configuring the Custom Settings for Asset Management Flow to learn how to configure the above flows.

Configuring Asset Management Flows

For the different Asset Management Flows explained here, you must configure different custom settings.

To enable the Quote/Proposal Flow

1. Go to **Setup > Build > Customize > Develop > Custom Settings > Proposal System Properties > Manage**.
2. Click **Edit** next to **System Properties**.
3. Search for **Auto Create Order** checkbox and select it. For more information, Configuring Proposal Settings.
4. Click **Save**.

To enable the CSR (Order) Flow

Asset Manager is a simple formula field that enables the creation of an order header which in turn is used as a container to manage a customer's assets without a proposal.

1. Click **Setup > Create > Objects**.
2. Select the **Account** object and go to the **Custom Fields & Relationships** related list.
3. Create a new formula field with the label **Asset Manager** and *Text* as **Return Type**.
4. In the advanced formula editor, enter the query string with the following parameters.

```
HYPERLINK("/apex/Apttus_Config2__AccountOrderCreate?id=" & Id &
"&method=csrFlow&flow=csrFlow&priceListId=<pricelistid>&launchState=assets&auto
```

```
Finalize=true&activateOrder=true&retId=" & Id , IMAGE("", "Asset Manager"),"_se
lf")
```

In the above formula, replace **<pricelistid>** with the *ID* of your desired Price List. For the flow parameter, specify the name of the Flow that you have defined in the Flow Settings. To support this button across different Price Lists, you must create separate buttons for each Price List.

5. Click **Save**.

To enable the Contract Flow

1. Go to **Setup > Build > Customize > Develop > Custom Settings > Comply System Properties > Manage**.
2. Click **Edit** next to **System Properties**.
3. Search for **Auto Create Order** checkbox and select it. For more information, see **Comply System Properties** section in Configuring Custom Settings.
4. Click **Save.**

> ⚠ We recommend you to refrain from enabling **Auto Activate Order** from **Comply System Properties** as well as **Proposal System Properties**, simultaneously.

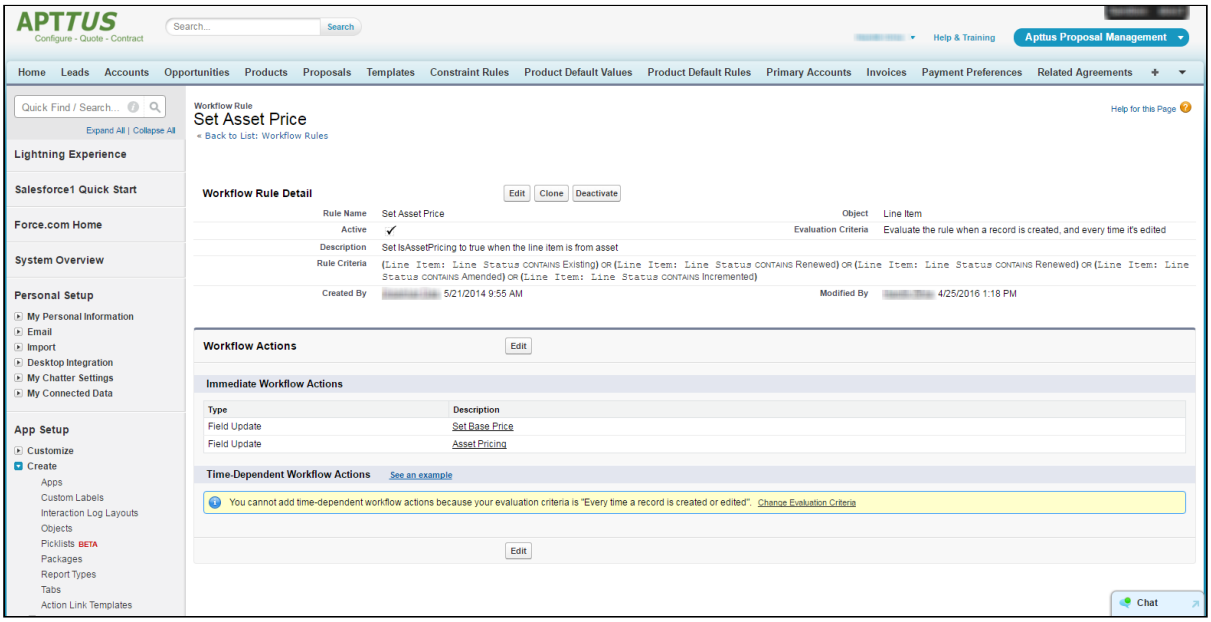
For detailed information on the behavior of these two checkboxes, see About Integration of Assets with Contracts.

Configuring Asset-based Pricing

The price loaded in an asset is based on the price mentioned in its Price List Item. As an administrator, you can decide if you want to retain the asset price or set the new base price in the further transactions on the Installed Products page. A workflow is used to set the the above actions for the asset pricing. If the **IsAssetPricing** field of the Line Item object is set to *True*, the system retains the price of the Asset Line Items for any future renewals or amendments.

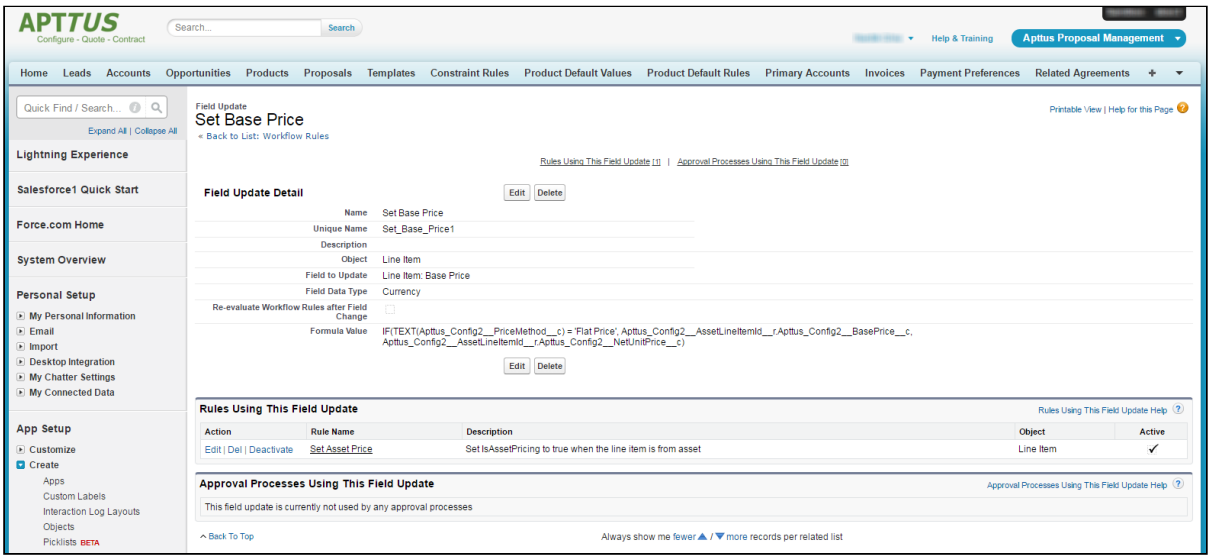
For example, the pricing for a cell phone plan has increased from $20 to $30 per month, and the Price List Item record is updated to reflect the new list price. However, the selling company has decided that customers who already have the product installed as an asset will not be charged the new price for renewals, amendments, etc. These customers shall be allowed to purchase the same subscription at the base price at which they purchased it originally.

You must create a Workflow Rule for the Line Item object, as described in the Salesforce Documentation.



For the Workflow actions,

1. Set the IsAssetPricing check box to *True*.
2. Set the formula for Base Price (depending on your requirement).



Ensure that all the Options products of your Bundle follow the Asset Pricing if you have configured a Workflow Rule.

Configuring Asset-Based Pricing without Workflow Rules

Using a workflow rule to obtain the price of an asset during Asset-based operations has certain limitations. You could use workflow rules to fetch asset pricing on quantity and

term updates but the process gets complicated when it involves attribute or other field changes.
Asset pricing without workflow rules helps you pick the selling price of an asset or its new price from the Price List item as and when needed.

With this feature, you can choose to

- Retain the asset price on the cart (default asset price)
- Go back to the Price List Item and fetch the new price of the asset. This happens when the parameters driving pricing undergo change.

Configuration

To turn ON Asset-based pricing without creating a workflow rule, you must configure the following two parameters:

- **Default Asset Pricing Indicator**: When checked, the system loads the asset price by default in the cart and monitors the changes done to fields added in the Asset Pricing Criteria.
  Go to **Custom Settings** > **Config System Properties** to turn ON the pricing indicator.
- **Criteria Field Names**: Enter the API name of the field like Quantity or attribute that drives pricing.
  Go to **Custom Settings** > **Config Asset Pricing Criteria Fields** to provide the field APIs. Any changes to the fields specified here will trigger the flip and you will see the price from PLI on the cart. If you revert the update, the price will change to the default asset price.

- If the default asset pricing indicator is off, then no asset pricing criteria should be defined.
- Asset pricing is always turned on if there asset pricing criteria defined.
- Asset pricing indicator is at a global level. Any requirements to turn it off on a selective basis must be handled through a workflow rule to deselect the indicator on the cart load.

Use Case 1

1. Create a product 'B' with tiered pricing. The per unit price of the product is $1200.
2. Ensure **Default Asset Pricing Indicator** is set to True.
3. Create a Quote with one unit of product 'B' for 1200. Offer a discount of $200 to see that the price of asset 'B' now becomes $1000.
4. Navigate to **Setup** > **Develop** > **Custom Settings** > **Config Asset Pricing Criteria Fields**. The Config Asset Pricing Criteria Fields page is displayed.

5. Click **New**. The Config Asset Pricing Criteria Fields Edit page is displayed.
6. In the Name field, enter a name for the asset pricing criteria.
7. In the **Criteria Field Names**, enter Apttus_Config2__Quantity__c (API name of the **Quantity** field).
8. Keep the **Line Status** field blank.
9. From the CSR flow, select this asset 'B' and perform the ABO operation of **Change**. You will see the asset price of $1000 on the cart.
10. Change the **Quantity** to *'5'* and you will see that asset price is now set to $1200 (from the PLI).

Use Case 2

1. Create a product 'B' with tiered pricing. The per unit price of the product is $1200.
2. Ensure **Default Asset Pricing Indicator** is set to True.
3. Create a Quote with one unit of product 'B' for 1200. Offer a discount of $200 to see that the price of asset 'B' now becomes $1000.
4. Navigate to **Setup** > **Develop** > **Custom Settings** > **Config Asset Pricing Criteria Fields**. The Config Asset Pricing Criteria Fields page is displayed.
5. Click **New**. The Config Asset Pricing Criteria Fields Edit page is displayed.
6. In the Name field, enter a name for the asset pricing criteria.
7. In the **Criteria Field Names**, enter Apttus_Config2__Quantity__c (API name of the **Quantity** field).
8. In the **Line Status** field, enter Renewed.
9. From the CSR flow, select this asset 'B' and perform the ABO operation of **Change**. You will see the asset price of $1000 on the cart.
10. Ensure that the line status changes to Renewed.
10. Change the **Quantity** to *'5'* and you will see that the asset price is still $1000.
The asset pricing criteria works for change in quantity, but the asset pricing criteria does not work for Renewed. Though you change quantity, you will still see the asset pricing.

Standalone Asset - Quantity Update

Similar to the above use-case, the parameter **Quantity** drives pricing in the following case.

Standalone Asset	Asset pricing Enabled	Base Price	Quantity	Term	Extended Price	Discount	Net Price	Asset Net Unit Price (Net price / Quantity*term)
Initial Sale	N/A	100	2	12	2400	350	2050	85.416
On load of the Cart	True	85.416 (fetched from Asset Line Item)	2	12	2050		2050	
On Change (asset)	True	85.416 (fetched from Asset Line Item)	4	12	4100	100	4000	83.333

Standalone Asset - Quantity Update

The Attribute **Service Level type** Gold has the price $100 and type Platinum $200. The Asset price will change based on the changes made to Service Level attribute.

Standalone Asset	Attribute (Service Level)	Base Price	Quantity	Term	Extended Price	Discount	Net Price	Asset Net Unit Price (Net price / Quantity*term)
Initial Sale	Gold	100	2	12	2400	350	2050	85.416

Standalone Asset	Attribute (Service Level)	Base Price	Quantity	Term	Extended Price	Discount	Net Price	Asset Net Unit Price (Net price / Quantity*term)
On load of the Cart	Gold	85.416 (fetched from Asset Line Item)	2	12	2050		2050	
On Change (asset)	Platinum	200 (fetched from Price List Item)	2	12	4800	100	4700	195.8333

Overriding the Default Asset Pricing for Line Statuses

CPQ enables you to override the default asset pricing for some of the defined line statuses.

Prerequisites

Ensure that the **Default Asset Pricing Indicator** is turned off at **Config Settings** > **System Properties**.

To override default asset pricing for a line status

1. Navigate to **Setup** > **Develop** > **Custom Settings**.
2. Click **Manage** next to **Config Asset Pricing Default**. The Config Asset Pricing Default page is displayed.
3. Click **New**. The Config Asset Pricing Default Edit page is displayed.
4. In the **Name** field, enter the name of status for which you want to override the default asset pricing.
5. Select the **Default Asset Pricing Indicator** checkbox.

When you come to cart, if the line is in the status you configured above, you will see that the value is from PLI, not from the asset pricing because you have overridden the default setting.

Configuring the Installed Products Page

The Sales and Customer Service Reps use one of the flows to navigate to the Installed Products page. You must complete the following procedures to define the content and aesthetics of the Installed Products.

1. Configuring Column Settings for the Installed Products Page.
2. Defining the Number of Products Displayed on the Installed Products Page.
3. Configuring Asset Visibility on the Installed Products Page.

The values you define for the pages listed above, determines how the Sales and Customer Service Reps access and use the Installed Products page.

Accessing the Installed Products Page

To access the Installed Products Page, you must create a new quote. When you configure products, the system creates a new cart with an Account ID attached to it.

Add a few products to the new quote and finalize the cart and accept the quote. Make sure that the quote status has changed to *Accepted*. After you click the **Installed Products** button, the Installed Products page displays all the asset line item records from the account information linked to the current cart.

To configure direct navigation to the Installed Products page from the Quote/Proposal page

1. Click **Setup** > **Create** > **Objects**.
2. Select the Quote/Proposal object and from the Custom Fields & Relationships related list, select **Configure Products** with the Installed Package of Apttus Quote/Proposal-Asset Integration.
3. Click **Edit** and replace the query string with the following parameters.
   For the classic CPQ,

```
IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/
Apttus_QPAsset__ProposalConfiguration?id=" &Id, IMAGE("/resource/
Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"), NULL)
```

   For the new UI,

```
IF ( LEN( Apttus_QPConfig__PriceListId__c) > 0 , HYPERLINK("/apex/
Apttus_QPConfig__ProposalConfiguration?id="&Id+"&flow=ngflow&launchState=assets
",
IMAGE("/resource/Button_Configure", "Configure Products"),"_self"), NULL)
```

4. Click **Save**.

On the Quote/Proposal page when you click the **Configure Products** button, the Installed Products page appears instead of the Catalog page.

To access the Installed Products page directly from the Account page

1. Go to **Setup** > **Create** > **Objects**.
2. Select the **Account** object and from the **Custom Fields & Relationships** related list, create a new custom formula field.
3. In the Advance Formula editor, enter the query string with the following parameters.

   ```
   HYPERLINK("/apex/Apttus_Config2__AccountOrderCreate?id=" & Id &
   "&method=csrFlow&priceListId=<pricelistid>&flow=csrFlow&launchState=assets&
   activateOrder=true&retId=" & Id , IMAGE("", "Asset Manager"),"_self")
   ```

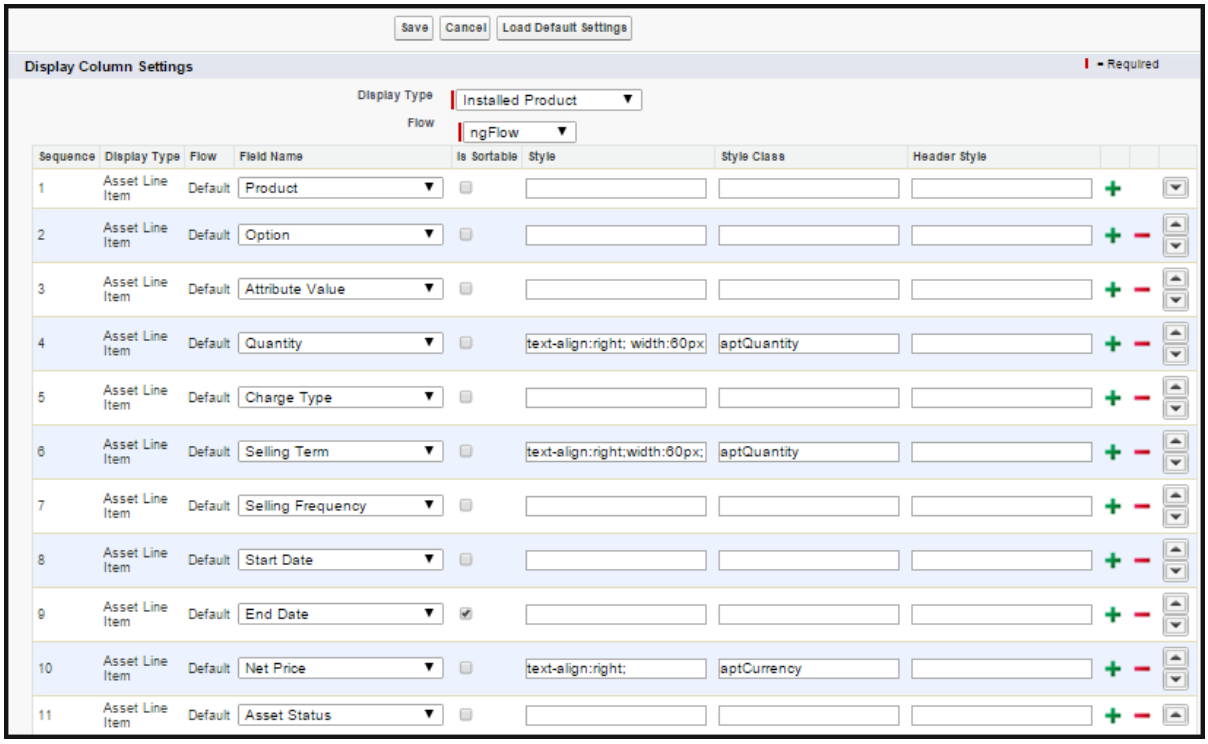
   In the above formula, replace **<pricelistid>** with the *ID* of your desired Price List. For the flow parameter, specify the name of the Flow that you have defined in the Flow Settings.
4. Click **Save**.
   This button creates an order specific to a Price List and Account ID. You may have to create different asset manager buttons for different accounts.

Configuring Column Settings for the Installed Products Page

You can customize the Installed Products page to suit both, your business requirements and the ease-of-use for the Sales and Customer Service Representatives in your organization.

To configure these settings,

1. Click **All Tabs** (+) > **Config Settings** > **Display Column Settings**.



2. Add the **Product** field first, followed by the **Option**, **Attribute**, and **Charge Type**.

> **Note**
>
> You must also ensure that the Product field in the Cart page settings in NOT EDITABLE and always the first column.

3. Click **+** to add any subsequent fields that you want on the Installed Products page.
4. Deselect the **Is Sortable?** check box for all the fields you have added.
5. Click **Save**.

Defining the Number of Products Displayed on the Installed Products Page

For the New UI, you can use the following custom setting to customize the number of products on the Installed Products page. This number can be for a specific user, based on the user's preference. The products are divided across the pages using the Pagination concept.

To enable the above enhancement:

1. Go to **Setup > Develop > Custom Settings** and search for **Config User Preferences.**
2. Click **Manage** next to **Config User Preferences**.

3. For the required user, do any of the following:
   Click **View** to view the details and click **Edit**.
   - Or -
   Click **Edit** to edit and make changes to the existing user preferences.
4. In the **Catalog Products Per Page** field, enter the number of products that you want to display on the Catalog page per page. Valid values are *10*, *20*, *50*, and *100*. This setting is available specially for the New UI.
   In the **Selected Products Per Page** field, enter the number of products that you want to display in the mini cart on the Catalog page. Valid values here are *5*, *10*, *15*, *20*, and *25*. This setting is available for both, the enterprise CPQ and the New UI in CPQ.
5. Click **Save**.

Now, when you browse the catalog page as a specific user, you will find the specified number of products on a single page and the rest of the products are divided in multiple pages using Pagination.

Configuring Asset Visibility on the Installed Products Page

You can navigate to the Installed Products from the Catalog page, Cart page, or through the Account page with the help of different formula buttons. You can control the assets (bundles, options and attributes) displayed on the Installed Products page. You can also see critical attribute values at both, the bundle and sub-bundle level when expanding a Bundle product on this page.

> ⓘ Attributes for which you select the **Is Primary** check box on the Attribute Configuration page are called Critical Attributes.

Filtering the assets on the Installed Products page

- Search Filters: You must create Search Filters for the Installed Products page to enable the Sales or Customer Service Reps in your organization to search for specific purchased products that customers ask to be changed, swapped, or terminated.
- **Callback class:** You must use the code in the sample AssetLineItemCallback class to further filter the assets on the Installed Products page. For more information, see Asset Line Item Callback Class.

To customize the Filter Fields on the Installed Products page

1. Click **All Tabs** (➕) > **Config Settings** > **Installed Products Settings**.

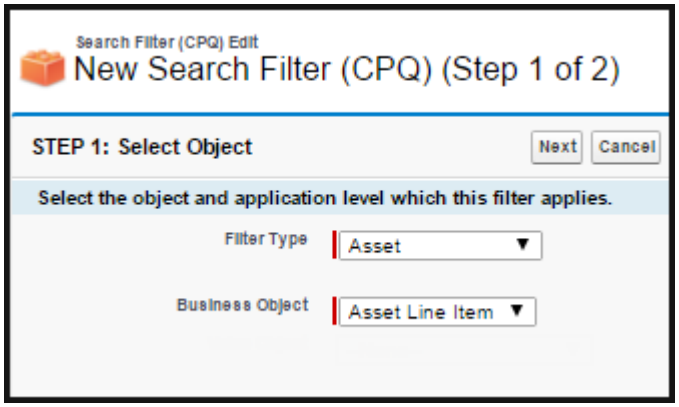
2. In the Filter Fields, specify the API names of the Asset Line Item fields that you want to use as filter fields.
   Currently, the fields of type *Text*, *Picklist, Multi-picklist, Date, Datetime* and *Boolean* are supported.
3. Click **Save**.

These fields will show up as filter fields on the Installed Products page. These fields work as temporary filters and you can change the values in these filters dynamically on the Installed Products page.

Creating Search Filters for the Installed Products Page

To create a Search Filter on the Installed Products page

1. Click **All Tabs** (➕) > Search Filter (CPQ) > New.



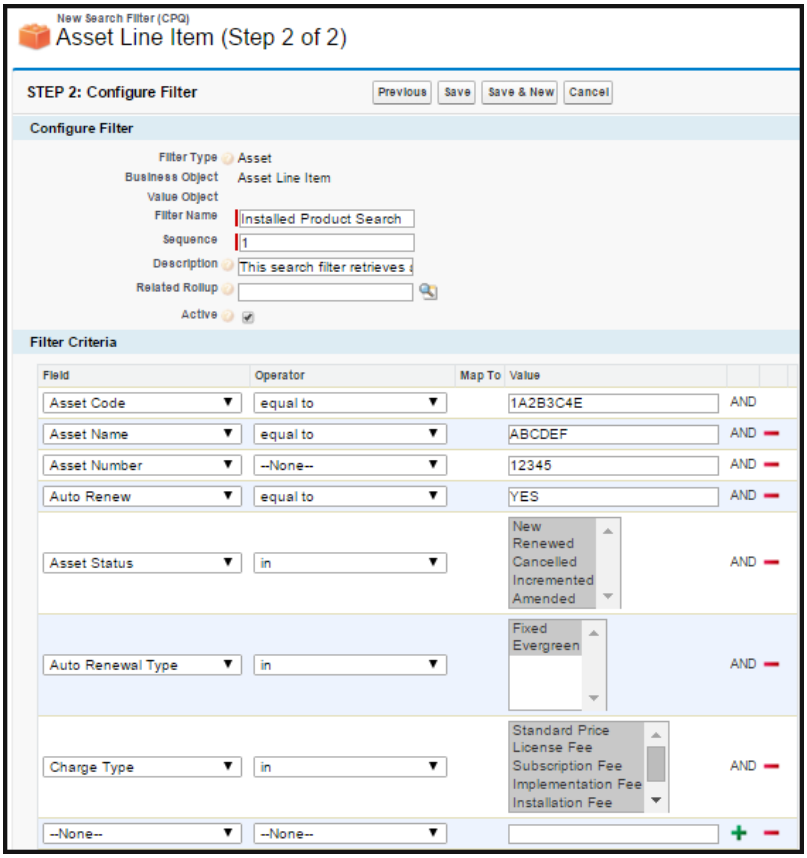
2. Enter values for the following fields.
   Filter Type - Asset
   Business Object: Asset Line Item
3. Click **Next**.
4. In the Configure Filter section, enter values for the fields described in the following table.

Field	Description
Filter Name	Enter a unique name for this search filter.
Sequence	Enter the sequence in which the Search filter is displayed on the Installed Products page.

Field	Description
Description	Describe the function and purpose of this search filter to avoid duplication.
Related Roll up	Lookup and select a pre-defined roll up on which you want to base your search results.
Active	Select the checkbox so the Sales and Customer service representatives can use the filter.

5. In the Filter Criteria section, click + to add the fields you want to include to your search criteria. Use the available operators to establish the relationship with the value you define for each field.
   For example, you want the Sales and Customer service representatives in your organization to be able to search for assets based on the following parameters.
   - Asset Code
   - Asset Name
   - Asset Number
   - Asset by Status
   - Assets with **Auto Renew** enabled
   - Assets of any Auto Renew Type

- You can add fields to the Search Filter and define values for these fields that are based on your business requirement.



6. Click **Save**.

> ⓘ When you specify the value in a filter field, only those records that match the criteria of the Search Filter are displayed. For example, for **Start Date** and **End Date** filter fields, if you specify the **Start Date** as *07/01/16* and **End Date** as *07/31/16*, the system displays the assets having **Start Date** as 07/01/16 or **End Date** as 07/31/16 and not the assets falling in the range of these two dates.

Displaying Assets from Multiple Accounts on the Installed Products Page

By default, the assets that CPQ displays on the Installed Products page are from a single account related to the current quote. As an administrator, you can configure CPQ to display assets from multiple accounts, which are related to an opportunity, on the Installed Products page. You must complete the following configurations to display assets from multiple accounts:

1. Enable the admin setting *APTS_EnableInstallBaseFilteringAcrossAccounts*.
2. Configure a custom callback class.

To enable asset filtering across accounts

1. Go to **All Tabs** > **Admin**.
2. Click **New** to create a new record. The New Admin page is displayed.
3. Fill in the following details:

Field	Value
Name	*APTS_EnableInstallBaseFilteringAcrossAccounts*
Value	Enter one of the following values: • true: CPQ displays assets from multiple accounts. • false: This is the default value. CPQ displays assets from a single account related to the quote. It is recommended to keep the value as False. Set it to True only if there is a business requirement to fetch assets from multiple accounts.
Code	Leave the field blank.

4. Click **Save**.

To configure a custom callback class

1. Configure a custom callback class to fetch assets from the list of accounts. The sample code below enables you to fetch assets from multiple accounts.

```
/*
 * This class is used in Asset Line Item callback to show asset on Cart page.
 */
global with sharing class CurrentUserFilter_AssetLineItemCallback implements
 Apttus_Config2.CustomClass.IAssetLineItemCallback4
    {

    private List<String> assetSearchScope = null;
    private String assetSearchFilter = null;
    private ID userId = null;

    /**
```

```
     * Callback at the beginning of the asset selection call.
     * Use the start method to initialize state
     * @param cart the cart object or null if there is no cart context
     * @param assetSearchFilter the preset static filter used in the asset
search or null if there is no preset filter
     * @param assetSearchScope the list of asset fields to match the search
text or empty to use the default list of fields
     */
    global void start(Apttus_Config2.ProductConfiguration cart, String
assetSearchFilter, List<String> assetSearchScope) {
        this.userId = cart.getConfigSO().CreatedById;
    }

    /**
     * Callback to return part of SOQL filter clause
     * This filter is used in listing installed products
     * @param accountId is the context account id
     * @return The query filter is like the following.
     *         Name LIKE 'A%' AND Quantity__c > 100
     *         Id IN ('000123', '000124')
     */
    global String getQueryFilter(ID accountId) {
        // all Asset Lines created by the current user
        return 'CreatedById = \'' + userId + '\' AND CreatedDate > LAST_MONTH';
    }

    global Boolean validateAssetTermination(Set<ID> assetIds, Set<ID>
accountIds, Date eDate) {
        return true;
    }

    global Date getAssetTerminationDate() {
        return Date.newInstance(2018,01,01);
    }

    /**
     * Callback to return the filter expression for the asset query where
clause
     * This filter is used in listing installed products
     * @param params the parameters for the method
     * @return the filter expression or null to use the default filter.
     * e.g. Name LIKE 'A%' AND Quantity__c > 100
     *         Id IN ('000123', '000124')
     */
```

```
    global String getFilterExpr(Apttus_Config2.CustomClass.ActionParams params)
{
        return getQueryFilter(null);
    }

    /**
     * Gets the asset search scope
     * @return the asset search scope or null to use the default asset search
scope
     */
    global List<String> getAssetSearchScope(){
        return this.assetSearchScope;
    }

    /**
     * Callback after the filter is used
     * Use the finish method to release state
     */
    global void finish() {

    }
}
```

2. Copy the custom callback class in the **Config Custom Classes** under **Asset Line Item Callback Class**. For more information, see Asset Line Item Callback Class.

This table shows how CPQ behaves with different combinations of the admin setting and the custom callback class.

APTS_EnableInstallBaseFilteringAcrossAccounts	Custom Callback Class	CPQ Behavior
True	Active	CPQ displays assets from multiple accounts.
False	Active	CPQ displays assets from the current account.
True	Inactive	CPQ displays assets from the current account.

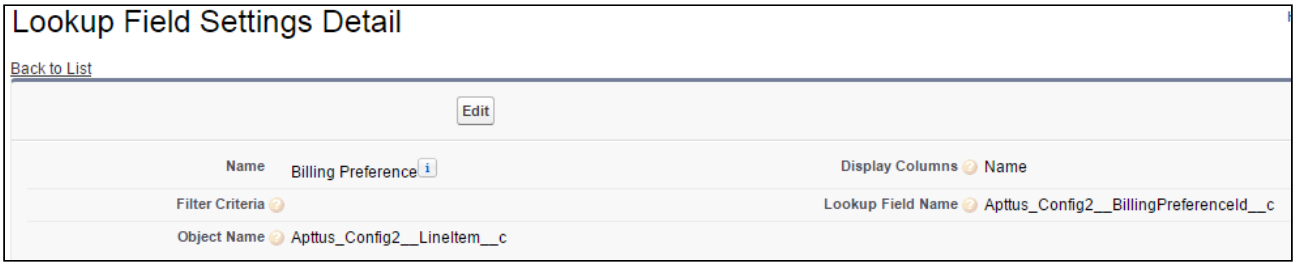
APTS_EnableInstallBaseFilteringAcrossAccounts	Custom Callback Class	CPQ Behavior
False	Inactive	CPQ displays assets from the current account.

Configuring Lookup Field Settings for Asset-Based Ordering

Each new lookup field shown in the cart needs a corresponding lookup field settings record. You must follow the explanation described here to create a new record for a lookup field settings.

Billing Preference look up setting

Field	Value
Name	Billing Preference
Display Columns	Name
Filter Criteria	Optional, to further filter the records
Lookup Field Name	API name of the Billing Preference field, *Apttus_Config2__BillingPreferenceId__c*.
Object Name	API name of the Line Item object, *Apttus_Config2__LineItem__c*



Agreement ID look up settings

Field	Value
Name	Agreement ID
Display Columns	Name
Filter Criteria	Optional, to further filter the records
Lookup Field Name	API name of the Agreement field, *Apttus_CMConfig__AgreementId__c*.
Object Name	API name of the Line Item object, *Apttus_Config2__AssetLineItem__c*.

Defining Allowed Actions for Assets

To meet specific business requirements, you can select which of the **Renew**, **Change**, **Swap**, **Terminate** actions are available for a sales user to perform for each Asset Line Item.

For example, you are an administrator for a news publication. Your company has several digital offerings that subscribers buy on a monthly term. One subscription has reached its end-of-life. To prevent you sales representatives from renewing a customer's subscription for this product, you can exclude **Renew** from the list of Allowed Actions.

To define the Allowable Actions manually,

1. Select the Asset Line Item and click **Edit**
2. From the **Allowed Actions** field, select the actions you want to enable and move them to the Chosen column.
3. Click **Save**.

Note

If you exclude any action, the corresponding button is grayed out on the Installed Products page. If you do not select any action, by default, all the action buttons are available to the sales user for use.

For **Renew**, **Terminate** and **Change** actions, you can select multiple Asset Line Items. For **Swap**, you can select one Asset Line Item at a time.

Defining Flow-based ABO Actions

Depending on your business units and type of purchase, the sales administrator can hide the action buttons on the Installed Products page. The ability to hide the action buttons is available with the flow settings defined for a Quote/Proposal. This is supported in conjunction with hiding the action buttons at the asset line item level (explained above) and global level (refer to Installed Product Settings).

Use case

For a sales-driven industry with products such as equipment (new purchase) and service (purchased for an equipment), you might want to achieve the following:

- As a manufacturing sales representative, for a service flow, you want to view the **Renew** and the **Terminate** button. For an equipment flow, you want to view the **Change** and the **Swap** button.
- As a subscription sales representative, you want to see the **Renew** button only when I make a purchase for a quote of type *renew*. For an add-on quote or an upgrade quote, you do not want to see the **Renew** button.

To define flow-based ABO action

You must have configured Installed Product settings and Flow settings.

1. Create a new flow using Flow Settings or choose an existing flow for your Quote/Proposal.
2. Go to **Setup > App Setup > Develop > Custom Settings > Config System Properties > Manage**. You will see an existing record named *System Properties*.
3. Create a new record with the same name as your Flow name.
   For example, if you have created a flow named *ngCPQflow*, enter *ngCPQflow* in the **Name** field.
4. In the **Hide Asset Actions** field, enter the values of action buttons that you want to hide for your flow. In our example, for a service flow, enter *Change* and *Swap*. For an equipment flow, enter *Renew* and *Terminate*.
5. Click **Save**.

About Integration of Assets with Contracts

CPQ allows you to manage assets through contracts. A contract or an agreement is a legally binding arrangement between two or more entities.

Prerequisites

- For the Assets and Contracts integration to work, the following packages must be available in your org:
    - Conga Contract Lifecycle Management
    - Conga Quote CLM Integration
    - Conga CLM Configuration Integration
- The **Auto Create Order** setting in **Comply System Properties** must be enabled if you want to create an order and asset as soon as an agreement is activated.

How asset and contract flows work

There are two process flows you must consider:

- Contract + Proposal Flow: In this flow, both Contract and Quote/Proposal are involved.
- Contract Flow: In this flow, only Contract is involved (no Quote/Proposal). The assets are created through contracts/agreements using the normal contract life cycle flow. After you finalize and activate a contract (containing agreement line items), the corresponding asset line items are created.

The following table explains the CPQ behavior when the **Auto Create Order** setting is configured.

Flow	Auto Create Order (Proposal System Properties)	Auto Create Order (Comply System Properties)	CPQ behavior
Contract + Proposal Flow	False	True	A Quote/Proposal is created followed by a Contract. The Asset Line Items and Order Line Items are created when you activate a Contract.

Flow	Auto Create Order (Proposal System Properties)	Auto Create Order (Comply System Properties)	CPQ behavior
Contract Flow	False	True	Contract is created without a Quote/Proposal. The Asset and Order Line Items are created when you activate a Contract.

> It is recommended to refrain from enabling **Auto Activate Order** from **Comply System Properties** and **Proposal System Properties** simultaneously.
> The Conga Contract Lifecycle Management package version must be 8.4.0325 (8.325.1) or higher to use the features mentioned above.

For more information, see Managing Assets through Contracts.

Creating Approval Workflows for Asset-Based Ordering

You can enable Advanced Approvals for your purchased products, using a few configuration steps.

Following are the sample use cases when you would want to trigger approvals:

- When your customer requests for the Renew of an asset.
- When your customer requests a change in the fields of an asset, such as Quantity or Discount Type, Discount Amount, and more.

Refer to Advanced Approvals Guide to understand and implement Advanced Approvals for your object, Assets Line Items. Since this is a custom object, you must ensure to use *Apttus_Config2__AssetLineItem__c* as the API name for customization in all of the procedures mentioned in the guide.

Once enabled, you see **Submit for Approval** button on the Cart, every time you perform action on the asset.

Configuring Mass Update for Assets

From the **Allow Mass Change** setting on the **Installed Products Settings** under **Custom Settings** page, you can enable or disable mass changes for *must configure* assets on the Installed Products page.

Action	Allow Mass Change	Type of the Asset	Type of selection	Behavior
Change	Unchecked	Must Configure	Single Select	Launch configure product page
Change	Unchecked	Must Configure	Multi Select	Disable Change
Change	Unchecked	Combination of Must configure and not must configure	Multi Select	Disable Change
Change	Unchecked	Not must configure	Single Select	Load into cart directly
Change	Unchecked	Not must configure	Multi Select	Load into cart directly
Change	Checked	Must Configure	Multi Select	Load into cart directly
Change	Checked	not Must Configure	Multi Select	Load into cart directly
Change	Checked	Must Configure	Single Select	Load into cart directly
Change	Checked	not Must Configure	Single Select	Load into cart directly

Creating Replacement Rules to Swap Assets

In order to use the **Swap** action for your purchased products, you must create a Constraint Rule of **Action Type** as *Replacement.* This rule fetches the available product in exchange of the product that you want to swap. Specify the product which you want to swap for in the Constraint Rule Condition and the product with which you want to swap to in the Constraint Rule Action.

> CPQ supports replacement rules on ABO carts only for Swap and Split-Swap.

For more information, see Managing Constraint Rules

Configuring the Custom Attributes Page

You can use a pre-defined custom attribute page while navigating from the Installed Products (Assets) page. By default, CPQ chooses the *Apttus_Config2__ProductAttributeDetail3* Visualforce page. This configuration is useful when you have designed a custom attribute page with the new look and design and you want to use it while configuring the attributes.

This capability allows customers to have their own UI implementation for the product configuration experience. You can enable Sales and Customer Service representatives to select options for assets and capture product attribute details for assets.

To customize your flow settings, perform the following steps.

1. Go to **All Tabs (⊞) > Config Settings** and click **Flow Settings**.
2. In the **Product Attribute Detail Page** field, enter the name of your custom attribute page.
3. Click **Save**.

Configuring Auto Renewal of Assets

CPQ allows Sales Representatives to auto-generate renewal quotes based on the fulfillment of order or the expiration of subscription. Through renewal quotes, they can forecast and analyze the sales pipeline. With renewal quotes, they get the finer visibility into the pricing and configuration of assets within the renewal pipeline. The benefits of auto renewals are:

- Accurate forecasting on the asset state and sales pipeline
- Faster and simpler processing with automated renewal
- Automatic closure of opportunity on asset expiry

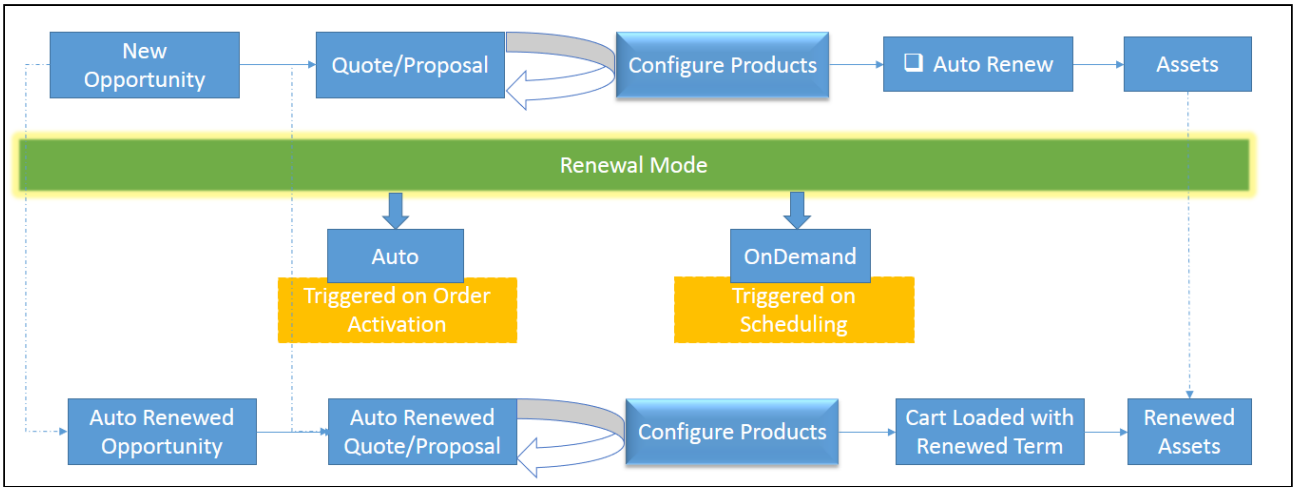
Types of Asset Renewal Modes

There are two types of renewal modes, defined in **Renewal Execution Mode**:

- Auto
- OnDemand

> It is recommended not to use these renewal modes interchangeably.

The following diagram explains how auto renewals work:



Configuring Renewal Settings

Access **Config Settings** > **Installed Products Settings** to see a bunch of following Renewal Field Settings.

Field	Description
Renewal Business Object Type	Enter the Business object for which this renewal is taking place. Currently, *Proposal* and *Agreement* are supported.

Field	Description
Renewal Default Price Book	Enter a Renewal Price Book name which you want to associate with renewals.
Renewal Execution Mode	Indicates if the Renewal of Asset Line Items must happen automatically or based on your specified conditions. • Enter **Auto** to automatically create the Renewal Opportunity on Order activation. Make sure you check **Auto Renew** on the product **PLI**, from the **Default** tab. • Enter **OnDemand** to create the renewal Opportunity before a certain lead time. You must enter a **Renewal Lead Time** for this mode to work successfully. To create a renewal Opportunity 90 days before the Asset Expiry, set the **Execution Mode** to *OnDemand* and **Renewal Lead Time** to *90*.
Renewal Group Fields	Indicates how to group the Asset Line Items when a Renewal Opportunity is created. Specify the API names of the fields you want to use for grouping. For Execution mode set to **Auto**, the system will group the renewal Opportunity by default by the **Auto Renew** Flag - One for the Asset Lines with **Auto Renew** as *True* and the other for Asset Lines with **Auto Renew** as *False*. For Execution mode set to **OnDemand**, the system can group the renewal Opportunity by Account and the Price List. However, if the implementation teams want to group the renewal opportunities by other parameters on the Asset Line Item, they can do so by specifying a comma separated list of API names of the fields in this section. It is recommended to limit the grouping to a maximum of 4 fields.
Renewal Lead Time	Indicates when the Renewal Opportunity will be created after an Order is activated. By default, the value is 0, which indicates that the Renewal Opportunity is created immediately after the Order is activated, with the same number of Line Items in the Order. If you specify 30, the Renewal Opportunity is created 30 days before the Asset End Date.
Default Renewal Price List	Specify the Name of the Price List which is a mandatory field for Quote creation.

Field	Description
Create Renewal Opportunity	Indicates if you want to create renewal opportunities along with renewal quotes. By default, this checkbox is not selected. This is to prevent the system from running into locking issues when an account has a large number of assets and the batch process creates the renewal opportunities for all these assets. When set to *true*, the Create Renewal Opportunity scheduled job runs and creates the renewal opportunities along with renewal quotes. When set to *false*, the Create Renewal Opportunity scheduled job runs and does not create the renewal opportunities and creates only the renewal quote.

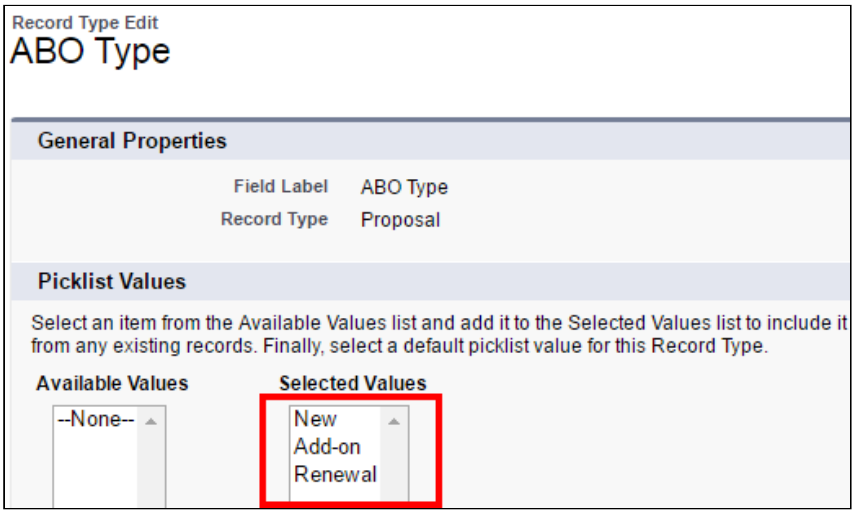
AutoSync Asset Changes to an Opportunity-Based Renewal Quote

When you select the **Sync Assets to Quote** flag on a proposal, whenever an asset is changed or updated, the asset line item in the Opportunity based renewal quote is also updated.

- Asset-based Orders sync does not support Swap and Cancellation actions.
- Blocking the Asset Based Order Synchronization should always be done in the renewal quote.
- Opting out of a renewal opportunity does not pull the line out from the auto-renewal quote.
- For Assets having **Auto Renewal Type** = *Evergreen*, no renewal opportunities and quotes are created and the **Renew** button is greyed out.

Adding Picklist Values to the Proposal Object

1. Go to the **Proposal** object
2. Click to open Proposal from the **Record Types**.
3. Edit the Picklist **ABO Type** from the list.
4. Add *New*, *Add-on*, and *Renewal* to the list of **Selected Values**.



Renewal Email Templates

To set up the email templates:

1. Navigate to **Setup** > **Communication Templates** > **Email Templates**.
2. Create templates for Asset Renewal Notification.
3. Select **Apttus CPQ Email Templates** from the **Folder** picklist to access existing templates or create a new one.

In the current version, you will not receive an Email notification on the generation of a Renewal Quote.

To schedule a batch job for auto renewal

You must schedule the batch job *AssetRenewalJobScheduler* for the first time. This job runs every 5 minutes after you have scheduled it.

1. Go to **Setup** > **Develop** > **Apex Classes**.
2. Click **Schedule Apex**. The Schedule Apex page is displayed.
3. In the **Job Name** field, enter a name for the job.
4. In the **Apex Class** lookup, search and select *AssetRenewalJobScheduler*.
5. Under **Schedule Apex Execution**, select the **Frequency**, **Start** date, **End** date, and **Preferred Start Time** for the job.
6. Click **Save**.

To monitor the job, go to **Setup** > **Monitoring** > **Apex Jobs**. You can see *AssetRenewalJobScheduler* in the list of jobs. You will get an email notification after this job is successfully completed.

To schedule a batch job for creating renewal opportunity

You must schedule the batch job *CreateRenewalOpportunityJobScheduler* for the first time. This job runs every 5 minutes after you have scheduled it.

1. Go to **Setup** > **Develop** > **Apex Classes**.
2. Click **Schedule Apex**. The Schedule Apex page is displayed.
3. In the **Job Name** field, enter a name for the job.
4. In the **Apex Class** lookup, search and select *CreateRenewalOpportunityJobScheduler*.
5. Under **Schedule Apex Execution**, select the **Frequency**, **Start** date, **End** date, and **Preferred Start Time** for the job.
6. Click **Save**.

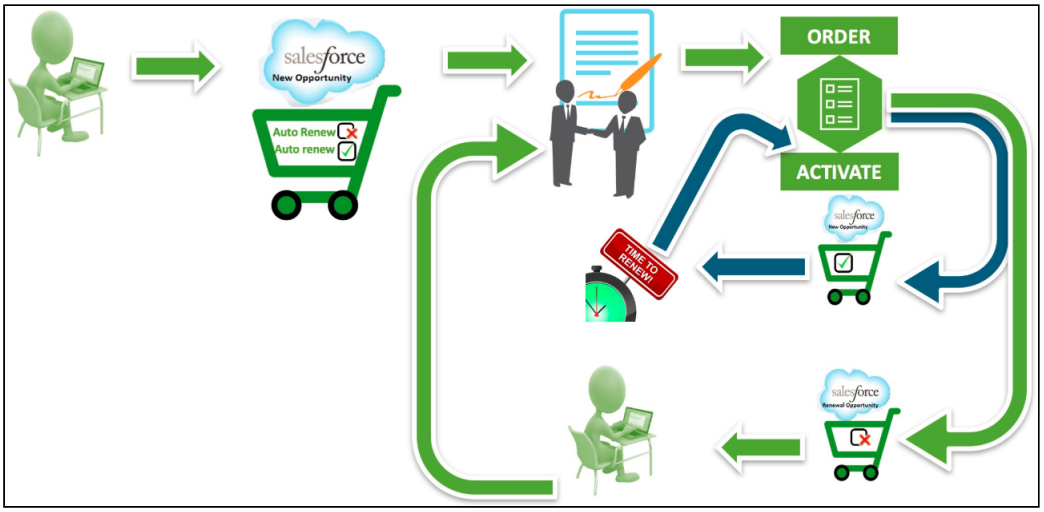
To monitor the job, go to **Setup** > **Monitoring** > **Apex Jobs**. You can see *CreateRenewalOpportunityJobScheduler* in the list of jobs. You will get an email notification after this job is successfully completed.

Configuring Asset Renewal in Auto Renewal Mode

When **Renewal Execution Mode** is set to *Auto*, CPQ automatically renews an existing quote for all assets it has, as soon as the order is activated. You may want to renew an existing opportunity in Auto renewal mode when:

- You require renewal quotes for forecasting
- You must work on renewal quotes in advance

The following diagram describes the flow of **Auto** renewals.



Prerequisites

- Set the **Renewal Business Object Type** to Proposal.
- Set the **Renewal Execution Mode** to *Auto*.
- Define the **Renewal Group Fields**.
- Schedule the AssetRenewalJobScheduler batch job.

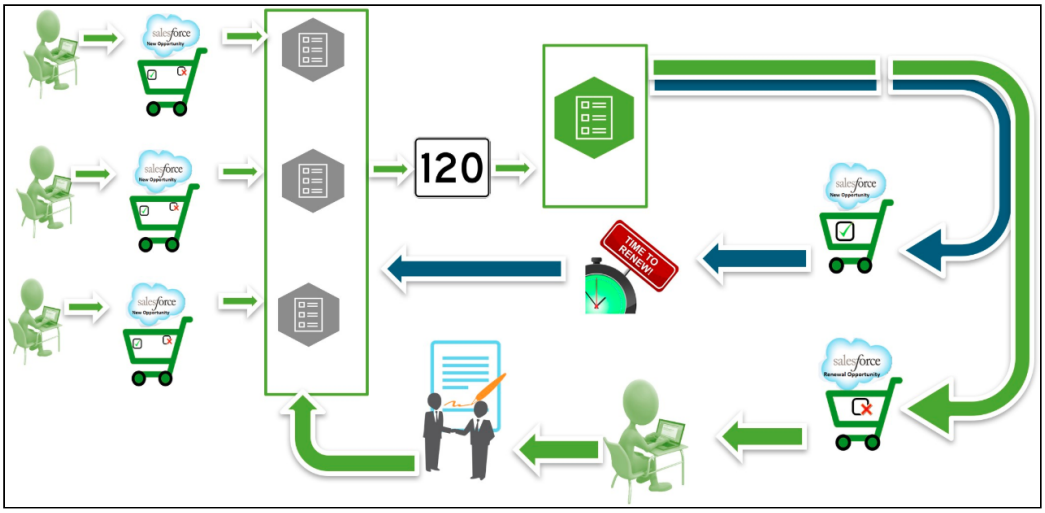
For more information, see Configuring Renewal Settings.

Configuring Asset Renewal in OnDemand Renewal Mode

When **Renewal Execution Mode** is set to *OnDemand*, CPQ automatically renews all assets on the account (possibly from different quotes) based on the defined **Renewal Lead Time** after you click a **custom button** on the account. You may want to renew an existing opportunity in OnDemand renewal mode:

- During one-time legacy asset data migrations
- During one time renewal quote creation for existing active asset base
- When minimal changes are required on the renewal quotes because of any ABO changes

The following diagram describes the flow of **OnDemand** renewals. In this diagram, **Renewal Lead Time** is 120 days.

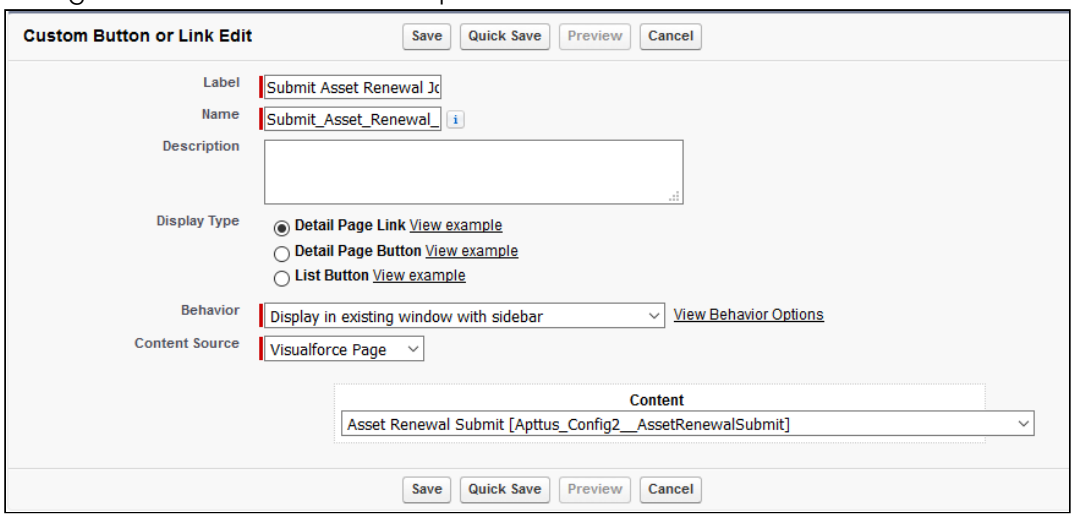


Over a period of time, you sell some products/services through different quotes, which might have varying end dates. After the original quotes are converted into orders and the order are activated, assets are generated. All assets are visible on the account. When you want to renew the assets that will expire in 120 days from today, you click the **custom button** on the account. CPQ compares the asset line items' end dates with the **Renewal Lead Time**

defined and triggers the creation of renewal quotes for all those assets (from different quotes) that will expire in 120 days.

Prerequisites

- Set the **Renewal Business Object Type** to Proposal.
- Set the **Renewal Execution Mode** to OnDemand.
- Define the **Renewal Lead Time**.
- Create the **custom button** on the account if you want to renew all assets in the account based on the defined Renewal Lead Time.
- Define the **Renewal Group Fields**.
- Schedule the AssetRenewalJobScheduler batch job.
- Navigate to Quote/Proposal and create a new link, set the content source as VF page and select *Apttus_Config2__AssetRenewalSubmit* as the VF page and click **Save**. If you want to renew assets from a quote based on the defined Renewal Lead Time, configure a custom link on the quote.



- To do this for multiple accounts, create a custom controller and VF page.
  Enter the following code in the custom controller. Create a reference for the custom controller in the VF page.

```
List<ID> accountIds =new List<ID>();


// gather account ids


// instantiate the OOTB controller and pass in the account ids
```

```
Apttus_Config2.AssetRenewalSubmitController baseController = new
 Apttus_Config2.AssetRenewalSubmitController(accountIds);


// submit the job

ID jobId = baseController.doSubmitJob();
```

For more information, see Configuring Renewal Settings.

Configuring Agreement-Based Renewals

CPQ supports Agreement-based renewals with a custom setting called **Renewal Business Object Type**, with value as *Agreement*. The entire functionality supported with Proposal object is supported with Agreement object. For information on how this feature works, Managing Assets in the Contract Flow.

For auto closure of renewal agreements, the **Status Category** of the agreement record should be one of the following:

- Request
- In Authoring
- In Signatures
- In Filing

The renewal agreement set for auto closure expires within 24 hours.
For manual submission of agreement-based renewals, you must create a new custom link for Agreement object, pointing to `Apttus_CMConfig_AgreementAssetRenewalSubmit` Visualforce page.

For more information on configuring data required to trigger renewal agreement creation, see Configuring Renewal Settings.

About Extending the Lead Time Functionality

A renewals administrator at a major healthcare company can be contractually mandated to create the renewal suite 90 days before the asset expiry for a large company and 30 days before asset expiry for a small company.

The CPQ Salesforce Installed products Settings such as Lead Time and Renewal Execution Setting is OnDemand a renewal quote is generated for all accounts globally. However, if

you want to generate renewals for a list of accounts for a particular lead time, you can use the following construct in the developer console.

```
// specify a lead time to override setting value
Integer leadTime = 1000;
// gather account ids
List<ID> accountIds = new List<ID>();
accountIds.add(‘0014C00000FQSuz’);
accountIds.add('0014C00000FQSuu');
// instantiate the OOTB controller and pass in the account ids
Apttus_Config2.AssetRenewalSubmitController baseController = new
 Apttus_Config2.AssetRenewalSubmitController(leadTime, accountIds);
// submit the job
ID jobId = baseController.doSubmitJob();
```

You can use the following construct to renew assets that belong to separate list of accounts for different lead times:

```
// create map to be used as the constructor argument
Map<Integer, List<ID>> accountIdsByLeadtime = new Map<Integer, List<ID>>();
// gather account ids to be renewed with a leadTime of 400
Integer leadTime = 1000;
List<ID> accountIds = new List<ID>();
accountIds.add('0014C00000FQSv9');
// add leadTime as key and accounts as value to map
accountIdsByLeadtime.put(leadTime, accountIds);
// gather account ids to be renewed with a leadTime of 1000
leadTime = 60;
accountIds = new List<ID>();
accountIds.add('0014C00000FQSv4');
// add leadTime as key and accounts as value to map
accountIdsByLeadtime.put(leadTime, accountIds);
// instantiate the OOTB controller and pass in the account ids
Apttus_Config2.AssetRenewalSubmitController baseController = new
 Apttus_Config2.AssetRenewalSubmitController(accountIdsByLeadtime);
// submit the job
ID jobId = baseController.doSubmitJob();
```

After executing the construct, view changes on the Temp Renew page. New records are created in the Temp Renew page.

Configuring Service CPQ

The Service CPQ feature caters to customer requirements related to services for assets (equipment purchased from Conga). A Sales Representative associates a service product with an asset. You can perform the following configurations to set up Service CPQ:

- Create service products by selecting **Product Type** as *Service*. For more information, see Creating Products.
- Create constraint rules to display relevant services on the Service Catalog for a selected asset. Service products and assets are associated based on the eligibility rules, which is the framework for service eligibility at large. The eligibility rules, which are client-side constraint rules, play a vital role in guiding the association between a service product and an asset.
  For example, when the user selects a specific asset, you want CPQ to hide some services on the Service Catalog. In the eligibility constraint rule criteria, you must define the Exclusion type constraint rule. The constraint rule action intent is scoped to other service products under a product family or product group to make them ineligible (block them) when the rule is triggered. For more information, see Creating Constraint Rule Conditions and Creating Constraint Rule Actions.
- The **Relate** button on the Installed Products page is displayed by default. You can change the label and alignment of the **Relate** button from **Display Action Settings > Display Type =** *Installed Products* for your flow name.
- Configure the **Submenu Actions** setting to display the **Relate Component** drop-down option on the **Relate** button. For more information, see Configuring Installed Products Settings.
- Set the **Relate Action Criteria Fields** to enable relate asset action. For more information, see Configuring Installed Products Settings.
- Create a price dimension for Service CPQ and use that price dimension in the required price matrix. For more information, see Creating Price Matrices and Dimensions.
- Define the **Service Price Distribution Method** setting to define the mode used for service pricing, at Setup> Develop> Custom Setting > Config System Properties. For more information, see Configuring Custom Settings.
- Define the **Service Line Split Criteria** setting to allow users to clone a service with some or all related line items using in the Service Cart, at Setup> Develop> Custom Setting > Config System Properties. For more information, see Defining Service Line Split Criteria.
- Configure the **Show Service Coverage** setting to display service coverage for a primary service, at Setup> Develop> Custom Setting > Installed Product Settings. For more information, see Configuring Custom Settings.

- Configure the columns to be displayed on the **Related Purchases** tab on the Installed Products page, using **Display Column Settings** > **Display Type** = *Related Purchases* for your flow name. For more information, see Configuring Display Columns Settings.
- Configure the **Related Line Item Fields** setting in **Field Set Settings** to display the required fields on the Related Line Items pop-up. For more information, see Configuring Field Set Settings.
- Configure the **Asset Line Fields For Selected Assets** setting in **Field Set Settings** to display the required fields on the Selected Assets pop-up. For more information, see Configuring Field Set Settings.

  > It is mandatory to display the **Quantity** field on the Selected Assets pop-up for CPQ to calculate the correct number of assets selected on the Installed Products page.

- Configure how CPQ should display service product attributes on the Configuration page, while creating product attribute groups. For more information, see Creating Product Attribute Groups.

Configuring Service Pricing

The service pricing flow is as follows:

- Create as many dimensions in the matrix as the number of your asset dimensions. Set up the matrix pricing for the service bundle and the service component individually. CPQ relies on the first match for the service price. Specify entries with less ambiguity (wild card) at the beginning of the matrix. For more information, see Creating Price Matrices and Dimensions.
- Configure the **Service Price Distribution Method** setting at Setup > Develop > Events > Custom Settings > Config System Properties. For more information, see Configuring Custom Settings.
- For **Relate Component - Bundle to Bundle scenario**: Related line items created only at the bundle level for the equipment items selected. Service is priced individually at the service bundle level and at the service option level for each related line item Individual service prices are then averaged to obtain the base price of the service line item and are then further rolled up to arrive at the price for the service line.
- For **Relate Component - Bundle to Component scenario:** Related line items created only at the component level for the selected components of the equipment (Note the behavior change from the previous releases – no related line items are created for the equipment bundles – you can see bundle associated with the component as an associated column on the same related line item). Service is priced

individually at the service bundle level and the at the service option level for each related line item. Individual service prices are then averaged to obtain the base price of the service line item and are then further rolled to arrive at the price for the service line.

- If there is no matching entry in the price matrix, it will take price as zero.
- CPQ derives the weightage of the service line from its contribution.
- Define split criteria to enable users to clone a service with some or all related line items using in the Service Cart. For more information, see Defining Service Line Split Criteria.

Defining Service Line Split Criteria

The Split feature allows users to clone a service with some or all related line items using in the Service Cart. The Split feature is available only at the service bundle level. You can define split criteria that apply globally or create split criteria for the specific flow.

> The **Service Line Split Criteria** setting only supports API names of the fields created on the Related Line Item object only. These fields can be formula fields and can fetch their values from Asset Line item and Product through formula expressions. CPQ supports the following types of custom fields on the Related Line Item object for split:
>
> - Text
> - Picklist
> - MultiPicklist
> - Formula (return type: Text)

To define global split criteria

1. Go to **Setup> Develop> Custom Setting > Config System Properties > Service Line Split Criteria.**
2. Specify the API names of the fields from the Related Line Item object on the basis of which you want to split the service.
3. Click **Save**.

The Split functionality is enabled on the Service Cart page.

To create a flow-based split criteria

1. Remove the values in the **Service Line Split Criteria** setting in Config System Properties.
2. Create a new config system property with the name that matches with the flow name of the **Configure Product** button.
3. Add the split criteria for the new Config System Property.
4. Click **Save**.

The **Split** button is displayed only for the cart grid that is created based on the flow name.

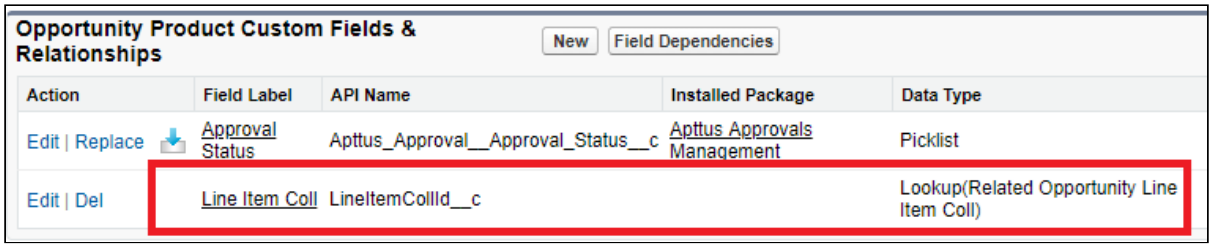
Synchronizing Related Line Items with Opportunity Line Items

As an administrator, you can enable synchronization of related line items with opportunity line items. The following are the methods of synchronization:

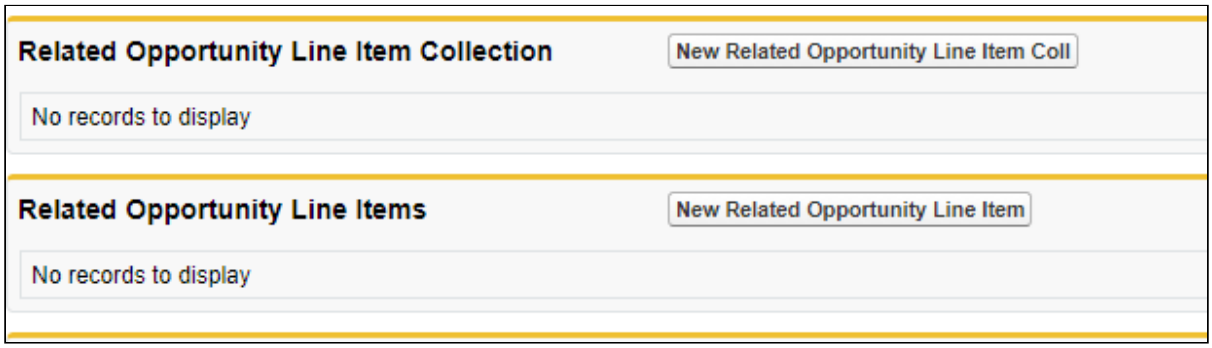
> ⓘ This feature is used only in Service CPQ.

Auto-Synchronization

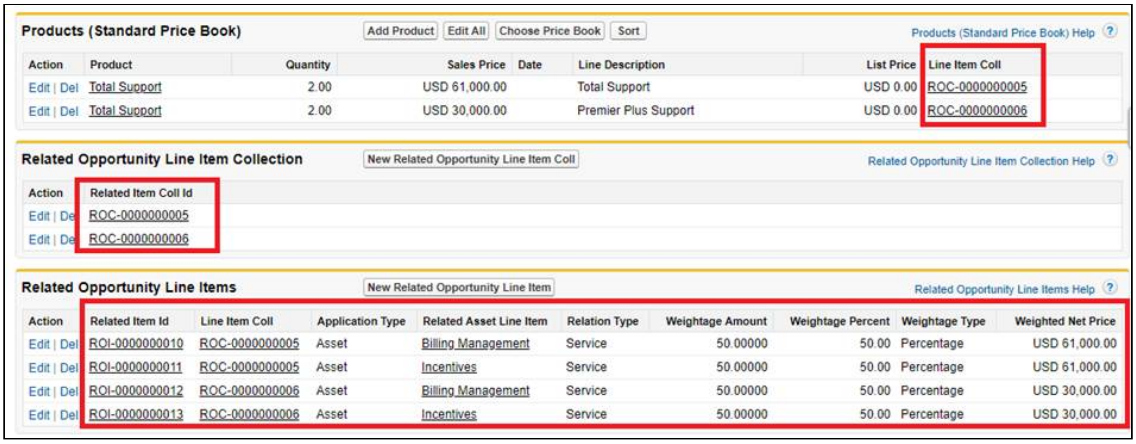
1. Enable the **Auto Sync With Opportunity** setting in the Proposal System Properties custom setting.
2. Create a lookup field with API name *LineItemCollId__c* on the *Opportunity Product* object with a lookup relationship to the *Related Opportunity Line Item Coll* object.



3. Display related lists **Related Opportunity Line Item Collection** and **Related Opportunity Line Items** on the opportunity layout.



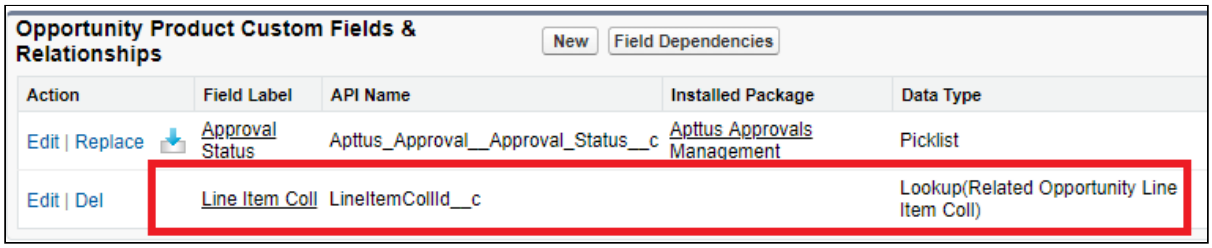
4. Ensure that the **Primary** checkbox is selected on the quote and the opportunity has a price book associated with it.
5. When you finalize the quote, you observe that:
    a. Proposal line items synchronize on the *Opportunity Product* object.
    b. The lookup field with API name *LineItemCollId__c* is populated with a related item collection ID.
    c. Each record in the *Opportunity Product* object has a unique related Item collection ID.
    d. The *Related Opportunity Line Item Collection* object links the opportunity product to the related line items associated with it. For every record in the *Opportunity Product* object, which has related lines associated with it, a unique *Related Opportunity Line Item Collection* is created.
    e. In the *Related Opportunity Line Item Collection* related list, the *Related Item Coll Id* of all collection records is displayed.
    f. The *Proposal Related Line Items* are synchronized with the *Related Opportunity Line Items* object and are displayed in its related list.



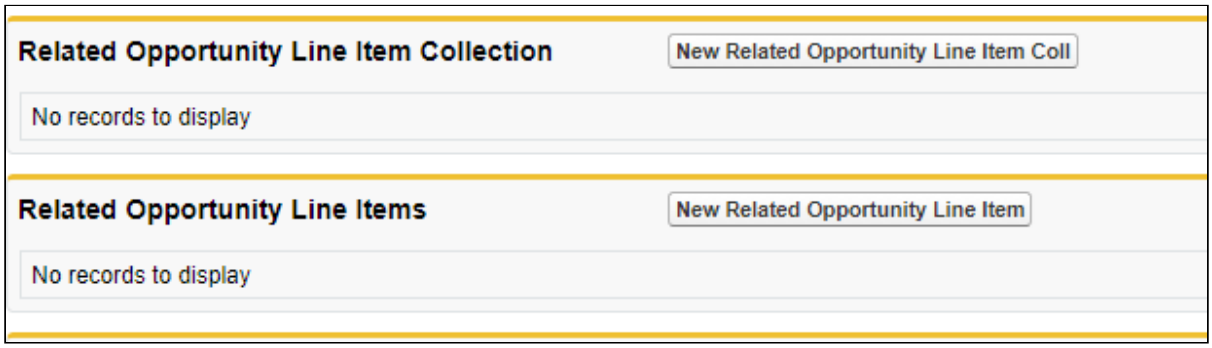
Manual Synchronization

1. Disable the **Auto Sync With Opportunity** setting in the Proposal System Properties custom setting.

2. Create a lookup field with API name *LineItemCollId__c* on the *Opportunity Product* object with a lookup relationship to the *Related Opportunity Line Item Coll* object.



3. Display related lists **Related Opportunity Line Item Collection** and **Related Opportunity Line Items** on the opportunity layout.



4. Ensure that the **Primary** checkbox is selected on the quote and the opportunity has a price book associated with it.
5. Ensure that **Synchronize with Opportunity** formula button in displayed on the proposal layout.
6. Finalize quote and click **Synchronize with Opportunity**.
   Observations are same as step 5 in Auto-Synchronization.

> ⓘ The options of the service bundle are synchronized to the opportunity only if the **Sync Option Products** setting is enabled in the Proposal System Properties custom setting.

Configuring the Execution of the RelatedLineItem Trigger

The out-of-the-box RelatedLineItem trigger, in it's update block, allocates the net price of the service line item to all associated related line item records based on the **Weightage Type** and **Weightage Amount** fields. However, some customers may have their own

allocation model and want to customize it based on their requirements. For such customers, the out-of-the-box RelatedLineItem trigger is redundant and can impact performance. CPQ allows you to control the execution of the RelatedLineItem trigger. You can use the global flag **isTriggerInitiatedUpdate** to conditionally skip the trigger logic, which allocates the net price of the service line item, and to avoid the unnecessary code execution and potentially improve performance in such cases.

The **isTriggerInitiatedUpdate** flag controls the "before and after update" block only. However, CPQ executes the "before and after insert" block regardless of the value of this flag. This flag remains set for the whole transaction until you unset the flag or until the transaction ends. Currently, CPQ respects this flag only in the RelatedLineItem trigger.

To configure the global flag

Purpose	Value of the Flag	Description
To set the global flag	Apttus_Config2 .RuntimeContext.getParameters ().put('isTriggerInitiatedUpdate',' true');	When the value of the flag is set to *True,* CPQ skips the execution of the RelatedLineItem trigger. CPQ does not use the out-of-the-box logic to allocate the net price of the service line item to all related line items on updating the related line items from the custom code.
To unset the global flag	Apttus_Config2.RuntimeContext .getParameters().put('isTriggerIn itiatedUpdate', 'false');	When the value of the flag is set to *False,* CPQ executes of the RelatedLineItem trigger. CPQ uses the out-of-the-box logic to allocate the net price of the service line item to all related line items based on the Weightage Type and Weightage Amount on updating the related line items from the custom code.

The following is an example of custom code for reference:

```
List<Apttus_Config2__RelatedLineItem__c> RLIs = [SELECT Id,
Apttus_Config2__WeightageType__c, Apttus_Config2__WeightageAmount__c FROM
```

```
Apttus_Config2__RelatedLineItem__c WHERE Apttus_Config2__LineItemId__c =
'a132i0000012WS9'];
RLIs.get(0).Apttus_Config2__WeightageType__c = 'Percentage';
RLIs.get(1).Apttus_Config2__WeightageType__c = 'Percentage';
RLIs.get(2).Apttus_Config2__WeightageType__c = 'Percentage';
RLIs.get(0).Apttus_Config2__WeightageAmount__c = 30;
RLIs.get(1).Apttus_Config2__WeightageAmount__c = 30;
RLIs.get(2).Apttus_Config2__WeightageAmount__c = 40;

Apttus_Config2.RuntimeContext.getParameters().put('isTriggerInitiatedUpdate','true');
Database.update(RLIs);
```

Here the **isTriggerInitiatedUpdate** flag is set to *True*.

Configuring the Cart

You can configure the following feature available on the cart.

- Configuring Flow Settings
- Creating Custom Buttons for Different Flows
- Sorting Main Categories on the Catalog Page
- Configuring the Location-Based Cart
- Configuring Column Settings on the Cart and Installed Products Page
- Configuring the Number of Products Displayed on the Catalog Page
- Configuring Custom Attribute Pages
- Hiding the Add to Cart and Configure Buttons for Option Products
- Displaying Leaf Products on the Catalog Page
- Cloning Sub-Bundles, Options, and Attributes on the Options Page
- Enabling Multiple Adjustments at the Line-Item Level
- Configuring Option Net Adjustment Rollup to Bundle
- Enabling Price Breakup for Products
- Displaying Attributes and Options Inline on the Configuration page or a Visualforce Page
- Sequencing the Favorite Category on the Catalog Page
- Configuring the Number of Option Products Displayed on the Cart
- Configuring Save as Favorite
- Configuring the Smart Cart
- Configuring Product Sort
- Configuring Cart Views
- Configuring Cart Activity History
- Configuring CSS Override

- Configuring Mass Option Selection on the Configuration Page
- Configuring Express Proposal

As compared to the existing setup, the following features are not introduced in this release:

Feature	Description
Dependent Picklist for Attributes	Salesforce Dependent Picklist is not yet supported on the new CPQ user interface. Instead of using SalesForce Dependent Picklist, you can drive the product attribute selection using the Attribute Value Matrices of CPQ. By using Attribute Value Matrices, you no longer need to maintain the attribute relationships in the SalesForce object and instead you can use admin interface.
Price Breakdown for cumulative tiers and related pricing	Price breakdown information icon is not displayed for the base price on the Cart page.
Custom Labels support for Custom Display Actions	New UI does not support the creation of a new label. You need to use one of the available Custom Action labels and change the key in the Display Action Setting. You should change the Custom Display action name under the Display Action Setting and refer to one of the standard "Custom Action" labels that are provided by the managed package.

The new user interface allows you to select products, configure their attributes and options, define pricing and make adjustments, select installed products, Change, Swap and Cancel orders on one single, user-friendly page.

Configuring Flow Settings

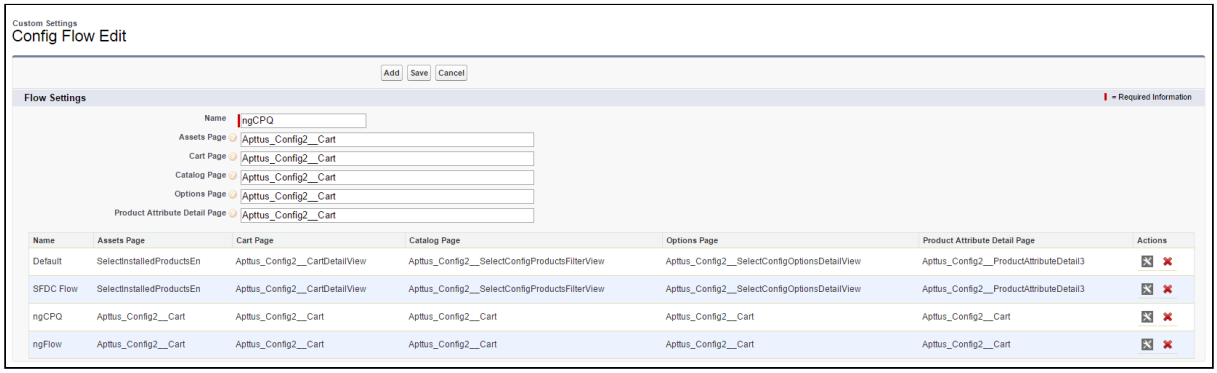
Flow settings are required so that appropriate pages are loaded for various sections (such as Catalog, Product Configuration, Cart, and Asset). You can configure options and attributes in a single page.

To configure Flow Settings for the new user interface

1. Go to **All Tabs (⊞) > Config Settings** and click **Flow Settings**. The page to create a new Flow setting opens.

2. Create a new record for the flow setting.
   - Or -
   Edit an existing record by clicking  in the **Actions** column.
3. Enter requisite information as explained in the following table:

Field	Description
Name	Enter a mandatory name for the flow setting. For example, enter ngFlow since we are defining the flow for the new user interface. ⓘ Do not use special characters in the name.
Assets Page	Enter the name of the VF page for displaying the Installed products or Assets. For example, *Apttus_Config2__Cart*.
Cart Page	Enter *Apttus_Config2__Cart* Visualforce page to access the new Cart page. If you want to view the cart with the new Grid UI, name the cart page as *Apttus_Config2__Cart#cartgrid.*
Catalog Page	Enter *Apttus_Config2__Cart* Visualforce page to access the new Catalog page.
Options Page	Enter *Apttus_Config2__Cart* Visualforce page to access the new Options page.
Product Attribute Detail Page	Enter *Apttus_Config2__Cart* Visualforce page to access the new Product Attributes page.



4. Click **Add** if you have created a new record for flow setting.
   - Or -
   Click **Save** to save your modifications to existing flow setting record.

Once saved, you can also configure the columns and buttons for the different pages in your custom flow by configuring the **Display Column Settings** and **Display Action Settings**.

Creating Custom Buttons for Different Flows

To use a different flow for CPQ based on the flow settings, you must create a custom button for the Quote/Proposal object. The custom buttons is named **Configure Product** with a custom label you define in the formula.

1. Go to **Setup > Create > Objects** and search for **Quote/Proposal** object.
2. In the **Custom Fields & Relationships** related list, click **New**.
3. From **Step 1. Choose the field type**, choose **Formula** as Data Type and click **Next**.
4. From **Step 2. Choose the output type**, Type *Configure (Flow)* in the **Field Label**. Choose **Text** as **Formula Return Type**.
5. In **Step 3. Enter formula**, you must paste one the following URL under the **Simple Formula** tab.

URL	Description
Regular	This is the regular URL available for **Configure Products** button. `IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/ Apttus_QPConfig__ProposalConfiguration?id="&Id&"&flow= ngFlow" , IMAGE("/resource/ Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"), NULL)`

URL	Description
Optimized	You can use the optimized URL to launch Catalog & Cart quickly compared to the above-mentioned URL. The reduction in time to launch Catalog and Cart compared using the optimized URL may vary and in some Orgs, the benefits may be minimal. The optimized URL primarily requires the following parameters: • Business Object ID • Flow All other parameters are supported. You can update the existing Configure Products formula with the optimized launch URL. If the org has existing customization, you must test and verify the optimized URL after replacing the existing URLs. The following mentioned URL is an example of an optimized URL for the **Configure Products** button. `IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/Apttus_Config2__Cart?businessObjectId="&Id&"&flow=ngFlow", IMAGE("/resource/Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"), NULL)`

ⓘ Ensure that you enter the name of your Flow setting in the *flow* parameter in the above formula. In our example, it is *ngCPQ*.

6. Click **Next**
7. Select appropriate Field Level Security and click **Next.**
8. Choose the page layouts on which you want to display this field.
9. Click **Save**.

You can use to following parameters in the formula to achieve different tasks.

Parameter	Description
productOrderByClause	Use this parameter to sequence the products within a Category. Refer to Sequencing Products within a Category for more details.

Parameter	Description
asyncFinalize	Set the value of this parameter to *True* to finalize any cart asynchronously. When the Sales rep clicks **Finalize**, the cart closes and the Sales rep is redirected to the Quote Details page. The finalization process is completed in the background. The Sales Rep must refresh to see the changed product configuration status. If the parameter value is *False*, then the cart is finalized without closing.
isCartTotalingDisabled	Set the value of this parameter to *True* to selectively skip totaling of pricing of products on the Cart, at a transaction level. When you add products on the cart, the totals section is not updated. ⓘ When you see the value to *True*, CPQ does not execute Pricing Callback in ADJUSTMENT mode.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>useAdvancedApproval</td><td>
Set the value of this parameter to False to selectively skip the approval of cart, at a transaction level. CPQ invokes the approval process only if this flag is set to True.

If any workflow processes are setup to trigger approvals based on criteria, they will occur when the criteria are matched. The Approval Stage inside the Quote/Proposal record is set to Approval Required. When the Approval Stage changes, the Submit for Approval button becomes available and you can submit the Quote for approval to management if any approval processes have been configured.

If this parameter is set to False or if it is absent, CPQ skips the approval process even when the criteria are matched.

To support quote approvals, you must use the isCartApprovalDisabled parameter.
<table>
<tr><th>useAdvancedApproval</th><th>isCartApprovalDisabled</th><th>Approval Type</th><th>Approval Check</th></tr>
<tr><td>True</td><td>True</td><td>Quote Approval</td><td>No</td></tr>
<tr><td>True</td><td>False or absent</td><td>Cart Approval</td><td>Yes</td></tr>
<tr><td>False or absent</td><td>True</td><td>None</td><td>No</td></tr>
<tr><td>False or absent</td><td>False</td><td>None</td><td>No</td></tr>
</table>
When the useAdvancedApproval parameter is enabled, it enables cart approvals automatically unless cart approval is explicitly disabled. When cart approval is disabled, CPQ reverts to quote approvals.
</td></tr>
</table>

Parameter	Description
useDealOptimizer	Set the value of this parameter to *False* to selectively skip deal guidance, at a transaction level. CPQ shows the deal guidance on the Cart page only if this flag is set to *True*. If you have created rules that define the various criteria of a good deal, the deal rating is available to the Sales Representatives at the time of negotiation. If this parameter is set to *False* or if it is absent, CPQ skips the deal guidance and does not show on the Cart page.
deferPricingUntilCart	Set the value of this parameter to *True* to set Defer Pricing on, at a transaction level. When Defer Pricing is on, CPQ performs pricing of products only after you click **Go to Pricing**. When Defer Pricing is off, CPQ performs pricing as and when you add or delete products.
useAdvancedCurrency	Set the value of this parameter to *True* to enable dated currency conversion on the Cart, at a transaction level.
styleSheetURL	Define the URL to the custom style sheet as the value of this parameter to override the default style sheet in CPQ.

Accessing the Cart in Read-Only Mode

CPQ enables you to create a Cart in which your Sales rep can just view the product configurations and not make any changes to the field values for the Product Line Items. In order to achieve this functionality, you must specify a new parameter *mode=readOnly* while creating a custom button to launch the New UI.

```
IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 ,HYPERLINK("/apex/
Apttus_QPConfig__ProposalConfiguration?id=" &Id& "&flow=ngFlow"&"&mode=readOnly",IMAG
E("/resource/Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"),
NULL)
```

By applying this formula to your Configure button, the checkboxes of products and the action buttons are disabled on the Cart page when the sales rep accesses the cart in read-only mode. This prevents your sales rep from applying any mass updates to the products in the Cart. In read-only mode, the Sales rep can still create views, search the products, and filter the line items on the cart.

Configuring Custom Button to Access Configuration Page Directly

You can define a formula action to configure products and service bundles directly from any page in the org without having to navigate through the Catalog or Cart page. The custom button allows the Sales rep to configure products or service bundles from any other page in the org by opening the Configuration or Service Configuration pages directly. The Sales rep can configure attributes, options, sub-bundles, and related line items on the configuration pages. For example, you can add the custom button on the Quote Detail page or the Line Item Detail page.

> ⓘ For Service CPQ flow, service products and related line items must already be configured before using this feature. You cannot add a new service bundle to the cart using this feature.

You can define the custom button to either add a new product to the cart or edit an existing product or service in the product configuration which is saved or finalized. For service bundles, you can only configure existing service bundles. The formula for the custom button is different based on the functionality you want to define.

You must modify URL structures to either add a new product to the cart or edit an existing product in the product configuration.

- You must use the following structures of URL to navigate directly to the Configuration page.

Task	URL
Add a new product	*<instanceURL>/apex/Apttus_Config2__Cart#!/flows/<flow name>/businessObjects/<businessObjectsId>/products/<productId>/configure*
Edit an existing product or service bundle in the quote	*<instanceURL>/apex/Apttus_Config2__Cart#!/flows/<flow name>/businessObjects/<businessObjectsId>/lines/<primaryLineNumbers>/configure*

- To modify the URL you must gather the following values.

Value	Description
Instance URL	You can find the Instance URL from the URL of any page in your org. For example, ***https://cpq--test.my.salesforce.com****/apex/Apttus_Config2__Cart#!/flows/CartGrid/businessObjects/a0Y7A000001kWce/products/01t3C000002Rao3/configure.*
Flow name	The name of the flow using which you want to configure your product. For example, *https://cpq--test.my.salesforce.com/apex/Apttus_Config2__Cart#!/flows/**CartGrid**/businessObjects/a0Y7A000001kWce/products/01t3C000002Rao3/configure.* ⓘ Smart Cart flows and Partner Portal are not supported.
Business Object ID	The ID of your quote/proposal, order, or agreement which you can find in the URL of the Quote Detail page. For example, *https://cpq--test.my.salesforce.com/apex/Apttus_Config2__Cart#!/flows/CartGrid/businessObjects/**a0Y7A000001kWce**/products/01t3C000002Rao3/configure.*
Product ID	The ID of the product which can find in the URL of the Product Detail page. You need the ID of the product to add a new product to the cart. For example, *https://cpq--test.my.salesforce.com/apex/Apttus_Config2__Cart#!/flows/CartGrid/businessObjects/a0Y7A000001kWce/products/**01t3C000002Rao3**/configure.* ⓘ In the following scenarios, the Sales rep is directed to the Catalog page instead of the Configuration page and CPQ displays anerror message: • The Product ID provided is incorrect • The Product ID belongs to a different price list that is not associated with the Business Object • The Business Object is not associated with any price list.
Primary Line Numer	The Primary Line Number of a product is generated on the line item after the product is added to the cart. You can find the Primary Line Number of the product on the Line Item Detail page. You need the Primary Line Number of the product to edit an existing product in the configuration. For example, *https://cpq--test.my.salesforce.com/apex/Apttus_Config2__Cart#!/flows/QuotingCartGrid/businessObjects/a0Y7A000001kWce/lines/**1**/configure.*

The following use cases describe how to define both the functionalities. The custom button on the Quote Detail page allows the Sales rep to add a new product or service each time

the button is clicked. However, the custom button on the Line Item Detail page allows the Sales rep to update and modify the existing product or service.

Custom Button on Quote Detail Page to Add a New Product

On the Quote Detail page, the Sales rep must provide the URL of the product in a custom text field to open the Configuration page for that product. When the Sales rep clicks the custom button on the Quote Detail page, the product corresponding to the URL provided in the custom text field is added to the cart and the Configuration page of that product is displayed. If the Sales rep clicks the button multiple times, the same product is added to the cart again. You must create a custom text field on the **Quote/Proposal** object where the Sales rep can enter the product URL.

Pre-requisite

Create a custom text field in the **Quote/Proposal** object.

To create a custom button on Quote Detail page

1. Go to **Setup > Create > Objects** and search for **Quote/Proposal** object.
2. In the **Custom Fields & Relationships** related list, click **New**.
3. From **Step 1. Choose the field type**, choose **Formula** as Data Type, and click **Next**.
4. From **Step 2. Choose the output type**, Type *Configure (Flow)* in the **Field Label**. Choose **Text** as **Formula Return Type**.
5. In **Step 3. Enter formula**, paste the following URL under the **Simple Formula** tab and click **Next**:

```
IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/
Apttus_Config2__Cart#!/flows/CartGrid/businessObjects/" &Id & "/products/"&Prod
uctId__c&"/configure" , IMAGE("/resource/Apttus_Config2__Button_Configure",
"Configure Products"),"_self"), NULL)
```

> Ensure that you enter the name of your Flow setting in the *flow* parameter in the above formula. In our example, it is *CartGrid*.
> Ensure that you replace *ProductId__c* with the name of the custom text field that you created in the **Quote/Proposal** object.

6. Select appropriate Field Level Security and click **Next.**
7. Choose the page layouts on which you want to display this field.
8. Click **Save**.

Custom Button on Line Item Detail page to Edit an Existing Product or Service Bundle

You can add the custom button on the Line Item Detail page to allow the Sales rep to update or modify the existing product configuration of a product or a service bunble. The formula for the custom buttons refers to the Primary Line Number that is generated by CPQ upon adding a product to the cart, to open the Configuration page for the specific product or service bundle.

To create a custom button on the Line Item Detail page

1. Go to **Setup > Create > Objects** and search for **Line Item** object.
2. In the **Custom Fields & Relationships** related list, click **New**.
3. From **Step 1. Choose the field type**, choose **Formula** as Data Type, and click **Next**.
4. From **Step 2. Choose the output type**, Type *Configure (Flow)* in the **Field Label**. Choose **Text** as **Formula Return Type**.
5. In **Step 3. Enter formula**, paste the following URL under the **Simple Formula** tab and click **Next**:

```
IF ( LEN( Apttus_Config2__PriceListId__c ) > 0 &&
TEXT(Apttus_Config2__LineType__c) = "Product/Service" , HYPERLINK("/apex/
Apttus_Config2__Cart#!/flows/CartGrid/businessObjects/" &
Apttus_Config2__ConfigurationId__r.Apttus_QPConfig__Proposald__c & "/lines/" &
TEXT(Apttus_Config2__PrimaryLineNumber__c)&"/configure", IMAGE("/resource/
Apttus_Config2__Button_Configure", "Config"),"_self"), NULL)
```

> Ensure that you enter the name of your Flow setting in the *flow* parameter in the above formula. In our example, it is *CartGrid*.

6. Select appropriate Field Level Security and click **Next.**
7. Choose the page layouts on which you want to display this field.
8. Click **Save**.

Configuring a Custom Button to Launch the Installed Products Page Directly for the Add/Remove Flow

You can launch the Installed Products page directly for the Add/Remove flow (Service CPQ), where you can modify the association of assets with services. CPQ enables you to define

parameters on the URL to launch the Installed Products page directly from the custom cart page.

The following is the syntax of the URL:

```
<instanceURL>/apex/Cart?
businessObjectId=<businessObjectId>&flow=<flowName>&servicePLN=<servicePrimaryLineNum
ber>#!/assetsgrid
```

Here you can pass the value servicePLN as follows:

- servicePLN=1, which means the primary line number 1, representing the first service product
- servicePLN=2, which means the primary line number 2, representing the second service product

To create a custom button

Perform the following steps to create a custom button:

1. Go to **Setup** > **Create** > **Objects** and search for the **Quote/Proposal** object.
2. In the **Custom Fields & Relationships** related list, click **New**.
3. From **Step 1. Choose the field type**, choose **Formula** as Data Type, and click **Next**.
4. From **Step 2. Choose the output type**, enter the following details and click Next:
    a. In the **Field Label** field, enter Configure Products <Flow>.
    b. In the **Field Name** field, enter Configure_Products_<Flow>.
    c. Choose **Text** as **Formula Return Type**.
5. In **Step 3. Enter formula**, paste the following URL under the **Simple Formula** tab and click **Next**:

    ```
    IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/Cart?
    businessObjectId=<businessObjectId>&flow=<flowName>&servicePLN=<servicePrimaryL
    ineNumber>#!/assetsgrid", IMAGE("/resource/Apttus_QPConfig__Button_Configure",
    "Configure Products"),"_self"), NULL)
    ```

6. In **Step 4. Establish field-level security**, select the appropriate Field Level Security and click **Next**.
7. In **Step 5. Add to page layouts**, choose the page layouts on which you want to display this field.
8. Click **Save**.

Example URL:

```
IF ( LEN( Apttus_QPConfig__PriceListId__c ) > 0 , HYPERLINK("/apex/
Apttus_Config2__Cart?businessObjectId="&Id&"&flow=LAngflow&servicePLN=1#!/assetsgrid",
 IMAGE("/resource/Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"),
NULL)
```

Here servicePLN=1 means the primary line number 1, representing the first service product.

Sorting Main Categories on the Catalog Page

Using this feature, you can decide the sequence of the main categories. You can control the sequence of the category using the new field called **Sequence** on the Price List Category record under the Category. This helps you to customize and have a better control over the way your categories are sequenced on the Catalog page.

You must have categories created for grouping the bundle and standalone products.

To implement this enhancement:

1. Go to **All Tabs** (), search for **Categories** and click on it.
2. Select the main category you want to sequence.
3. In your category record, look for **Price List** related list.
4. Click **Edit** for an existing price list category.
   - Or -
   Click **New Price List Category** to create a new record for associating your category to the price list.
5. In the **Sequence** field, enter the number in which you want to sequence the selected category.
   If you are creating a new price list category record, select the required price list.
6. Click **Save**.
7. Perform the above steps for all the necessary main categories.

All the main categories will appear on the Catalog page in the sequence you defined.

Configuring the Location-Based Cart

Just like Enterprise CPQ, the new AngularJS based CPQ also allows grouping of products based on the location of the customer. These locations are in sync with the locations specified at the Opportunity or Account from where you create a Quote. Sales representatives can easily differentiate between products according to the location of the customer. The filtration of products is performed before loading the Cart page.

Once you save or finalize your configuration in the Enterprise CPQ, when you configure the products in the new AngularJS based cart, you have the option to group the products by location or product.

> **Note**
>
> Ensure that you have selected **Enable Location** check box available at **Setup > Develop > Custom Settings > Config System Properties > Manage > System Properties**.

To enable this feature in the new AngularJS based cart,

1. On your Quote/Proposal record page, search for **Proposal Location** related list and click **New Proposal Location**.
2. Create records using the look-up icon, in sync with the locations specified in the Account or Opportunity from which you have created your Quote/Proposal. For example, USA, India, UK, and China.
3. Click **Configure Products** for your Enterprise CPQ.
4. On the top-right corner, use the **Location** drop-down list to configure the products for different locations and add them to the Cart. For example, choose 2 products for the India location, 3 products for the US and so on.
5. After configuring the options and attributes for the chosen products, click **Save** on the Cart page.
6. Once you return to the Quote/Proposal detail page, click **Configure Products (new-CPQ)** or the button which you have defined for configuring the products using the new AngularJS CPQ.
7. Just above the products list, choose **Group By > Location** to see the products filtered according to the configuration you did in Step 4. You will see the products differentiated under each location.
   Not Used in Quote under a location indicates that either no product is linked to the Quote/Proposal or you have not created a **Proposal Location** record for this.

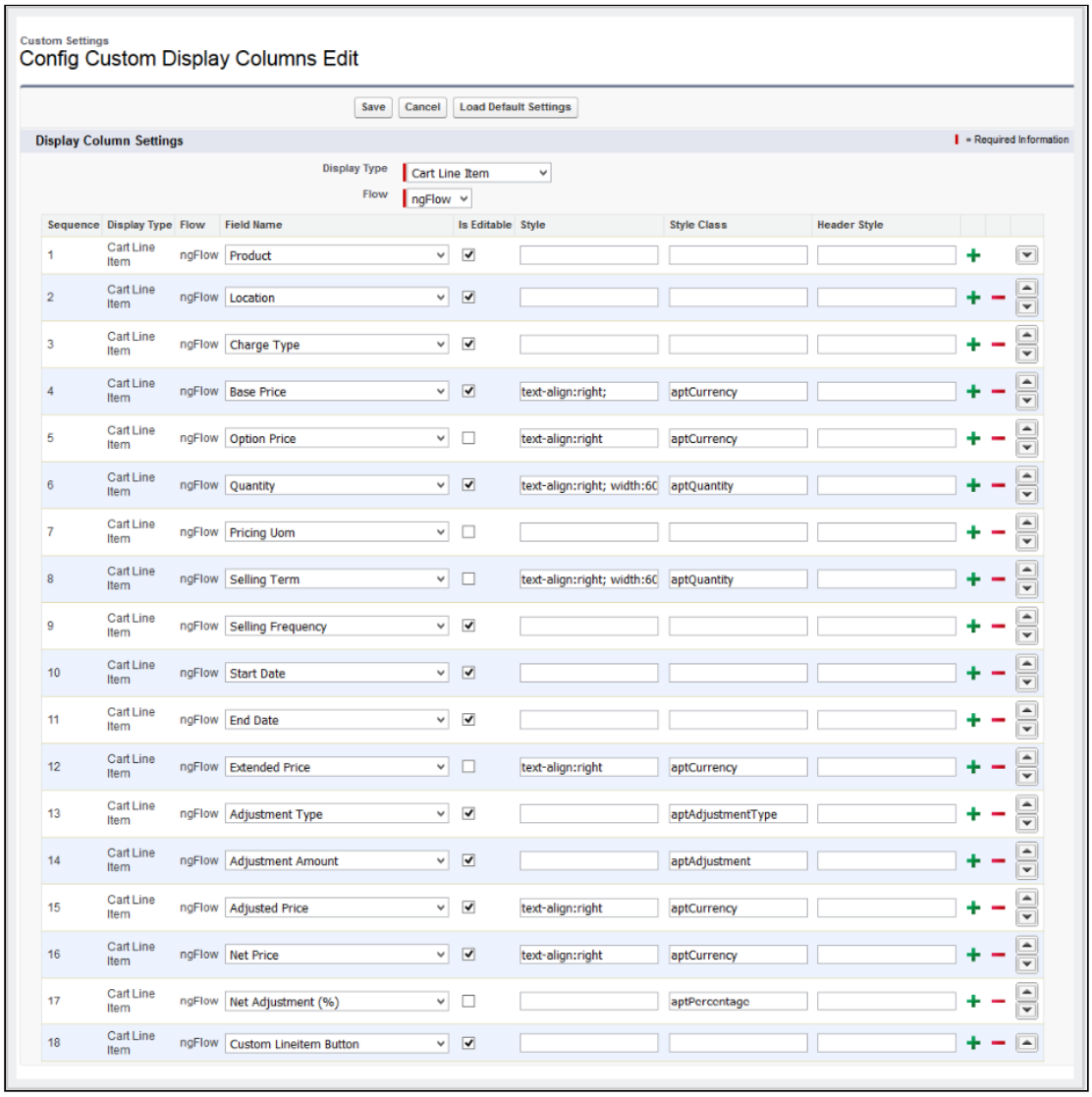
> **Important**
>
> Ensure that you have created the **Proposal Location** records in sync with the locations of your Account or Opportunity, else you may not see the product grouping by location.

Configuring Column Settings on the Cart and Installed Products Page

The custom settings for columns defined at **All Tabs > Config Settings > Display Column Settings** are now applicable for the new AngularJS CPQ as well. Using this custom setting, you can control column style, column header, sections such as Cart Line Items and Totals which are displayed on the Cart and Installed Products (Asset) page.

To use the above custom settings, you must have defined a flow for the new AngularJS CPQ under **All Tabs > Config Settings > Flow Settings**.

1. Go to **All Tabs > Config Settings** and click **Display Column Settings.**
2. From the **Display Type** drop-down list, select the required type in which you want to customize the columns. For example, if you want to customize the columns in the main Cart Line Item section, containing columns such as Product Name, Charge Type, and more are shown, choose *Cart Line Item*.



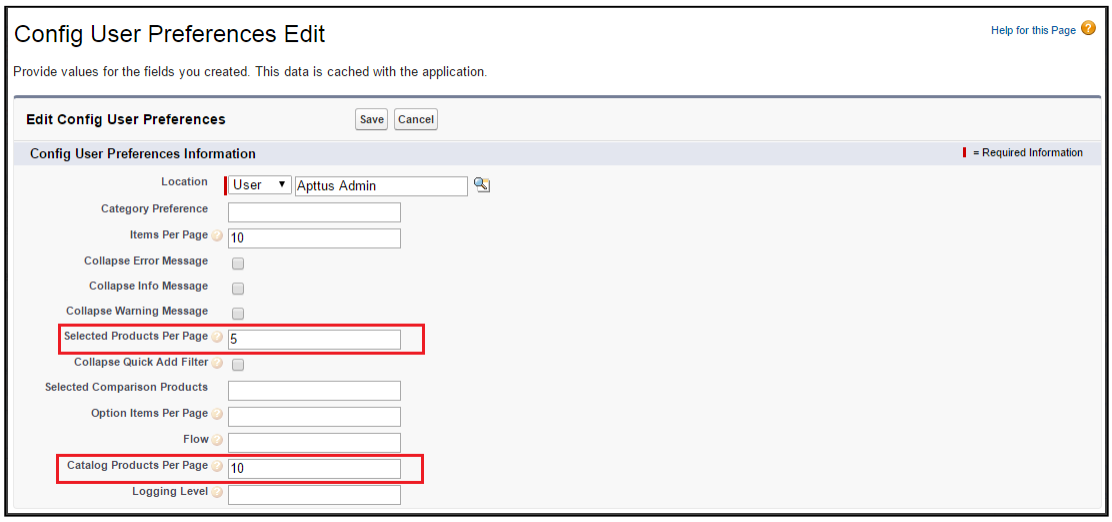
3. From the **Flow** drop-down list, choose the name of the Flow settings you have defined under **All Tabs > Config Settings > Flow Settings**.
4. Specify the requisite information in the **Style**, **Style Class** and **Header Style** for each field, based on your requirement.
5. Click **Save**.

Configuring the Number of Products Displayed on the Catalog Page

For the New UI, customize the following custom setting to display the number of products on the Catalog page for a specific user. The products are divided across the pages using the Pagination concept.

To customize the user preferences

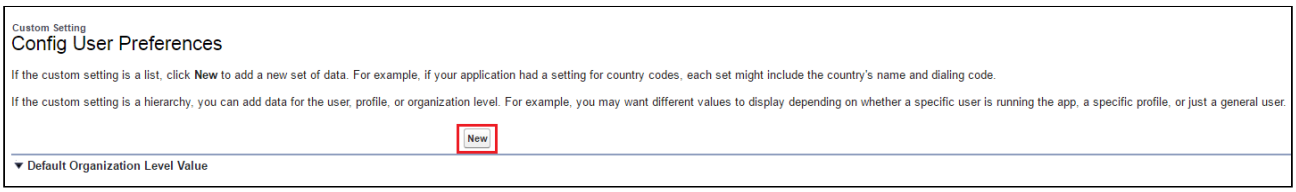
1. Go to **Setup > Develop > Custom Settings** and search for **Config User Preferences.**
2. Click **Manage** next to **Config User Preferences**.
3. For the required user, do any of the following:
   Click **View** to view the details and click **Edit**.
   - Or -
   Click **Edit** to edit and make changes to the existing user preferences.



4. In the **Catalog Products Per Page** field, enter the number of products that you want to display on the Catalog page per page. Valid values are *10*, *20*, *50*, and *100*. This setting is available, especially for the new user interface.
   You may also customize the number of products in the mini cart on the Catalog page by entering the requisite number in the **Selected Products Per Page** field. Valid values here are *5*, *10*, *15*, *20*, and *25*. This setting is available for both, the enterprise CPQ and the new user interface in CPQ.
5. Click **Save**.

When you browse the catalog page as a specific user, you will find the specified number of products on a single page and the rest of the products are divided into multiple pages using Pagination.

If you want to set up the above settings at an org level, click **New** on the Config User Preferences page.



Configuring Custom Attribute Pages

You can now use your predefined custom attribute page while navigating from the Catalog page, Mini Cart, Installed Products (Assets) page, and the Cart page in the AngularJS based CPQ UI framework. By default, the system chooses the *Apttus_Config2__ProductAttributeDetail3* VFpage. This capability allows the customers to have their own UI implementation for the product configuration experience using underlying Conga APIs. This enhancement is useful when you have designed a custom attribute page with the new look and design and want to use it while configuring the attributes.

To use this enhancement, customize your flow settings under the Custom Settings as described below.

1. Go to **All Tabs** > **Config Settings** and click **Flow Settings**.
2. In the **Product Attribute Detail Page** field, enter the name of your custom attribute page.
3. Click **Save**.

Now, when you configure your attributes page by clicking **Configure** from the following pages, you will see your customized attributes page.

- **Configure** button on the Catalog page.
- Wrench icon on the Mini Cart, next to the respective product on the Catalog page.
- **Configure** button on the Installed products (Assets) page.
- Wrench icon for configuration under the Actions column, next to the respective product on the Cart page.

Hiding the Add to Cart and Configure Buttons for Option Products

In order to prevent your sales representative from mistakenly selling the option products without a bundle, you can now hide the **Add to Cart** and **Configure** buttons from the Catalog and Bundle pages. This enhancement is also useful when your customer searches for an option product using the Global search and tries to add it to the cart without buying its associated bundle product. This creates a clear difference between the standalone and option product on the Catalog page.

With this feature, for each option product, the text **Sold as Option** is displayed when you add the option product from the Catalog page without its bundle product.

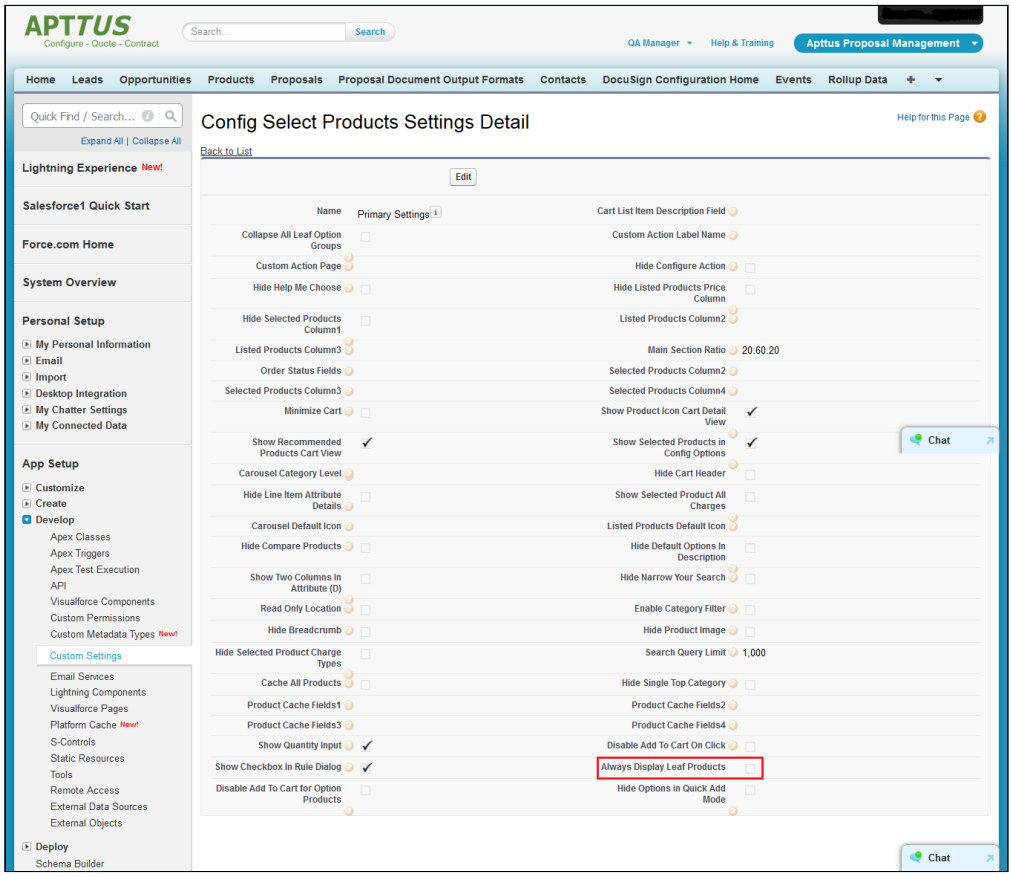
To use the above enhancement, go to **Custom Settings**> **Config Select Products Settings** and select **Disable Add To Cart for Option Products**. For more details, refer to Configuring Custom Settings.

Displaying Leaf Products on the Catalog Page

The new UI supports the custom setting **Always Display Leaf Products** to load and display all the leaf nodes for a specific category. These leaf nodes have products associated with them. With this feature, it is easier for a sales representative to just click the main category or any sub category to load all the sub-categories (and products) at one click. This feature is already available in the Enterprise CPQ.

To configure this setting,

1. Go to **Setup > Develop > Custom Settings > Config Select Product Settings**.
2. Next to **Config Select Product Settings,** click **Manage.**



3. Select the **Always Display Leaf Products** check box.

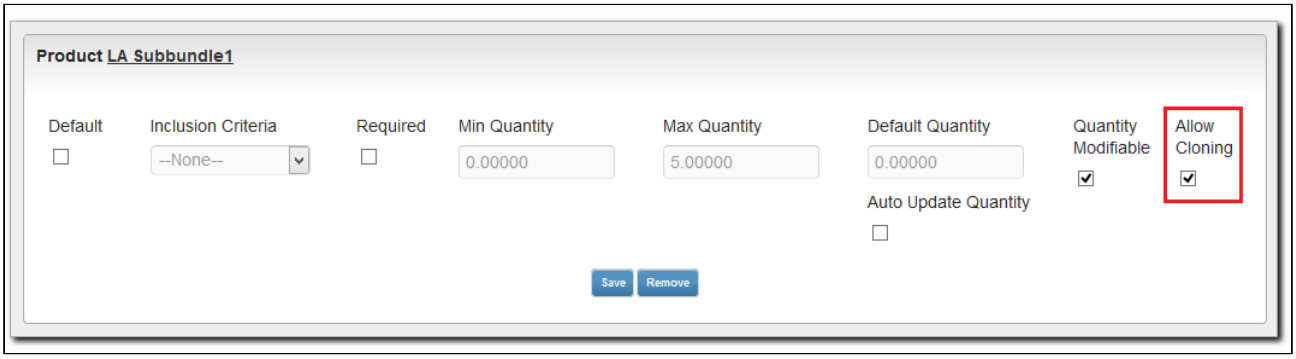
Now, if you click any category on the Catalog page, all its leaf products are displayed by default.

Cloning Sub-Bundles, Options, and Attributes on the Options Page

This feature allows to copy the options and attributes of a bundle product.

If **Allow Cloning** check box is selected for a specific product option or sub-bundle, the system will show clone icon next to the option on the configuration page while configuring the product. When you click this icon, system will clone all associated attributes as is.

This is helpful when you want to replicate the same options and its attributes under a single sub-bundle product. A single icon click makes it easier for the sales representative to clone all of these at once.



Enabling Multiple Adjustments at the Line-Item Level

Sales reps can apply multiple adjustments at the line level either by applying the adjustments in the corresponding columns or clicking the dollar icon ($) next to the line item from the cart page.

You can allow the sales rep to create more than one adjustment only if the Max Adjustment Lines custom setting has a value greater than 1. This setting is available at **All Tabs > Config Settings > System Properties > Pricing Settings > Max Adjustment Lines.**

The following table describes the different values of this setting and its behavior.

Max Adjustment Lines Value	Behavior
Blank field	$ sign does not appear.
1	$ sign appears against the line item. However, you need to use the adjustment type and adjustment amount columns in the line item itself.
>1	$ sign appears against the line item. You can click on the $ sign to open a multiple adjustment popup. You can create as many adjustments as the value defined in the Max Adjustment Lines.

To customize the fields displayed in the multiple adjustment popup, navigate to **Config Settings > Display Column Settings**. Select *Adjustment Line Item* from the **Display Type**, and *ngFlow* from the **Flow** drop down.

You can also display custom fields that you have defined for Adjustment Line Item object on the cart by using the above procedure.

You can enable users to apply an adjustment on the price of a bundle without applying that same adjustment on the options within the bundle that share the same charge type name.

You can enable users to apply an adjustment type on the bundle or total using % Discount or % Markup so that the Net Adjustment % on the Bundle or Total is set to the user requested value. CPQ allocates the adjustment to the group members (options in the case of a bundle and standalone products and options of bundles if the group is a total). The following requirements must be fulfilled to enable this feature:

- Group Adjustment Spread is set to True.
- Max Adjustment Line must be blank or 1.
- Bundle price must be zero.

The sales rep can select an adjustment from the following Adjustment Types and enter a value in Adjustment Amount column on the Cart page:

> Some of the values might not be available by default. You must add the required value to the **Adjustment Type** field in the Line Item object.
> Perform the following steps to add a value to the **Adjustment Type** field in the Line Item object:
>
> 1. Go to **Setup> Create> Object> Line Item> Adjustment Type**.
> 2. Under the section **Values**, click **New**.
> 3. Enter *<Value_Name>* and click **Save**.

Option	Description
% Discount	Indicates that a percentage discount is applied on the line item based on Adjustment Amount.
Discount Amount	Indicates that a flat discount is applied on the line item based on Adjustment Amount.
% Markup	Indicates that a percentage markup is applied on the line item based on Adjustment Amount.
Markup Amount	Indicates that a flat markup is applied on the line item based on Adjustment Amount.

Option	Description
Price Override	Indicates that the Adjustment Amount entered overrides the net price of the product.
Base Price Override	Indicates that the Adjustment Amount overrides the base price of the product and all the other calculations on the product would happen based on the new base price. **Allow Manual Adjustment** must be selected on the product line item, in order to make **Base Price Override** editable.
% Discount Off List	This is similar to % Discount, except that it calculates the discount amount always on the List Price of a Bundle and applies to the Extended Price.
% Uplift	This gets applied every time to products that are added from the Asset.
Price Factor	Indicates that the Adjustment Amount is multiplied with the **Extended Price** to calculate the new **Net Price**.
Base Price Discount	Indicates that the adjustment amount is reduced from the Base Price.
Base Price Markup	Indicates that the adjustment amount is added to the Base Price.
% Discount (Bundle Only)	Indicates that a percentage discount is applied on the bundle without applying that same adjustment on the options within the bundle.
Discount Amount (Bundle Only)	Indicates that a flat discount is applied on the bundle without applying that same adjustment on the options within the bundle.
% Uplift (Bundle Only)	This is applied to the bundle every time a product is added from the Asset.
% Markup (Bundle Only)	Indicates that a percentage markup is applied on the bundle without applying that same adjustment on the options within the bundle.

Option	Description
Markup Amount (Bundle Only)	Indicates that a flat markup is applied on the bundle without applying that same adjustment on the options within the bundle.
Price Override (Bundle Only)	Indicates that the Adjustment Amount entered overrides the net price of the bundle without applying that same adjustment on the options within the bundle.
Price Factor (Bundle Only)	Indicates that the Adjustment Amount is multiplied with the **Extended Price** to calculate the new **Net Price** of the bundle.
% Discount Effective	Indicates that a percentage discount is applied on the bundle or total so that the Net Adjustment % on the Bundle or Total is set to the user requested value. This adjustment type is applicable when the following requirements are fulfilled: • Group Adjustment Spread is set to True. • Max Adjustment Line must be blank or 1. • Bundle price must be zero.
% Markup Effective	Indicates that a percentage markup is applied on the bundle or total so that the Net Adjustment % on the Bundle or Total is set to the user requested value. This adjustment type is applicable when the following requirements are fulfilled: • Group Adjustment Spread is set to True. • Max Adjustment Line must be blank or 1. • Bundle price must be zero.
Price Factor Effective	Indicates that the Adjustment Amount is multiplied with the **Extended Price** to calculate the new **Net Price** of the bundle or total so that the Net Adjustment % on the Bundle or Total is set to the user requested value. This adjustment type is applicable when the following requirements are fulfilled: • Group Adjustment Spread is set to True. • Max Adjustment Line must be blank or 1. • Bundle price must be zero.

Configuring Option Net Adjustment Rollup to Bundle

You must set the Admin setting *APTS_RollUpOptionNetAdjustment* to *True* so that when a Sales Representative applies adjustments on options, CPQ displays the adjustments on options at the bundle and summary totals level.

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record. The New Admin page is displayed.
3. Fill the following details

Field	Value
Name	*APTS_RollUpOptionNetAdjustment*
Value	*true* or *false*
Code	Leave the field blank

4. Click **Save**.

Enabling Price Breakup for Products

The Price Breakup provides additional information if the line item has related pricing or a price matrix with a dimension having cumulative range. A small icon () next to the base price is displayed if the line item uses either of the two pricing features. You can click the icon to view the additional information.

You can add custom fields on the **Price Breakup** object and add the same in **Display Column Settings** for the Price Breakup table so that when sales representatives open the Price Breakdown pop-up on the Cart, they can see columns for the added custom fields with their corresponding values. If a product line is priced using related pricing, when the sale representative clicks the price breakup icon () next to the base price on the cart, the custom field values are displayed on the Price Breakdown pop-up.

For example, a product record has various fields such as Tier 1, Tier 2, and Tier 3. These are the parameters that help the sales representative identify the product's family and its line of business. You can display these fields on the Price Breakdown pop-up so that the sales representative can understand which parameters are contributing to the related price calculation.

Prerequisite

You must have already created a Price Dimension.

To enable Price Breakup for products

1. Go to your Product and click **Product Console.**
2. Click on the PLI record and click **Edit**.
3. Under the **Matrices** tab, click **New.**
4. Select **Matrix Type** as *Dimension* and select your Price Dimension in the **Dimension 1** drop-down list.
5. In the **Dimension 1 Value Type** drop-down, select *Cumulative Range*.
6. Click **Close** to close the Matrices.
7. Click **Save** to save the PLI.

Displaying Attributes and Options Inline on the Configuration page or a Visualforce Page

For the new products (bundle or sub-bundle), you can choose to display the options and attributes as inline or on a new Visualforce page. A new drop-down list, **Config Type** is introduced to indicate this enhancement for options and attributes.

When you associate attributes or options to your product, you can choose to select *Inline* or *New Page*.

- *Inline:* The attributes and options can be configured on the configuration page.
- *New Page:* A wrench icon is displayed near the bundle or sub-bundle product which has to be configured. When you click this icon, a new page is opened for configuration of options and attributes.

By default, *Inline* option is chosen for the new products, for both, options and attributes.

If you do not choose any option from the drop-down list, no attributes and options are loaded for your product.

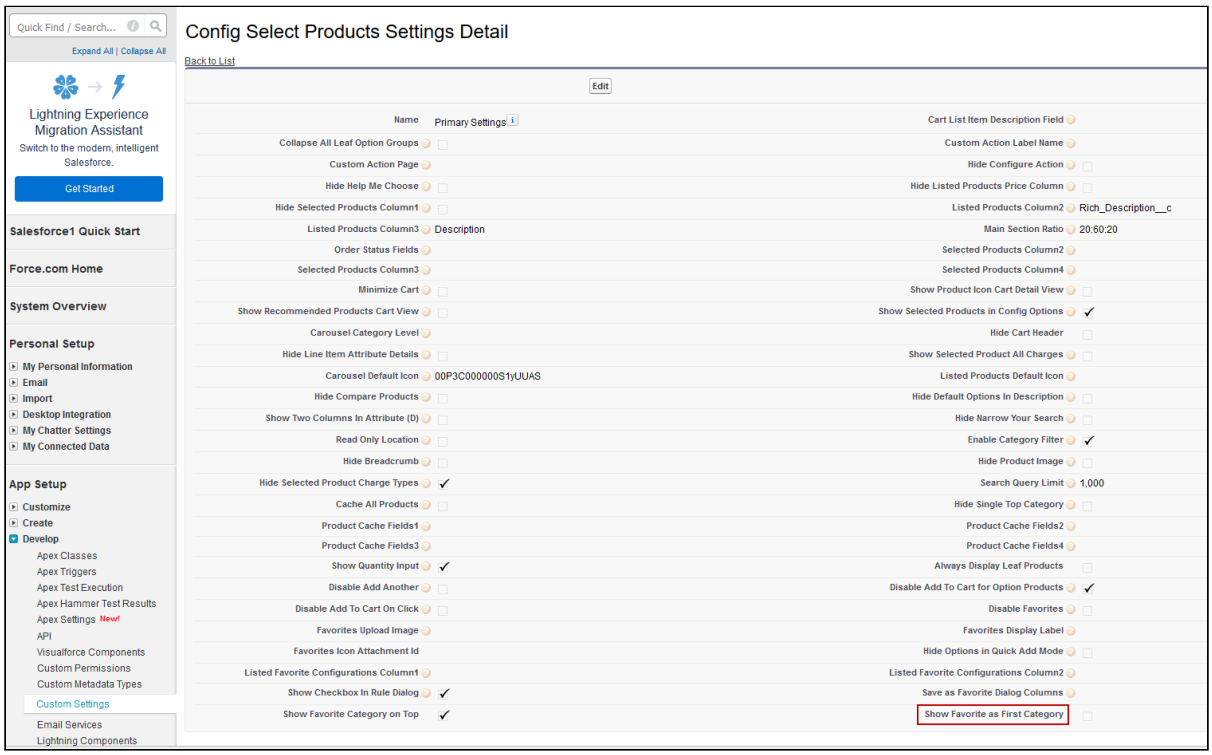
> **This enhancement does not affect your existing product configurations and is applicable for the new products only.**

Sequencing the Favorite Category on the Catalog Page

The new UI supports the custom setting **Show Favorite as First Category** to display your favorite category at the top.

To configure this setting,

1. Go to **Setup > Develop > Custom Settings > Config Select Product Settings**.
2. Next to **Config Select Product Settings,** click **Manage.**



3. Select the **Show Favorite as First Category** check box.

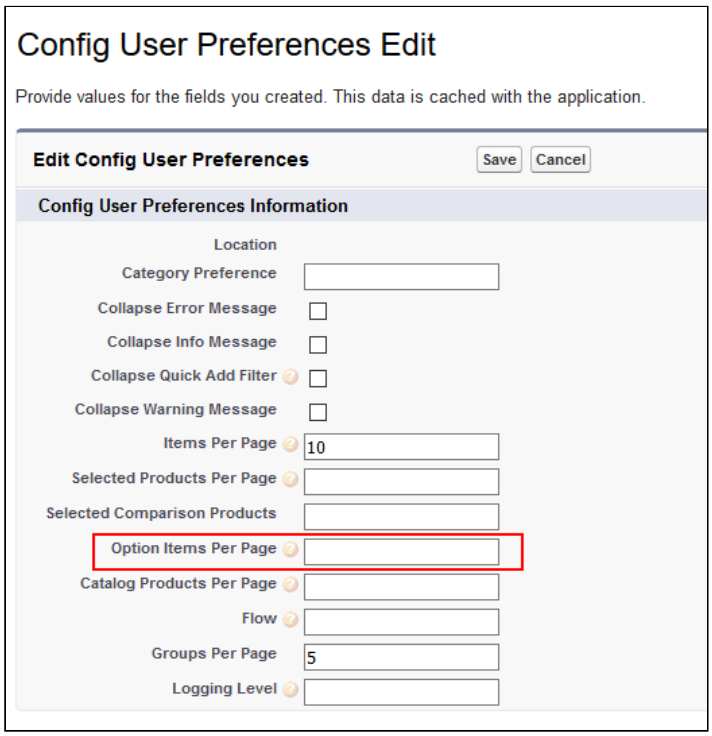
Now, when you have multiple product offerings, you can quickly find your favorite category on the cart. For more information, see About Favorite Configurations on the Cart.

Configuring the Number of Option Products Displayed on the Cart

For the New UI, customize the following custom setting to display the number of option products on the cart page for a specific user. The products are divided across the pages using the Pagination concept.

To customize the user preferences

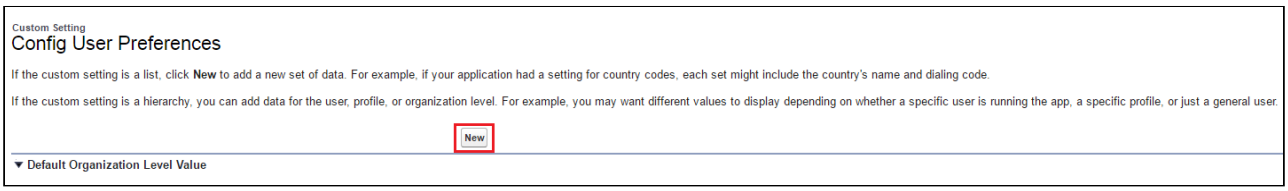
1. Go to **Setup > Develop > Custom Settings** and search for **Config User Preferences.**
2. Click **Manage** next to **Config User Preferences**.
3. For the required user, do any of the following:
   Click **View** to view the details and click **Edit**.
   - Or -
   Click **Edit** to edit and make changes to the existing user preferences.



4. In the **Option Items Per Page** field, enter the number of option products that you want to display on the cart per page. Valid values are *5, 10, 15, 20*, and *25*. This setting is available for both, the enterprise CPQ and the new user interface in CPQ.
5. Click **Save**.

When you browse the catalog page as a specific user, you will find the specified number of option products on a single page and the rest of the products are divided into multiple pages using Pagination.

If you want to set up the above settings at an org level, click **New** on the Config User Preferences page.



Configuring Save as Favorite

The Save as Favorite feature allows the sales rep to save a cart configuration for reuse. The saved configuration can be used multiple times in the same quote or across different quotes. A saved configuration includes all the selected products on the cart with the options and attributes defined within the product. The configuration can be public for all the sales reps or private to the sales rep who saved the configuration.

As an administrator, you can use various functionalities to configure the Save as Favorite feature to control the usage of the feature by the sales reps. Following functionalities are available to you to configure Save as Favorite feature.

To disable the Save as Favorite feature

By default, the feature is enabled. You can disable the feature globally or selectively apply permission to individual user or profile. Disabling the feature only disables the feature icon on the Cart page. To disable Favorite feature globally follow the steps below.

1. Go to **Setup** > **App Setup** > **Develop** > **Events** > **Custom Settings**.
2. Find **Config Select Product Setting** and click **Manage** next to the setting.
3. Click **Edit** next to **Primary Settings**.
4. Select **Disable Favorite**.
5. Click **Save**.

As a result of following the steps above the Save as Favorite icon( ) on the Cart page is disabled.

You can also control which user or profile can use Favorite Feature by defining access permissions for a user or profile.

To define permissions to use Save as Favorite feature

The administrator must define proper access permissions for user profiles to use Save as Favorite Feature. Otherwise, the user cannot see the Save as Favorite feature on the Cart page. Follow the steps below to define access permissions for the user.

1. Go to **Setup> Administrator Setup> Manage Users> Profiles.**
2. Find the profile for which you want to set permissions and click **Edit**.
3. Find **Favorite Configuration** under the section **Custom Object Permissions**.
4. Select one or more permission from the list below to define access for the profiles.

Name	Impact on the Cart Page
Read	A disabled **Save As Favorite** button or a star icon () is displayed on the Cart page.
Create	The **Save As Favorite** button is enabled on the Cart page.
Edit	The Sales rep can view and import Favorites configuration saved by others on the Catalog page when you define Edit permission
Delete	**Delete** option is enabled for Favorites on the Catalog page.

   If you do not select any permission, the Favorite feature is disabled on the Cart page.
5. To enable **Publish** button on the Cart page, find **Published Favorite** under **Custom Object Permissions** and select **Create**.
6. Click **Save**.

Based on the permissions you define here, the sales rep can use different functionality available to them.

To hide scope field in Save as Favorite pop-up on the Cart

Follow the steps below to hide the Scope field.

1. Go to **Setup**. From the **Manage Users** drop-down, select **Profiles**.
2. Select a custom Profile.

   > ⓘ You can only edit these permissions for a custom profile.

3. Under the **Field-level Security** section find **Favorite Configuration** and click **View**.
4. Click **Edit**
5. To hide the Scope field clear the checkboxes **Read Access** and **Edit Access**.
6. To make the Scope field visible to the profile select the checkboxes **Read Access** and **Edit Access**.
7. Click **Save**.

The above action hides or displays the **Scope** field for a specific profile. When the **Scope** field is visible, the users can save a configuration record as **Public** or **Private** by selecting from the picklist. Otherwise, the configuration record is saved as **Private**.

Categorization of Favorite Configuration

CPQ provides sales rep the ability to categorize the saved configuration using pre-defined filters. These fields are displayed on Save as Favorite pop-up on the Cart page while saving the configuration. The fields in the filter are of picklist and multi-select picklist type that you must create in the Favorite Configuration object to use as filters. Any existing picklist or multi-select picklist fields can also be used, except for **Scope**. If you define default values or make the fields required, they are honored on the pop-up as well. Only the first five fields that you define are displayed on the pop-up.

To define filters to save a configuration

Follow the steps below after creating picklist or multi-select picklist type fields in Favorite Configuration object.

1. Go to **Setup** > **App Setup** > **Develop** > **Events** > **Custom Settings**.
2. Click **Manage** next to **Config System Properties** setting.
3. Click **Edit** next to **System Properties**.
4. Find **FavoriteFilters** and enter the API names of picklists you created, separated by comma or a new line.
5. Click **Save**.

Configuring the Smart Cart

The Smart Cart feature allows the sales rep to manage the cart with a large number of line items. In Smart Cart flow, line items in the cart are divided into groups for pricing based on the threshold and split criteria you define. The threshold defines the number of line items in a group. The split criteria list the fields based on which the line items are grouped. You can define different threshold and criteria for different flow. You must configure the Smart Cart feature for the sales rep to use.

When you configure the product in the Smart Cart flow, you must take the following concepts into consideration.

- The option line items along with bundle line item are included in the threshold count. For example, if the threshold is 10, then in a bundle you can only select 9 options as that occupies an entire group. So, when the groups are divided on pricing, if your bundle and option line items exceed the threshold, pricing is not calculated. However, if the bundle doesn't fit in a half-full group a new group is created for that bundle as long as the line items are within the threshold limit.
- A bundle is not divided across multiple carts. If the options are auto-included or added by default and the split cart threshold is exceeded, the split function is not executed.
- Line items with multiple charge lines are divided across multiple carts if required.

Follow the steps below to configure the feature.

To add QTC profile field on Quote Detail page layout

You must add the field **QTC Profile** to the Quote Detail page layout. This field is used by the sales rep to enable the Smart Cart flow for their quote.

To define the threshold for the Smart Cart

The threshold defines the size of the group that the line items are divided into for pricing. Only the primary line items are considered in the threshold to split the cart. The primary line items consist of the line items for standalone products, bundle products and option products.

You must define the threshold considering the pricing complexity and the number of line items your pricing engine can process. If the threshold exceeds the limit of the number of line items processed in the price engine, Smart Cart does not split the cart properly and the price is not calculated. However, if you define a low threshold when you have a relatively large number of line items, Smart Cart creates a large number of groups, and pricing is not calculated properly.

Also, the bundles are not divided across multiple carts. If the number of bundles and option line items is greater than the threshold, the split function is not executed. You can increase the threshold keeping the pricing engine in consideration or restructure the bundle.

Split Cart Threshold is the primary field that Smart Cart uses to split the line items in the cart. Even if you don't define **Split Cart Criteria Fields**, the line items are grouped based on just the threshold.

Perform the following steps to set the threshold.

1. Go to **Setup > Develop > Custom Settings > Config System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. Enter a number in the **Split Cart Threshold** field.
4. Click **Save**.

To define the split criteria for the Smart Cart

The split criteria define the criteria based on which the line items are grouped. Defining **Split Cart Criteria Fields** is optional as Smart Cart can split the cart based on just the threshold. If you define the **Split Cart Criteria Fields**, then Smart Cart considers the criteria first then the threshold. Hence, when the line items are divided, the groups are created based on the criteria first and then the number of line items in a group is controlled by Smart Cart.

Perform the following steps to set the criteria.

1. Go to **Setup > Develop > Custom Settings > Config System Properties > Manage**.
2. Find **System Properties** and click **Edit**.
3. in the **Split Cart Criteria Fields** field, enter the API names of the fields from the Line Item object, separated by commas. Only the fields with the following data types are supported as criteria:
    - Text
    - Picklist
    - Number
    - Formula fields evaluating to Text output
    - Relationship fields.

    > ⓘ Other data types such as Boolean, Text Area, and Rollups are not supported.
4. Click **Save**.

> If the fields you entered in **Split Cart Criteria Fields** are uncommon fields that are not used in any prominent custom settings or display columns, execute *Refresh Field Metadata* from Maintenance Console to update the data cache.

To define line item processing batch size

Smart Cart uses **Split Cart Threshold** to split the line items into groups to calculate pricing. This threshold is defined based on the pricing complexity and the number of line items your

pricing engine can process. However, the threshold can exceed the governer heap limit if the size is large for a single pricing call. You can use the admin setting **APTS_CartLineItemsUpdateChunkSize** to define the number of line items in a batch to be processed in a single pricing call. You can use this admin setting to avoid hitting heap size errors.

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record. The New Admin page is displayed.
3. Fill in the following details

Field	Value
Name	*APTS_CartLineItemsUpdateChunkSize*
Value	Enter the number of line items to be processed in a batch for pricing.
Code	Leave the field blank

4. Click **Save**.

To enable actions on the cart page

In the Smart Cart flow, the action buttons are different than the actions available in a regular flow. You must enable the action buttons to display them in the action menu of the cart for Smart Cart flow. Follow the steps below.

1. Click the All Tabs icon ().
2. Go to **Config Settings**>**Display Action Settings.**
3. Select *Cart Page* in **Display Type** and the flow that you want to use in Smart Cart flow in **Flow**.
4. Find and select the checkbox **is Enabled** for the following actions.
    - Submit for Pricing (Async)
    - Pre-Price
    - Post-Price
5. Click **Save**.

Use Case for Configuring the Smart Cart

Description: This use case describes how the line items are grouped in Smart Cart.

Suppose you are a CPQ administrator of a company selling health care equipment to hospitals in bulk. You sell to hospitals with branches in different locations. Selling to such hospitals in bulk results in quotes with a large number of line items and configuration based on different criteria such as Location. The quotes also include the pricing details including the adjustments applied on the line items. Your Sales Rep regularly deals with such a large number of line items in a single quote. A large number of line items are difficult to handle and navigate through.

Using the Smart Cart feature, the pricing of the line items calculated in chunks based on a threshold and criteria. The cart is divided into Config and Pricing configurations based on the threshold and criteria you define.

For example, if you want to divide the line items into groups of 500 line items and create the groups based on the location of different branches of the hospital. You must define the **Split Cart Threshold** as 500 and **Split Cart Criteria Fields** as the API name of the location field.

Result

If the total number of line items the Sales Rep adds to the cart is 2000 and clicks **Submit for Pricing(Async)**. The line items are divided into groups on the quote page represented by the configurations. Consider the below screenshot as the sample of the products added to the cart with location.

Location	Product
Loc1	Orbital Sandler
Loc2	Orbital Sandler
Loc1	Orbital Sandler
Loc4	Orbital Sandler
Loc5	Orbital Sandler
Loc6	Orbital Sandler
Loc7	Orbital Sandler
Loc8	Orbital Sandler
Loc9	Orbital Sandler
Loc10	Orbital Sandler
Loc1	Orbital Sandler
Loc2	Orbital Sandler
Loc1	Orbital Sandler
Loc4	Orbital Sandler
Loc5	Orbital Sandler
Loc6	Orbital Sandler
Loc7	Orbital Sandler
Loc8	Orbital Sandler
Loc9	Orbital Sandler
Loc10	Orbital Sandler
Loc1	Orbital Sandler
Loc2	Orbital Sandler
Loc1	Orbital Sandler
Loc4	Orbital Sandler

Based on the values of **Split Cart Threshold** and **Split Cart Criteria Fields**, the Child Carts are created as depicted in the table below, while the number in the parenthesis denotes the number of line items in the Child Cart.

Pricing Configuration	Line Item Division
1	Location 1 (200)
2	Location 2 (300)
3	Location 3 (500)
4	Location 4 (10)
5	Location 5 - 1 (500)
6	Location 5 - 2 (490)

Configuring Product Sort

Product Sorting enables the Sales rep to sort products on the Catalog page. The Catalog page is by default sorted in ascending order of Product Name, the Sales rep can change that in the **Sort** drop-down. The Sales rep can select to sort the product in either ascending or descending order of the field. In **Sort** drop-down on the Catalog page, two options are provided by default:

- *Product Name Ascending*
- *Product Name Descending*

You can add two more fields in the drop-down by defining the fields in **Product Sorting Fields** in Config System Properties. Each field you define in **Product Sorting Fields** has two entries in the **Sort** drop-down on the Catalog page, one for ascending order and other for descending order of the products.

Defining Product Sorting Fields

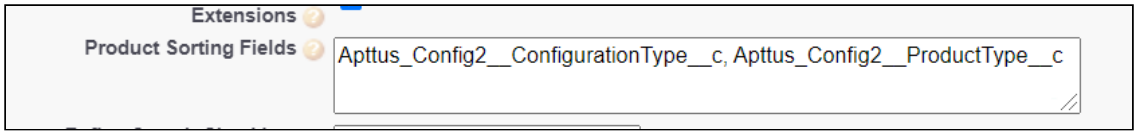
You can define a maximum of two additional fields in **Product Sorting Fields** to be displayed in the **Sort** drop-down apart from the field **Product Name. Product Name** is

added by default. If you add more than two API names in **Product Sorting Fields** then only the first two values are displayed in the **Sort** drop-down.

To Define Product Sorting Fields

1. Go to **Setup** > **Develop** > **Custom Settings**.
2. Click **Manage** next to **Config System Properties**.
3. Click **Edit** next to **System Properties.**
4. In **Product Sorting Fields**, enter the API names of the fields from Product object to display the **Sort** drop-down. You can define a maximum of two fields at a time. You can add fields only from the Product object with the following data types:
    - Text
    - Number
    - Checkbox
    - Date
    - Time
    - Date/Time
    - Picklist
    - Currency

    An example of how the API name fields are added to the **Product Sorting Fields** is shown below:

5. Click **Save**.

Configuring Cart Views

This section provides information on configuring cart views.

- About Cart Views
- Enabling Cart View Creation in CPQ Admin Console
- Running the Migration Script for Cart Views

About Cart Views

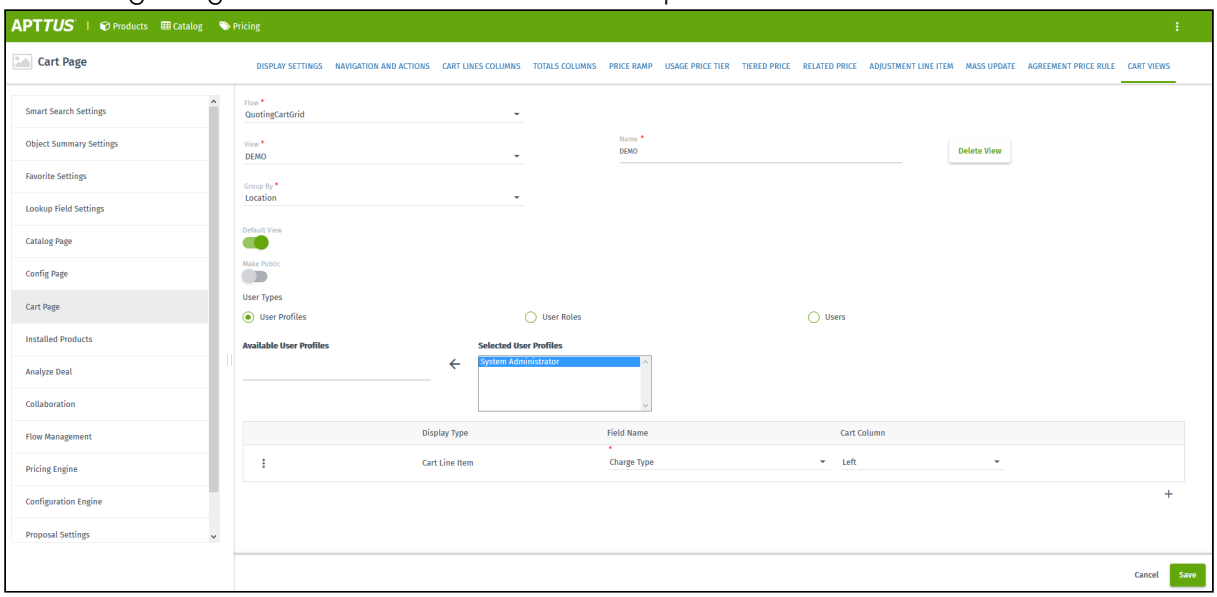
CPQ provides admin cart views.

You can set a cart view as default or admin for a group of users in order to provide them a tailor-made experience per business requirements.

- You can set an exisiting or a new cart as default or admin.
- You can create a user group based on profiles, roles, and users to assign the default view.
- You can disassociate the user groups from the default view.
- You can format the admin view layout.

The cart views uses following two objects for storing the information related to views instead of only ConfigSetting object.

- Config Field Set object
  This object is used to store the field list for a particular Cart View that is created by a user.
  Go to **Custom Setting** > **Config Field Set** to see the view that a user has created from the Cart. The cart views created on the cart by the users are *Private* and are not visible to other users.
- User Views object
  This object is used to store the association between the Cart Views and the User Type (User Profiles or User Roles or Users). It also stores information such as Flow, Default setting, Group By field, and Type of profile.
  Go to **All Tabs** > **User Views** to see the view that an admin has created from the New Admin UI (CPQ Admin) > **Application Management** > **Cart Page**.
  On the New Admin UI, an admin can configure a view with a Flow, View Name, Group By field, default setting, Public or Private profile type, and the User Type. Ususally the view created by an admin is *Protected* by default and is visible to other users based on the eligibility. The admin can make the view public to make it visible to all users.



A user can delete a view which is created only by the user (Private view) while an admin can delete a view which is created only by admin (Protected and Public views).
If a default view is deleted, then the next default view in the following order of precedence is displayed:

- User view = Private
- Admin views = Profile, Role, User, Public

If a user does not set the default view, then the first view that is associated with the user profile is displayed as default.

Enabling Cart View Creation in CPQ Admin Console

You need to configure the settings for creating a cart view in the New Admin Console of CPQ.

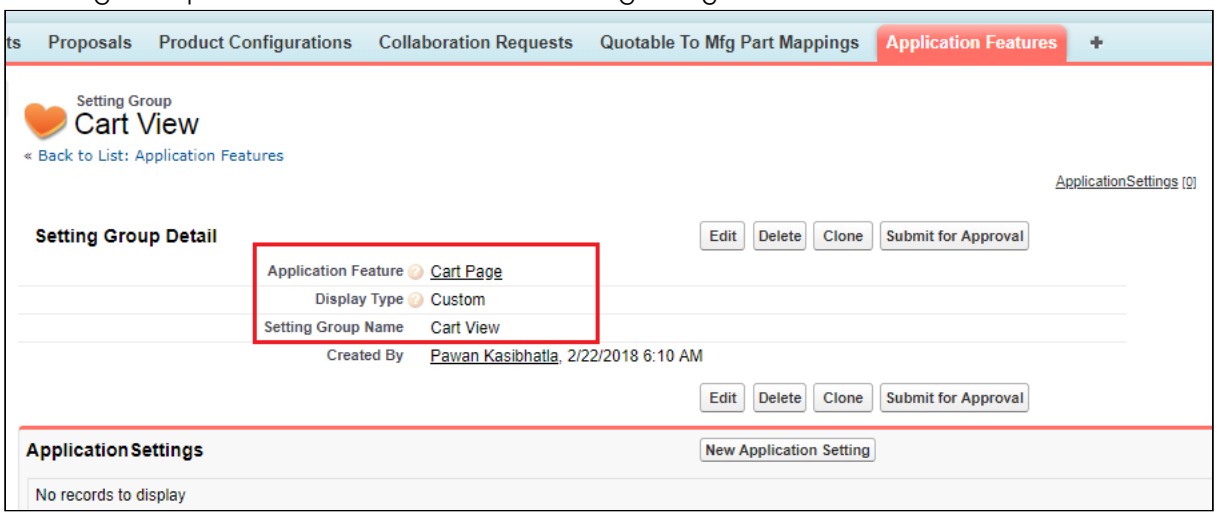
To create a Cart View in New Admin Console

1. Go to **Custom Object Setting Group** and click **Display Type** picklist field.
2. Add the picklist value **Custom**.
3. Create a tab for **Application Features** object.
4. Go to **Application Features** tab and then click the **Cart Page** record.

   If you do not see any records under **Application Features** tab, then you need to click **Load Defaults** on CPQ Admin by navigating to **All Tabs > CPQ Admin > Application Management > Load Defaults**.

5. Edit the layout of the "Cart Page" record and add "Setting Groups" related list to it.
6. Click **New Setting Group** in the "Setting Groups" related list on the "Cart Page" layout. Fill the below details in the new setting group record for "Cart View" and save it.

7. After you create the "Cart View" setting group record, it will be available in the "Setting Groups" related list on the "Cart Page" layout as shown below:



Cart View page gets available on the CPQ Admin console. You can access this page from **All Tabs > CPQ Admin > Application Management > Cart Page > Cart Views**.

> To create view for a flow, first you need to configure the flow for Cart Grid by adding "#cartgrid" after the flow name.
> Only those flows which are configured for Cart Grid are available in the "Flow" field on the "Cart View" page.

Running the Migration Script for Cart Views

As an existing customer, if you are upgrading to the current release from a release version older than Summer 2018, you must run a post-installation script manually through the developer console to update your existing Cart Views to the new framework.

> For customers setting up org on release version later than Summer 2018, the post-install script runs automatically after you install the CPQ package. In such a case, you do not need to execute the migration script manually.

Migration Script

Apex Code: Create a new class in the org with the code given below.

```
public class APTS_CreateCartViews{
```

```
/**
 * Checks if the given string value is null or empty.
 * @param strValue the string to check
 * @return <code>true</code> if the string value is null or empty, <code>false</code>
otherwise
 */
public static Boolean nullOrEmpty(String strValue) {
    // check if null or zero length string
    return (strValue == null || strValue.trim().length() == 0);
}

/**
 * Checks if the given list of objects is null or empty.
 * @param objList the list of objects to check
 * @return <code>true</code> if the list is null or empty, <code>false</code>
otherwise
 */
public static Boolean nullOrEmpty(List<Object> objList) {
    // check if null or empty
    return (objList == null || objList.isEmpty());
}

/**
 * returns comma separated value
 */
public static String arrayToCSV(String[] arr){
    String str = '';
    Boolean isFirst = true;
    for(String ele : arr){
        if(isFirst){ isFirst = false; }else{
            str += ', '; //adding space since to display rule action info properly in
the layout
        }
        str += ele;
    }
    return str;
}

public static void saveFieldList(String dataStore, List<String> fieldNames) {

    String names = arrayToCSV(fieldNames);
    Integer len = names.length();
    Integer recordCount = Integer.valueOf(Math.ceil(len * 1.0 / 255));
```

```
    //delete existing records
    Apttus_Config2__ConfigFieldSet__c dataCache =
Apttus_Config2__ConfigFieldSet__c.getInstance(dataStore + '_' + 0 );
    if (dataCache != null) {
        List<Apttus_Config2__ConfigFieldSet__c> queryFieldList = new
 List<Apttus_Config2__ConfigFieldSet__c>();
        Integer index = 1;
        while(dataCache != null){
            queryFieldList.add(dataCache);
            dataCache = Apttus_Config2__ConfigFieldSet__c.getInstance(dataStore + '_'
 + index );
            index += 1;

        }
        delete queryFieldList;

    }

    List<Apttus_Config2__ConfigFieldSet__c> queryFieldList = new
 List<Apttus_Config2__ConfigFieldSet__c>();

    for (Integer i = 0; i < recordCount; i++) {
        Integer startIndex = i * 255;
        Integer endIndex = (i + 1) * 255;
        Integer isLast = 0;
        if (endIndex >= len) {
            endIndex = len;
            isLast = 1;
        }
        queryFieldList.add(new Apttus_Config2__ConfigFieldSet__c(Name = dataStore +
'_' + i, Apttus_Config2__Data__c = names.substring(startIndex, endIndex),
Apttus_Config2__IsLast__c = isLast));

    }

    insert queryFieldList;
}

public class MigrateCartView {
    public String GroupByField;
    public String viewName;
    public List<columnDO> Columns;
    public Boolean isDefault = false;
```

```
}

//Data structure used to hold the deserialized JSON of cartviewsetting.
public class columnDO {
    public String FieldName;
    public Integer Sequence;
    public String Pinned;
}

//this method deserialize the cartview setting related data stored in JSON format.
private static List<MigrateCartView> parse(String json) {
    return (List<MigrateCartView>) System.JSON.deserialize(json,
List<MigrateCartView>.class);
}

/**
 * Create the new Cart Views Setting Group for Application Management feature of New
Admin
 */
public static void createCartViewsSettingGroup() {
    List<Apttus_Config2__ApplicationFeature__c> cartPageAppFeatures = [SELECT
Id,Name,Apttus_Config2__ApplicationName__c
                                                    FROM
Apttus_Config2__ApplicationFeature__c
                                                    WHERE
Apttus_Config2__ApplicationName__c = 'CPQ'
                                                    AND Name = 'Cart Page'
                                                    LIMIT 1];

    List<Apttus_Config2__SettingGroup__c> cartViewsSettingGroups = [SELECT Id,Name
                                                FROM Apttus_Config2__SettingGroup__c
                                                WHERE Name = 'Cart Views'
                                                LIMIT 1];

    // Create the new Cart Views Setting Group only if the Cart Page Application
Feature exists and the Cart View Setting Group DOES NOT exist
    if (cartPageAppFeatures != null
        && cartPageAppFeatures[0] != null
        && nullOrEmpty(cartViewsSettingGroups)
        ) {

        Apttus_Config2__SettingGroup__c newCartViewsSettingGroup = new
 Apttus_Config2__SettingGroup__c();
        newCartViewsSettingGroup.Name = 'Cart Views';
```

```
        newCartViewsSettingGroup.Apttus_Config2__ApplicationFeatureId__c =
cartPageAppFeatures[0].Id;
        newCartViewsSettingGroup.Apttus_Config2__IsCustom__c = false;
        newCartViewsSettingGroup.Apttus_Config2__Sequence__c = 12;
        newCartViewsSettingGroup.Apttus_Config2__DisplayType__c = 'Custom';
        newCartViewsSettingGroup.Apttus_Config2__ConfigFlow__c = 'Default';

        insert newCartViewsSettingGroup;
    }
}

/**
 * Post installation script to migrate cartview setting related data into new object
model.
 */
public static void migrateCartViewSetting() {
    String nsPrefix = 'Apttus_Config2__';
    List<Apttus_Config2__UserView__c> newCartViewSettings = new
 List<Apttus_Config2__UserView__c>();
    Integer isProductIdCount = 0;
    List<String> allColumns;
    List<MigrateCartView> allOldData;
    String pinned;
    String viewName;

    //
    List<Apttus_Config2__ConfigSettings__c> configSOs = [SELECT Name,
                                                Apttus_Config2__Scope__c,
                                                Apttus_Config2__ViewSettings__c,
                                                Apttus_Config2__Flow__c
                                        FROM Apttus_Config2__ConfigSettings__c
                                        WHERE Apttus_Config2__Scope__c = 'User'
                                        LIMIT 100];
    for (Apttus_Config2__ConfigSettings__c configSO : configSOs) {
        isProductIdCount =0;

        // Deserialize the JSON text stored in viewSettings__c into MigrateCartView
data structure.
        allOldData = parse(configSO.Apttus_Config2__ViewSettings__c);

        // One record can hold setting data for more than one view. Loop thru all
views.
        for (MigrateCartView oldView : allOldData) {
```

```
            allColumns = new List<String>();

            // Current design allows max 36 chars for view name. Remaining chars to
be truncated below.
            viewName = oldView.viewName.length() > 36 ? oldView.viewName.left(36) :
oldView.viewName;
            newCartViewSettings.add(new Apttus_Config2__UserView__c (Name = viewName,
                                                    Apttus_Config2__Type__c =
'Public',
                                                    Apttus_Config2__Flow__c =
configSO.Apttus_Config2__Flow__c,
                                                    Apttus_Config2__GroupBy__c =
oldView.GroupByField,
                                                    Apttus_Config2__IsDefaultView__c
= oldView.isDefault,

Apttus_Config2__IsDefaultRecord__c = oldView.isDefault));

                // Loop through every column settings
                for(ColumnDO ColumnD : oldView.Columns) {

                    //Column pinned value left null to be changed to middle
                    if (ColumnD.Pinned == 'left') {
                        pinned = 'Left';
                    } else     if (ColumnD.Pinned == 'right') {
                        pinned = 'Right';
                    } else {
                        pinned = 'Middle';
                    }

                    allColumns.add(ColumnD.FieldName + '|' + ColumnD.Sequence + '|' +
pinned);

                    // Check whether the current setup contains ProductID field or
not
                    if (ColumnD.FieldName == nsPrefix + 'ProductId__r') {
                        isProductIdCount++;
                    }

                }

                // Add productID if the view doesn't contain
                if (isProductIdCount == 0) {
                    allColumns.add(nsPrefix + 'ProductId__r|1|Left');
```

```
                }

                // Create customsetting data for column values.
                saveFieldList(viewName, allColumns);
            }

            // Flag the processed data and not to migrate in next run.
            configSO.Apttus_Config2__Scope__c = 'Admin';
        }
        if (!nullOrEmpty(configSOs)) {
            update configSOs;
        }
        if (!nullOrEmpty(newCartViewSettings)) {
            insert newCartViewSettings;
        }
    }

    }
```

Run the following script manually using the developer console:

Run this script only after the abovementioned apex class is created successfully in the org.

```
Savepoint sp = null;
try {
    sp = Database.setSavepoint();
    // Method to create the new Setting Group in New Admin, since Load Defaults will
override existing user customizations
    APTS_CreateCartViews.createCartViewsSettingGroup();

    // Method to migrate old Cart Views to the New Cart Views data model
    APTS_CreateCartViews.migrateCartViewSetting();

} catch (Exception ex) {
    Database.rollback(sp);
    System.debug(ex.getMessage());
}
```

Significance of script

In Summer 2018 Release, Cart Views functionality has been enhanced to include Cart Views Admin setup. For more information, see About Cart Views.

The data model for Cart Views has been updated in order to provide better user experience and enable more features.

You have to migrate to new Cart Views for maintaining data that you created before Summer 2018 Release.

Mechanism of script

The script queries all the Cart Views that were created prior to upgrading to Summer 2018 Release and copies the related data over to the new Data Model.

It checks for Unique Name for the migration to ensure that the same Cart View is not migrated more than once.

It creates the new tab called **Cart Views** under **Cart Page** in New Admin UI. For more information, see Application Management Settings

> ⓘ Old Cart Views were created by individual users and were accessible to all users. Hence after migration such old Cart Views will be marked as Public, so that all users can access them. However, due to the nature of the new enhanced Cart Views, only admin users can edit the Public Cart Views.

Configuring Cart Activity History

Cart Activity History records the updates you make to product configuration. CPQ adds an entry to the **Activity History** when you create, finalize, and reconfigure the product configuration. It helps you track the changes made to the product configuration. You can choose to disable the generation of such history, if not needed. You can use the Admin Setting *APTS_DisableCartActivityHistory* to disable the creation of entries in the **Activity History**. If you create and enable this setting, CPQ does not add entries when you create, finalize, or reconfigure a product configuration.

> ⓘ This setting does not impact the Activity History of Proposal and Agreement object.

To disable cart activity history

You must follow the steps below to configure this setting:

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record. The New Admin page is displayed.
3. Fill the following details

Field	Value
Name	*APTS_DisableCartActivityHistory*
Value	*true* or *false*
Code	Leave the field blank

4. Click **Save**.

Configuring CSS Override

CPQ enables you to implement CSS adjustments on different CPQ pages based on your organization's needs using the **CSS Override** field in Config System Properties. You can use CSS Override to add custom styles to pages like Catalog, Configuration, and Cart, by overriding the default style. For example, you can add logos, images on CPQ pages. You can define CSS Override to specific flows or globally to all flows. When you configure CSS override for a specific flow, the CSS is applied only on that flow. However, when you apply CSS override globally, CPQ also overrides flow-specific CSS adjustments along with CPQ default style. For more information, see About Custom Setting Order of Precedence.

Pre-requisites

Follow the steps below before you configure CSS Override.

1. Create a CSS file with the adjustments you want to implement on CPQ pages.
2. Upload the file as a static resource in your org. For more information on static resources, see Using Static Resources in Salesforce.

To configure CSS Override globally

1. Go to **Setup > Develop > Custom Settings > Config System Properties > Manage**.
2. Find **System Properties** and click **Edit**.

3. Enter the name of the CSS file that you uploaded as static resources in the **CSS Override** field. This enables you to use the styles defined in the CSS file as opposed to default styles.
4. Click **Save**.

To configure CSS Override specifically for a flow

1. Create a new flow-specific Config System Properties data set or open an existing flow-specific data set. For more information, see Configuring Custom Settings for Different Flows.
2. Enter the name of the CSS file that you uploaded as static resources in the **CSS Override** field. This enables you to use the styles defined in the CSS file as opposed to default styles.
3. Click **Save**.

Example

You can use the **CSS Override** field to hide the Line Item Actions such as Price Ramp, available in the menu ( ) next to each line item on the Cart page. You can define such CSS Override to hide line item action globally or based on a flow. The sample code below is an example of CSS code that you can use to hide the line item menu actions from the Cart page.

```
/*To hide price ramp*/
#lla-price-ramp {
    display: none;
}

/*To hide related-lines*/
#lla-related-lines {
    display: none;
}

/*To hide multiple adjustments*/
#lla-adjustments {
    display: none;
}

/*To hide usage tier*/
#lla-usage-tier {
    display: none;
```

```
}

/*To hide incentives*/
#lla-incentives {
    display: none;
}

/*To hide water fall*/
#lla-price-waterfall {
    display: none;
}
```

Configuring Mass Option Selection on the Configuration Page

The Mass Option Selection feature allows the Sales rep to select, clear the selection, and update options and attributes in one go before processing pricing and expression and invoking rules. After configuring all options and attributes, the Sales rep must click the **Confirm Option Selection** to initiate the process of pricing and executing rules. You can enable this feature by selecting the **Check Many Options** Custom Setting. When you enable this feature, **Confirm Option Selections** button is enabled on the Configuration and Service Bundle Configuration page. If there are any errors encountered during the process, CPQ displays them after completion of the process.

By default, CPQ calculates pricing, processes relevant expression, and executes associated rules each time you select, clear, or update an option or attribute.

To enable the Mass Option Selection feature

Follow the steps below:

1. Go to **Setup** > **App Setup** > **Develop** > **Custom Settings**.
2. Find **Config System Properties** and click **Manage** next to the setting.
3. Click **Edit** next to **System Properties**.
4. Select **Check Many Options**.
5. Click **Save**.

You can also define **Check Many Options** for specific flows. For more information, see Configuring Custom Settings for Different Flows.

To add the Confirm Option Selection button on the Configuration page

Follow the steps mentioned in the "*To configure action buttons on the Summary panel on the Configuration page*" section in Configuring Display Actions Settings to add the **Confirm Option Selections** button on the Configuration page.

Configuring Express Proposal

Express Proposal allows the Sales rep to finalize the cart, select a template and generate the quote, email the document to the customers, and mark the quote as presented, in one place. The feature is invoked upon the click of the **Express Quote** button on the Catalog or Cart page, which must be added by configuring Display Action Settings in Config Settings. CPQ executes the following tasks when the Sales rep clicks the **Express Quote** button:

- Finalize the proposal in asynchronous mode.
- Generate the proposal in the desired template in PDF format.
- Email the proposal to the Quote Owner.
- Mark the proposal as presented.

You can configure the feature in either:

- Single-click mode: to execute the abovementioned tasks automatically or
- Prompt mode: to prompt the Sales rep to modify the task execution on a popup.

You can use the **showDetails** parameter in the **Action Page** column to drive the mode in which feature is executed. CPQ executes the feature differently for a different value of the **showDetails** parameter, which is described in the table below:

Value of **showDetails** parameters	Mode	Description
false or blank	Single-click	CPQ invokes the Express Quote feature in single-click mode where the tasks are executed automatically without displaying the popup on the page where the Sales rep clicks the **Express Quote** button. In this case, CPQ uses the default template from **Default Template Name** in Proposal System Properties, or a template filtered by query template filters to generate the document in PDF format and sends the email to the quote owner. ⓘ If no default template is defined or the query template filters returns multiple templates in result, CPQ displays the **Express Quote** popup.
true	Prompt	CPQ invokes the Express Quote feature in Prompt mode and displays the **Express Quote** popup on the page where the Sales rep can select document generation templates, add email recipients, and mark the proposal as presented.

For more information on where to add the **Express Quote** button and define the **showDetails** parameter, see Configuring Display Actions Settings.

For more information on how to use the express proposal, see Using Express Proposal.

Managing Translation of Fields and Values in Different Language

You can use this feature to display translated field labels when the user belongs to a different locale. Multi-Language support is available for out-of-the-box field names, picklist values, button labels, custom labels, and error messages.

You must follow different sets of instructions to translate Salesforce standard fields and Conga CPQ field values.

This section describes the following topics:

Translating CPQ Record Values

Conga CPQ provides the ability to translate the records to other languages. You can use this functionality to display translated values of the following type of records:

- Product
- Category and Option Group

For Sales reps using the org in a locale other than English, you can translate the values as necessary and display them on the Catalog, Configuration, and Cart pages. To translate the values you must add the translated text in the corresponding translation-related list for each record. For example, you must add translated text of the product field details in the **Product Translation** related list for each product.

> If you want to translate the custom fields you have created on the Product or Category object, you must create custom fields with the same API names on the Product Translation or Category Translation object as well. After you create the fields, you must add those fields in the Product Translation or the Category Translation edit pages layout.

Translating Product Details field value

You must translate the information in the product details field values in the desired language and add them to the Product Translation related list for each product.

To add translated information in the Product Translation related list

1. In Salesforce Classic UI, go to App Menu > **Apttus Products Setup** > **Products**.
2. Open the product for which you want to add translated information.
3. Under **Product Translations** related list, click **New Product Translation**
4. Fill in the following details.

Field	Description
Translation Name	Enter the name of the language.
Currency	Select the currency in which the prices are displayed.
ProductCode	Enter the translated Product Code.
Description	Enter the translated product description.
Product	Product name is populated automatically if you navigated to this page through the Product Details page. Otherwise, search and select the name of the product.
Family	Enter the translated Product Family
Language	Enter the 2 letter ISO code for language. For example, *FR* for French and *JP* for Japanese.
Name	Enter the translated Product Name

5. Click **Save**.

Translating Category Details field value

You must translate the information in the category, sub-category, and option groups detail field values in the desired language and add them to the **Category Translation** related list for each record.

To add translated information in the Category Translation related list

1. In Salesforce Classic UI, go to App Menu > **Apttus Products Setup** > **Category**.
2. Open the category, sub-category, or option group record for which you want to add translated information.
3. Go to the Category Hierarchy associated with the record you opened.
4. Under **Category Translations** related list, click **New Category Translation**
5. Fill in the following details.

Field	Description
CategoryTranslation Name	Enter the name of the translation record
Language	Enter the 2 letter ISO code for language. For example, *FR* for French and *JP* for Japanese.
Label	Enter the translated text for the record.
CategoryHierarchy	Enter the name of the Category Hierarchy associated with the record.

6. Click **Save**.

Translate Field Values on the Cart page

Unlike the Catalog and Configuration page, the Cart page retrieves values of fields from the Line Item object. To display the translated details on the Cart page, you must create formula fields on the Line Item object to enable CPQ to use the translated details from the Product Translation and Category Translation object.

To create a formula field in the Line Item object

1. Go to **Setup > App Setup > Create > Objects** and select **Line Item**
2. Under **Custom Fields & Relationships** section, click **New**.
3. Select **Formula** from the Field Type.
4. Click **Next**.
5. In **Field Label**, enter the name of the Product or Category field for which you want to translate the values.
6. Click the **Field Name** textbox, the value is populated automatically.
7. From **Formula Return Type**, select **Text**.
8. Click **Next**.
9. Enter the formula to return the text value of the product field value of which has to be translated. For example, if you want to display the field value of **Product Type** on the cart, enter *TEXT(Apttus_Config2__ProductId__r.Apttus_Config2__ProductType__c)*.
10. Establish field-level security and click **Next**.
11. Select appropriate page layouts where you want to add this field.
12. Click **Save.**

Translation of Field Labels

You can use this feature to translate the field labels. There are threes types of fields in a CPQ org, namely standard, custom, and CPQ managed package fields. You must follow different steps to translate the different types of fields.

This section describes the following topics:

- Translating Standard and Custom Field Labels
- Translating and Overriding Managed Packaged Field Label
- Using Translated Picklist Values in Document Generation

Translating Standard and Custom Field Labels

You can configure translation for out-of-the-box field names, picklist values, button labels, and error messages directly in the **Translation Workbench**. For custom label translation, you must upload Salesforce Translation File. Follow the steps below to configure the translation of content.

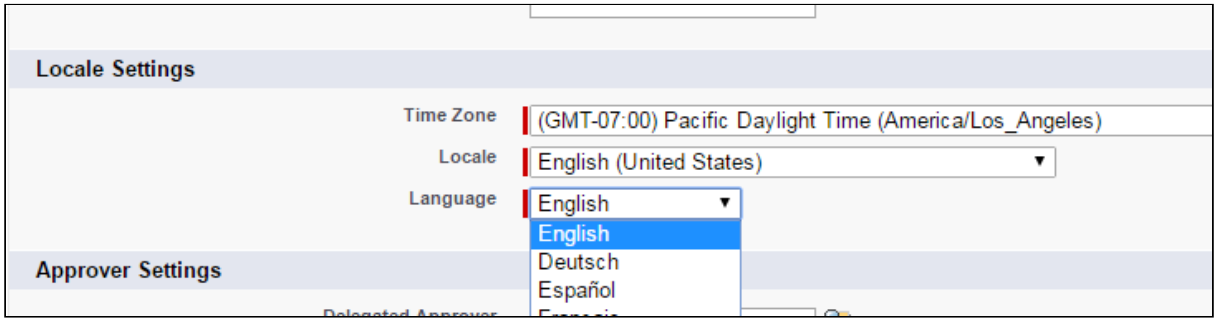
To configure Translation Workbench for translating the label data

1. Go to **Setup** > **Administration Setup** > **Translation Workbench** > **Translation Language Settings**.
2. Click **Add** button.
3. Select language in the **Language** dropdown.
4. Select **Active** to activate the language.
5. Under **Identify Translators for this Language**, you must define User who can be the translators. Select a User from **Available List** and click **Add**.
6. Click **Save**.

To change the language at the profile level

1. Go to **Setup** > **Manage Users** > **Users**.
2. Click **Edit** next to the desired profile.

3. Under the **Locale Settings**, change the value in the **Language** field and click **Save**.

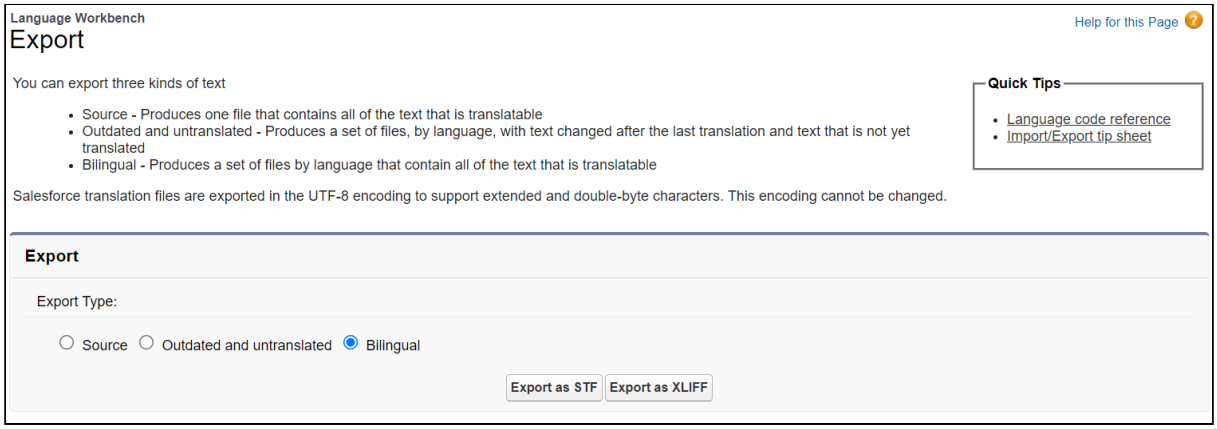


> You must modify the **Language** field to match the **Language** field in the Product Translation and Category Translation records. CPQ only displays the translated details when the **Language** field matches in both the records. However, the Locale itself is not a criterion.

To mass upload translations for custom labels

You need to upload a new Salesforce Translation File (STF) through the Translation Workbench to translate customer labels

1. Go to **Setup** > **Administration Setup** > **Translation Workbench** > **Export**
2. Select **Bilingual**.
3. Click **Export as STF**



4. Translate the STF (non-translated items will show up in English).
5. Go to **Setup** > **Translation Workbench** > **Import**
6. Click **Choose File**.
7. Select the translated STF file.
8. Click **Import**.

You can translate the field labels to the desired language.

To translate an individual custom label

1. Go to **Setup > Create > Custom Labels**.
2. Search and open the label you want to translate. The system displays the Custom Label Detail page.
3. Click the button, **New Local Translations/Overrides** in the Local Translations/ Overrides section. The system displays the New Translation page.
4. Select the appropriate language from the drop-down.
5. Enter the Translation Text as an Add-on in the text box.
6. Click **Save**.

Translating and Overriding Managed Packaged Field Label

By default, CPQ provides the labels in English and Japanese language. You must use the Override feature to customize the labels of components for CPQ managed package objects in English and Japanese. Whereas, you must use the Translate feature to translate the labels to any other language. Follow the steps below based on the task you want to acheive.

To override CPQ labels in English and Japanese languages

1. Go to **Setup** > **Administration Setup** > **Translation Workbench** > **Override**.
2. Define the fields based on the below description.

Field	Description
Package	Select the Conga package where the label you want to translate belongs.
Language	Select the desired language.
Setup Component	From the dropdown, select the type of component that you want to translate. For example, if you want to translate the label of the custom field, select *Custom Field* in **Setup Component.**
Object	Select the object where the component belongs to.

Field	Description
Aspect	Select the type of label you want to translate.

3. Find the label that you want to translate and add the translated text of the label in the **Field Label Override** column in the respective row.
4. Click **Save.**

To translate CPQ labels to languanges other than English and Japanese

1. Go to **Setup** > **Administration Setup** > **Translation Workbench** > **Translate**.
2. Define the fields based on the below description.

Field	Description
Language	Select the desired language.
Setup Component	From the dropdown, select the type of component that you want to translate. For example, if you want to translate the label of a button, select *Button and Link Label* in **Setup Component.**
Object	Select the object where the component belongs to.

3. Find the component label that you want to translate and enter the translated text of the label in the **Field Label Override** column in the respective row.
4. Click **Save**.

Use Case: Overriding Label of a Button

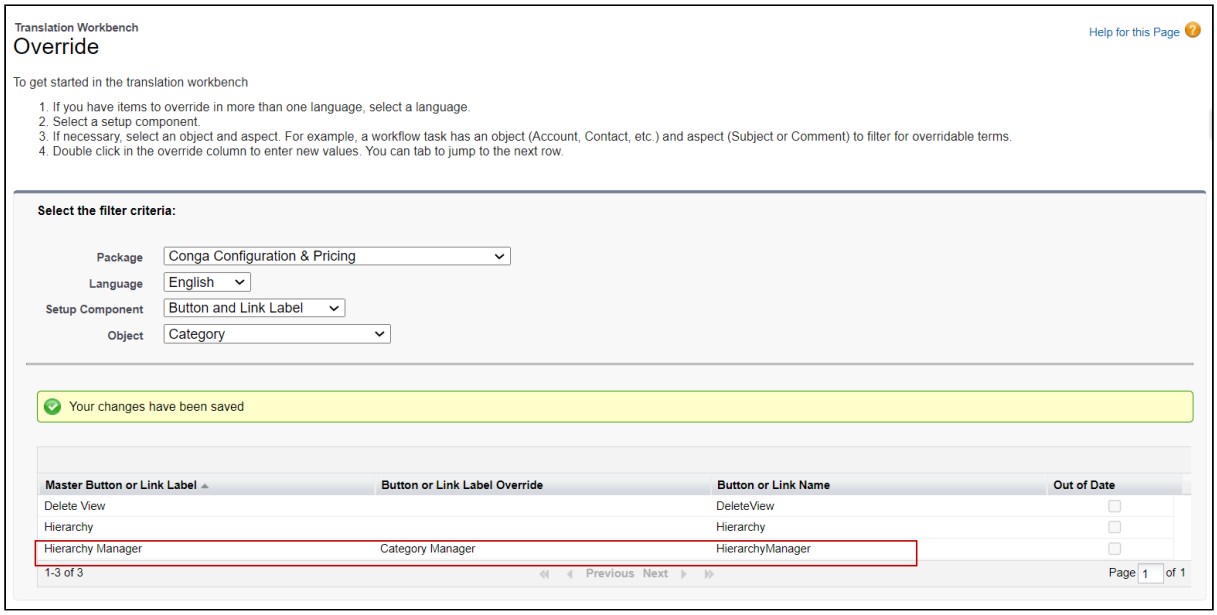
In the scenario where you want to change the label of a CPQ managed package components. For example, you need to override the label of the field **Hierarchy Manager** button to **Category Manager**. Follow the steps below.

1. Go to **Setup** > **Administration Setup** > **Translation Workbench** > **Override**.
2. Define the fields based on the below description.

Field	Description
Package	Select the **Conga Configuration & Pricing**.

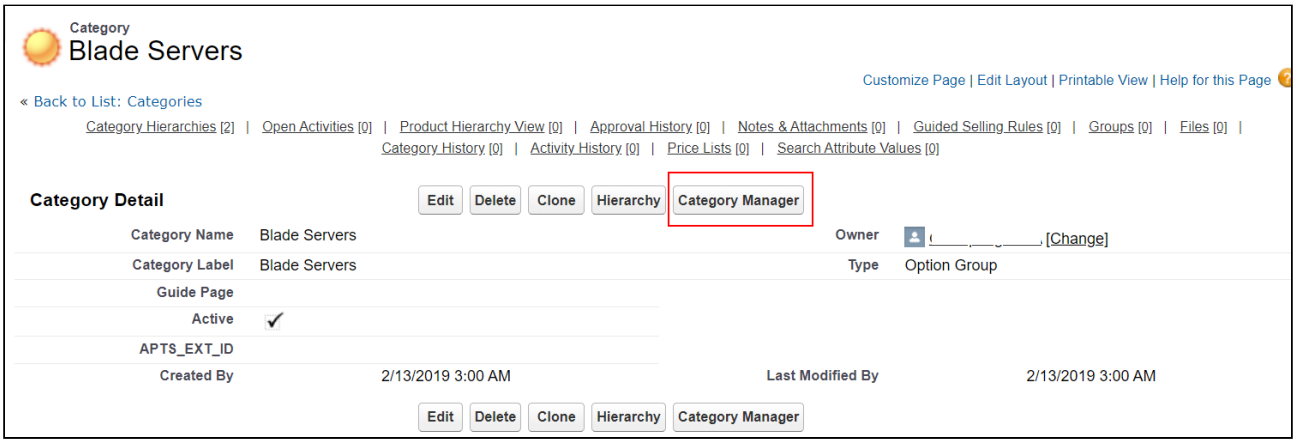
Field	Description
Language	Select **English**.
Setup Component	From the dropdown, **Button and Link Label**.
Object	Select **Category**.
Aspect	When you select **Button and Link Label**, the field **Aspect** is not required.

3. Find the label **Hierarchy** in the **Master Button or Link Label**.
4. Enter **Category Manager** in the **Field Label Override** column in the respective row.



5. Click **Save**.

The overridden value is displayed on the Category Detail page.



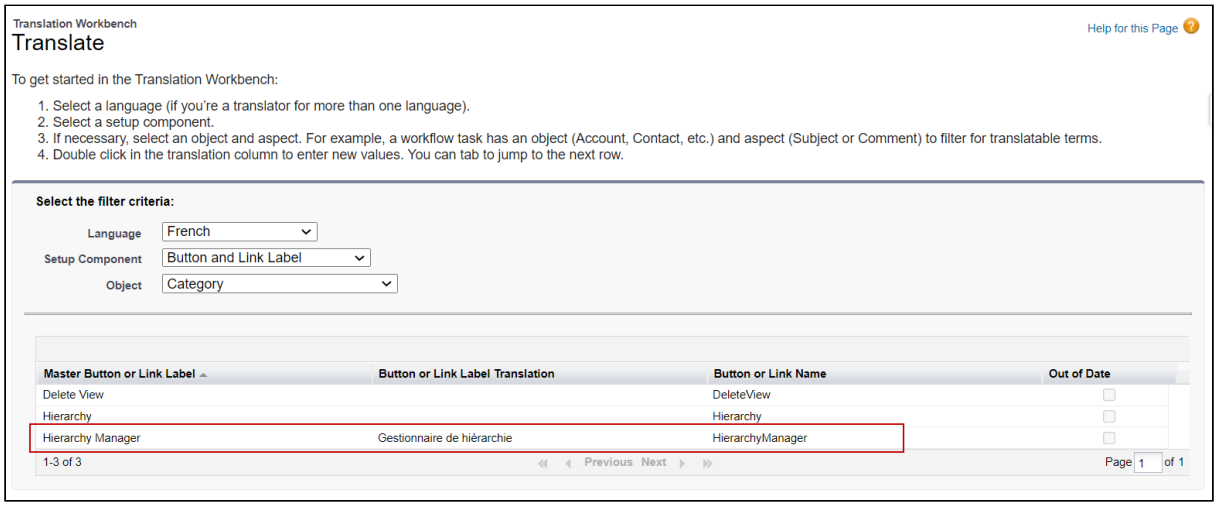
Use Case: Translating Label of a Button

In the scenario where your company has a base in the European regions and you need to translate the field labels in French for the Sales Rep working in a French locale. For example, you need to translate the label of the field **Hierarchy Manager** button to french. Follow the steps below.

1. Go to **Setup** > **Administration Setup** > **Translation Workbench** > **Override**.
2. Define the fields based on the below description.

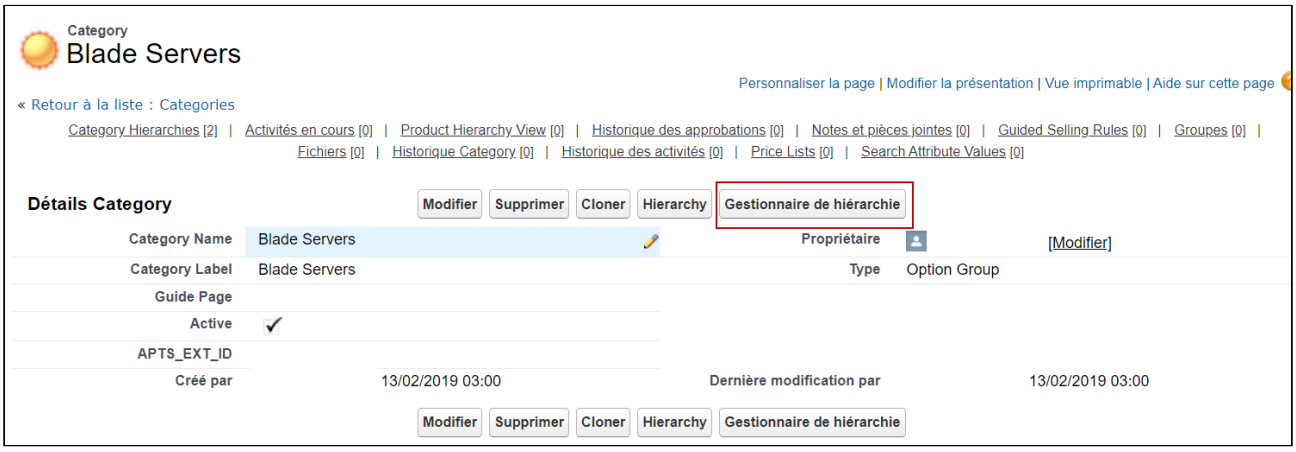
Field	Description
Language	Select the **French**.
Setup Component	From the dropdown, **Button and Link Label**.
Object	Select **Category**.
Aspect	When you select **Button and Link Label**.

3. Find the label **Hierarchy** in the **Master Button or Link Label**.
4. Enter the translated text of the label in the **Field Label Override** column in the respective row.



5. Click **Save**.

The translated value is displayed on the Category Details page.



Using Translated Picklist Values in Document Generation

You can enable non-English users to get translated values of picklist during document generation. To achieve this, you must set the **APTS_TranslatePickValuesInDocs** admin setting to true. If you do not configure this admin setting or set it to false, CPQ uses the values defined in English in the generated document instead of the translated values.

To enable use of translated picklist values in document generation

1. Go to **All Tabs** > **Admin**
2. Click **New**, to create a new record.
3. Fill in the following details

Field	Value
Name	*APTS_TranslatePickValuesInDocs*
Value	*true*
Code	Leave the field blank

4. Click **Save**.

Customizing CPQ Using Callbacks

Callbacks provide you with a mechanism to apply custom logic on different components of CPQ like line items, Cart page, Catalog page at run-time. For example, you can apply custom pricing or validation on the line items in the cart using Pricing Callback Class and Validation Callback Class. Callbacks are implemented using interfaces that are specific to each callback. These interfaces have various methods you can use to achieve your task. You must implement the interface in an Apex class and within that Apex class, you must configure your custom logic using the methods of the interface. Once you implement the interface, register that Apex class in Custom settings to invoke the callback class on run-time.

To register Callback Class in Custom Settings

To use the Callback Class you created in Apex Classes you must register the Callback Class in the **Config Custom Classes** Custom Settings. Follow the steps below to register the Callback Class

1. Go to **Setup** > **Develop** > **Custom Settings**.
2. Click **Manage** next to **Config Custom Classes**.
3. Click **Edit** next to the **Config Custom Classes** dataset.

   > ⓘ Avoid creating multiple datasets. Only create a new dataset if the **Config Custom Classes** dataset is missing. You must create a new entry with the name **Config Custom Classes.**

4. Enter the name of the custom Apex class next to the Callback you implemented.
5. Click **Save**.

In this section the following Callbacks are described:

- Display Action Callback Class
- Product Filter Callback Class
- Option Filter Callback Class
- Adjustment Line Item Callback Class
- Validation Callback Class
- Pricing Callback Class
- Pricing Extension Callback Class
- Product Attribute Callback Class
- Related Pricing Callback Class
- Asset Line Item Callback Class

- Asset Renewal Custom Callback Class
- Revalidation Callback Class
- Action Params Callback Class
- Action Invoker Callback Class
- Action Callback Class
- Cart Approval Callback Class
- Deal Optimizer Callback Class
- Adjustment Spread Callback Class
- Lifecycle Callback Class
- Tax Callback Class
- Formula Callback Class
- Metadata Callback Class
- Recommendation Callback Class
- Advanced Approval Callback Class

Display Action Callback Class

Display Action Callback provides you a mechanism to disable any of the action buttons on the cart. Following are few examples of custom buttons that you can disable:

- Abandon
- Approvals
- Generate
- Quick Save

To use the Display Action Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IDisplayActionCallback* interface and register the custom apex class with Display Action Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IDisplayActionCallback* interface:

Method	Signature	Description
setActionProperties()	*void setActionProperties(Apttus_Config2.CustomClass.ActionParams)*	You can use this method to enable or disable an action button.

Method	Signature	Description
start()	*void start(Apttus_Config2.ProductConfiguration, List)*	This is the first method that is executed when the callback is invoked.

Example

This is just a sample callback custom class to hide the visibility of the **Abandon** and **Generate** action buttons on the cart page. You may change the custom class to fit your business requirements

```
global with sharing class SampleDisplayActionCallback implements
 Apttus_Config2.CustomClass.IDisplayActionCallback{

    List<Apttus_Config2.DisplayActionInfo> displayActions;
    global void start(Apttus_Config2.ProductConfiguration cart,
List<Apttus_Config2.DisplayActionInfo>displayActions){
        this.displayActions = displayActions;
    }
    global void setActionProperties(Apttus_Config2.CustomClass.ActionParams
actionParam){
        for (apttus_config2.DisplayActionInfo action : displayActions) {
            if(action.ActionSO.Apttus_Config2__ActionName__c == 'Abandon'){
                action.isEnabled = false;
            }
            if (action.ActionSO.Apttus_Config2__ActionName__c == 'Generate'){
                action.isEnabled = false;
            }
         }
     }
}
```

Product Filter Callback Class

Product Filter Callback provides you a mechanism to control the visibility of products on the catalog page based on custom criteria that you define. You can define entry criteria to determine whether a product in a price list should be filtered to a certain subset of the full list. You can define the criteria to select standalone products and option products as well. When you click **Configure Products** from a Quote/Proposal or **Add More Products** from the Cart, the CPQ checks whether the criteria are applicable for the specific price list.

To use the Product Filter Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IProductFilterCallback* interface and register the custom apex class with Product Filter Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IProductFilterCallback* interface:

Method	Signature	Description
getOptionFilterExpr()	*String getOptionFilterExpr(Apttus_Config2.CustomClass.ActionParams)*	You can use this method to control the visibility of option on the Catalog page.
getProductFilterExpr()	*String getProductFilterExpr(Apttus_Config2.CustomClass.ActionParams)*	You can use this method to control the visibility of Stanalone and bundle products on the Catalog page.

Example

This is just a sample callback custom class to control the visibility of products on the Catalog page. The following only filter the Catalog products having color Red and the options products having color Green. You may change the custom class to fit your business requirements.

```
global with sharing class SampleProductFilterCallback implements
 Apttus_Config2.CustomClass.IProductFilterCallback {
    /**
     * Callback to return part of SOQL filter clause
     * This filter is used in listing catalog products
     * @param params is the Apttus_Config2.CustomClass.ActionParams that
     * contains accountIds, locationIds when available
     * @return The query filter is like the following.
     * Name LIKE 'A%' AND Quantity__c > 100
     * Id IN ('000123', '000124')
     */
    global String
getProductFilterExpr(Apttus_Config2.CustomClass.ActionParams params) {
        return 'Color__c = \'Red\'';
    }
```

```
14
15      /**
16       * Callback to return part of SOQLfilter clause
17       * This filter is used in listing option products
18       * @param parsms is the CustomClass.ActionParams that contains
    accountIds, locationIds when available
19       * @return The query filter is like the following.
20       * Name LIKE 'A%' AND Quantity__c > 100
21       * Id IN ('000123', '000124')
22       */
23      global String getOptionFilterExpr(CustomClass.ActionParams params) {
24          return 'ComponentProductId__r.Color__c =\'Green\'';
25      }
26  }
```

Option Filter Callback Class

Option Filter Callback provides you with a mechanism to control the visibility of the option products on the Configuration page based on the custom logic that you define.

The callback is executed in the following scenarios:

- when you click **Configure** on the Catalog page
- when the configuration of the product is pending and you click the wrench icon ( ) on the Cart page.

When callback is invoked, CPQ excludes the option products on the Configuration page.

> Default options and options included based on constraint rules are also hidden when the Option Filter Callback is used to hide options. In the scenario, where all the case all the options in an option group are hidden, the option group is also hidden on the Configuration page.

To use the Option Filter Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IOptionFilterCallback* interface and register the custom apex class with Option Filter Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IOptionFilterCallback* interface:

Method	Signature	Description
getExcludedOptionIds()	*List getExcludedOptionIds(Apttus_Config2.CustomClass.ActionParams)*	You can use this method to define custom logic to hide options on the Configuration page.

Example

This is just a sample callback custom class to control the exclusion of option products on the catalog page. The following custom code only removes a few option products. You may change the custom class to list out the ID of option products to fit your business requirements.

```
global with sharing class SampleOptionFilterCallback implements
 Apttus_Config2.CustomClass.IOptionFilterCallback {

    /**
     * Callback to return option IDs which are to be excluded from the bundle
     * This filter is used when we configure a bundle.
     * @param params is the CustomClass.ActionParams that contains bundleId and
productIds
     * @return List of option product IDs which will be excluded
     */
    global List<ID> getExcludedOptionIds(Apttus_Config2.CustomClass.ActionParams
params) {
        //return new List<ID>{ <18 CHARACTER PRODUCT ID> };
        //Example for returning THREE products to be excluded:
            return new List<ID>{ '01t3C000000DVwh', '01t3C000000DVwm','01t3C000000DVw
w'};
        //Example for returning ONE product to be excluded:
        //  return new List<ID>{ 01t50000004nmL1YYX'};
    }
}
```

Adjustment Line Item Callback Class

Adjustment Line Item Callback allows you to write custom logic to auto-create or auto-delete adjustments for the line items. For example, once a product is added to cart, some adjustments need to be auto-included based on the account contract or other terms. In

such a case, you can use Adjustment Line Item callback to automate this process. When you apply an adjustment on a proposal, the callback will overwrite manual adjustments.

Agents can change the discounts anytime and that will not reflect on the cart. Agents cannot do this manually per line item, so we have given the custom page for them to manage the discounts.

To use the Adjustment Line Item Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IAdjustmentLineItemCallback2* interface and register the custom apex class with Adjustment Line Item Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IAdjustmentLineItemCallback2* interface:

Method	Signature	Description
createAdjustmentLineItems()	*List createAdjustmentLineItems(Apttus_Config2.LineItem)*	You can use this method to create adjustments for the given parent line item
finish()	*void finish()*	This the last method that is executed after the callback is invoked.
start()	*void start(Apttus_Config2.ProductConfiguration)*	This is the first method that is executed when the callback is invoked.
beforeSaveAdjustmentLineItems()	*Boolean beforeSaveAdjustmentLineItems (Apttus_Config2.LineItem, List)*	You can use this method to apply custom logic after the user adjustments are finished and before the changes are saved.

Example

The given code example implements *createAdjustmentLineItems()* method to create new adjustment line items using the *IAdjustmentLineItemCallback2* interface.

```
1  global with sharing class SampleAdjustmentCallback
2                          implements
    Apttus_Config2.CustomClass.IAdjustmentLineItemCallback,
```

```
3
Apttus_Config2.CustomClass.IAdjustmentLineItemCallback2 {
4
5       /**
6        * Callback at the beginning of the adjustment call.
7        * Use the start method to initialize state
8        * @param cart the cart object
9        */
10      global void start(Apttus_Config2.ProductConfiguration cart) {
11          // do nothing
12      }
13
14      /**
15       * Callback to create adjustments for the given parent line item
16       * @param parentItem the parent line item associated with the
adjustments
17       * @return the list of adjustment line item sobjects
18       */
19      global List<Apttus_Config2.AdjustmentLineItem__c>
createAdjustmentLineItems (
20
Apttus_Config2.LineItem parentItem) {
21
22          List<Apttus_Config2.AdjustmentLineItem__c> adjItems = new
 List<Apttus_Config2.AdjustmentLineItem__c>();
23
24          // test data
25          if (Test.isRunningTest() == true &&
SystemUtil.isMultiAdjustmentEnabled() == true) {
26              // get the parent item sobject
27              Apttus_Config2__LineItem__c parentItemSO =
parentItem.getLineItemSO();
28
29              // create a line item adjustment
30              //AdjustmentLineItem__c adjItemSO = new
AdjustmentLineItem__c(LineItemId__c = parentItemSO.Id);
31              Apttus_Config2__AdjustmentLineItem__c adjItemSO = new
 Apttus_Config2__AdjustmentLineItem__c();
32              // line number
33              adjItemSO.LineNumber__c = 1;
34              // line type
35              adjItemSO.LineType__c = 'Manual';
36              // modifiable
```

```
            adjItemSO.IsModifiable__c = true;
            // adjustment type
            adjItemSO.AdjustmentType__c = '% Discount';
            // adjustment amount
            adjItemSO.AdjustmentAmount__c = 10;
            // adjustment applies to
            adjItemSO.AdjustmentAppliesTo__c ='Starting Price';

            adjItems.add(adjItemSO);
        }
        return adjItems;
    }
    /**
     * Callback before the adjustment line items are saved after user changes
     * @param parentItem the parent line item associated with the adjustments
     * @param adjItems the list of adjustment line items associated with the save operation
     * @return <code>true</code> if adjustment line items were modified by the callback,
     * <code>false</code> otherwise
     */
    global Boolean beforeSaveAdjustmentLineItems(Apttus_Config2__LineItem parentItem,
                                                 List<Apttus_Config2__AdjustmentLineItem__c> adjItems) {
        return false;
    }

    /**
     * Callback after all batches of adjustment line items are processed
     * Use the finish method to release state
     */
    global void finish() {
        // do nothing
    }
}
```

Validation Callback Class

Validation Callback provides a mechanism to implement custom validations on the line items in the cart. Validations are any constraints you define on the line item to restrict the

selection of that line item based on any condition or criteria. When you define validations, CPQ checks if the condition or the criteria is satisfied and displays an error message accordingly. For example, you can validate if the **Quantity** field for any product is negative and prevent the sales rep from finalizing the cart.

Validation Callback applies validation on the following aspects in the configuration:

- Line items in the cart
- Price Ramps

To use the Validation Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IValidationCallback3* interface and register the custom apex class with Validation Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IValidationCallback3* interface:

Method	Signature	Description
validateCart()	*Apttus_Config2.CustomClass.ValidationResult validateCart (Apttus_Config2.CustomClass.ActionParams, Apttus_Config2.ProductConfiguration)*	You can use this method to apply validation on line items in the cart. This method is invoked when the Sales rep performs the following actions: • **View Cart** ⓘ To invoke Validation Callback when click **View Cart** on the Mini Cart, you must enable **Run Validation Callback On Add** in Config System Properties custom setting. For more information, see Config System Properties. • **Reprice** • **Save** • **Finalize** • **Analyze Deal** • Any action that is available on the Cart and Catalog page. • Custom actions

Method	Signature	Description
validateRampLineItems()	*Apttus_Config2.CustomClass.ValidationResult validateRampLineItems (Apttus_Config2.CustomClass.ActionParams, Apttus_Config2.ProductConfiguration, List)*	You can use this method to apply validation on the ramp line items in a Price Ramp. This method is invoked when the Sales rep performs the following actions: • Click **Save** on the Price Ramp pop-up • Any action that is available on the Cart and Catalog page when price ramps exist for any line item.

The following are examples of the custom logic you can implement using Validation example

Example

You can use *validateCart()* apply the following validations:

- Check if the **Duration** for any product is null.
- Check if the value in the **Adjustment Amount** field is negative.

The given code example implements two methods of *IValidationCallback3* interface. The code sample implements the abovementioned validation using the method **validateCart()**.

```
global with sharing class Test_ValidationCallBack_UC1 implements
 Apttus_Config2.CustomClass.IValidationCallback3 {

    global Apttus_Config2.CustomClass.ValidationResult
validateCart(Apttus_Config2.CustomClass.ActionParams params,
Apttus_Config2.ProductConfiguration cart) {
        system.debug('***ValidationCallBack > validateCart() > ' + cart);

        Apttus_Config2.CustomClass.ValidationResult result = new
 Apttus_Config2.CustomClass.ValidationResult(true);

        list <Apttus_Config2.LineItem> allLines = cart.getLineItems();
        list<Apttus_Config2__LineItem__c> allLineItems = getLineItems(allLines);


        Id configurationId;
        if(allLineItems != null) {
            configurationId = allLineItems[0].Apttus_Config2__ConfigurationId__c;
        }
```

```
        Integer i = 0;
        System.debug('***ValidationCallBack > allLineItems: '+allLineItems.size() + '
<> ' + allLineItems);
        for(Apttus_Config2__LineItem__c lineItem : allLineItems) {
            system.debug('***ValidationCallBack > ' + (++i) + ' > lineItem : ' +
lineItem);

            // Check if the Duration for any product is null.
            if(lineItem.Apttus_Config2__SellingTerm__c == null) {
                result.Messages.add(new ApexPages.Message(ApexPages.Severity.ERROR,
'Value of Duration for product : \'' + lineItem.Apttus_Config2__Description__c + '\'
is None.'));
                result.isSuccess =  false;
            }
            // Check if the value in the Adjustment Amount field is negative.
            if(lineItem.apttus_config2__adjustmentamount__c < 0) {
                result.Messages.add(new ApexPages.Message(ApexPages.Severity.ERROR,
'Adjustment Amount for product : \'' + lineItem.Apttus_Config2__Description__c + '\'
is Negative. It should be >= 0'));
                result.isSuccess =  false;
            }
        }
        return result;
    }


    global Apttus_Config2.CustomClass.ValidationResult
validateRampLineItems(Apttus_Config2.CustomClass.ActionParams params,
Apttus_Config2.ProductConfiguration cart,List<Apttus_Config2.LineItem> lstLI) {
        system.debug('***ValidationCallBack > validateRampLineItems() > Cart : ' +
cart + ' <> lstTemp : ' + lstLI);
        Apttus_Config2.CustomClass.ValidationResult result;
        return result;
    }


    /* Gets the list of product line items associated with the Battery line
     * @param cart the cart object
     * @return the list of line item objects
    */
    private static List<Apttus_Config2__LineItem__c>
getLineItems(List<Apttus_Config2.LineItem> allLines) {
```

```
        list<Apttus_Config2__LineItem__c> lineItems = new
list<Apttus_Config2__LineItem__c>();
        // iterate through the cart and get the line items matching the battery code1
        for (Apttus_Config2.LineItem lineItemMO : allLines) {
            lineItems.add(lineItemMO.getLineItemSO());
        }

        return lineItems;
    }
 }
```

Pricing Callback Class

The Pricing Callback class enables you to add pricing logic to your cart that cannot be achieved by out-of-the-box pricing mechanisms, such as Price Rulesets and Price Matrices. The Pricing Callback is executed for each bundled product or standalone in the cart. A separate transaction call is initiated for each product in the cart. The Pricing Callback is invoked when you are on the cart page. The pricing callback is invoked during pricing action and executes the various methods in three modes: BASEPRICE, and ADJUSTMENT.

To use the Pricing Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IPricingCallback3* interface and register the custom apex class with Pricing Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IPricingCallback3* interface:

Method	Signature	Description
afterPricing()	*void afterPricing(Apttus_Config2.ProductConfiguration.LineItemColl)*	You can use this method to define the logic to be executed after Net Price calculation. The collection of line items currently in the cart is passed as an input parameter to this method.
afterPricingLineItem()	*void afterPricingLineItem(Apttus_Config2.ProductConfiguration.LineItemColl, Apttus_Config2.LineItem)*	You can use this method to define custom logic that has to be applied after the Net Price has been calculated. This method called for each line item in the cart.

Method	Signature	Description
beforePricing()	*void beforePricing(Apttus_Config2.ProductConfiguration.LineItemColl)*	You can use this method to define the logic for net price calculation. In this method, no pricelist item is associated to the cart line items. Price Matrix and Price Ruleset calculation occurs when the mode of the callback is BASEPRICE Mode.
beforePricingLineItem()	*void beforePricingLineItem(Apttus_Config2.ProductConfiguration.LineItemColl, Apttus_Config2.LineItem)*	You can use this method to define custom logic that has to be applied after the **Base Price** has been calculated but before the downstream prices like **Base Extended Price**, and **Extended Price** is calculated. This method called for each line item in the cart.
finish()	*void finish()*	This the last method that is executed after the callback is invoked.
setMode()	*void setMode(Apttus_Config2.CustomClass.PricingMode)*	You can use this method to specify the mode of the callback. Based on the mode of the callback you can define different business logic. The modes are: • **ADJUSTMENT**: The mode of the call back when adjustments are made to the cart and you click **Reprice**. • **BASEPRICE**: The mode is base price when you click **Go to Price**.
start()	*void start(Apttus_Config2.ProductConfiguration)*	This is the first method that is executed when the callback is invoked.

Method	Signature	Description
onPriceItemSet()	*void onPriceItemSet(Apttus_Config2__PriceListItem__c, Apttus_Config2.LineItem)*	You can use this method to associate custom fields with line items and apply logic related to price list item. This method is called on each line item.

BASEPRICE mode is used primarily to set quantity, term and so on in the line item *beforePricing()* method. Base price mode is called for each line number. If there are two line numbers, the BASEPRICE mode would be invoked twice. The following methods in the callback are invoked during the BASEPRICE mode:

- *start()*
- *setMode()*
- *beforePricing()*
- *onPriceItemSet()*
- *afterPricing()*
- *finish()*

The line item collection in *beforePricing()* may not match the line item collection in *afterPricing()* if the product has multiple charge types. In this case, the *afterPricing()* method would have more line items. Any modification to custom fields needs to be done in the *beforePricing()* method.

ADJUSTMENT mode is invoked for all items (including options) in the cart after applying adjustments to compute **Net Price**. Only the adjusted price or the adjustment amount may be changed at this point. The following methods in the callback are invoked during the Adjustment mode:

- *start()*
- *setMode()*
- *beforePricing()*
- *beforePricingLineItem()*
- *afterPricingLineItem()*
- *afterPricing()*
- *finish()*

Depending on the discount applied at the group level, the *beforePricingLineItem()* and *afterPricingLineItem()* may be applied at the option lines after the execution of the methods at the bundle lines level.

When you define a pricing callback, ensure that you verify the following:

- Implement the latest interface version 3 for your pricing callback class.

- Choose the correct mode for your custom logic because all or some of the methods in the class will be called during each mode.
- The most widely used modes are either BASEPRICE or ADJUSTMENT.
- It is recommended to have logic in the BASEPRICE mode only, in the *beforePricing()*, *beforePricingLineItem()*, *afterPricing()*, and *afterPricingLineItem()* methods. ADJUSTMENT mode is an exception and should be reviewed with the Product team (PM, PST, Engg) before approval for implementation.
- ADJUSTMENT mode does not have all option items in Large Cart mode and hence it is preferable not to use this mode.
- ADJUSTMENT mode is not executed when the parameter **isCartTotalingDisabled** is set to *true* in the **Configure Products** button URL.
- Any DML statement is not required. All changes related to line items are covered regardless of DML. However, if there are other object record changes then make sure that the query is not called multiple times in a single transaction of pricing. For example, if you want to compute the **Base Price** which is based on some custom logic, assign the final calculated value to **Quantity**, **Selling Term**, or any custom field on the line item.
- If your logic is dependent on **Base Price** calculation, then make sure to use **Base Price Override** field of Line Item. You can put the newly computed **Base Price** in this field and then in the next mode, the system will take the value of **Base Price** from the override field to compute **Extended Price** and **Net Price**.
- When you do a **Reprice** on the Cart page and change any field on the line item, which is used in Price Matrix or Price Ruleset, then BASEPRICE mode will get called along with ADJUSTMENT mode. If you change fields on the line item which are not impacting Base Price Calculations (e.g. Giving 10% discount on a bundle line item), then only ADJUSTMENT mode is called.
- Option Line Item are not be called in *beforePricingLineItem()* or *afterPricingLineItem()* methods in any mode. If you need to work on an option line item, then do it in *beforePricing()* or *afterPricing()* methods.
- If you want to write any custom logic that depends on price list item fields, then you should use *onPriceItemSet()* method. In this method, the system determines the corresponding price list item for a line item.
- If you have queries in the callback class, it is advisable to do it in *start()* method so that it is not repeated for each line item in the before/after methods.
- Use of *updatePrice()* method should be avoided unless there is approval from the Product team.
- The custom fields on Price List Item are not retrieved in the **onPriceItemSet** method automatically. To auto-retrieve the custom fields in this method, the Administrator must create a new entry named *APTS_PriceListItemCustomFields* in **Admin**. Enter the

API names of the custom field of the Price List Item in the **Code** field separated by a comma or on a new line.

```
/**
 * Apttus Config & Pricing
 * DefaultPricingCallback2
 *
 * @2011-2014 Apttus Inc. All rights reserved.
 */
global with sharing class SamplePricingCallback2 implements
 Apttus_Config2.CustomClass.IPricingCallback3 {

   private Apttus_Config2.ProductConfiguration contextCart;
   private Apttus_Config2.CustomClass.PricingMode currentMode;

 /**
 * Callback at the beginning of the pricing call.
 * Use the start method to initialize state
 * @param cart the cart object
 */
 global void start(Apttus_Config2.ProductConfiguration cart) {
 this.contextCart = cart;
 }

 /**
 * Callback to indicate the pricing mode
 * @param mode the pricing mode
 */
 global void setMode(Apttus_Config2.CustomClass.PricingMode mode) {
 this.currentMode = mode;
 }
 /**
 * Callback after the price list item is set on the given line item
 * @param itemSO the price list item associated with the line item
 * @param lineItemMO the line item
 */
 global void onPriceItemSet(Apttus_Config2__PriceListItem__c itemSO,
Apttus_Config2.LineItem lineItemMO){
 }

 /**
 * Callback before pricing the line item collection
 * Use this method to do all required pre-processing to prepare the line items for
pricing.
```

```
 * @param itemColl the line item collection to pre-process
 */
 global void beforePricing(Apttus_Config2.ProductConfiguration.LineItemColl itemColl)
{
 }

 /**
 * Callback before pricing the given line item in the line item collection
 * Use this method to do all required pre-processing to prepare the line item for
pricing.
 * @param itemColl the line item collectionholding the line item
 * @param lineItemMO the line item to pre-process
 */
 global void beforePricingLineItem(Apttus_Config2.ProductConfiguration.LineItemColl
itemColl, Apttus_Config2.LineItem lineItemMO) {
 }

 /**
 * Callback after pricing the given line item in the line item collection
 * Use this method to do all required post-processing after the line item is priced
 * @param itemColl the line item collection holding the line item
 * @param lineItemMO the line item to post-process
 */
 global void afterPricingLineItem(Apttus_Config2.ProductConfiguration.LineItemColl
itemColl, Apttus_Config2.LineItem lineItemMO) {
 }

 /**
 * Callback after pricing the line item collection
 * Use this method to do all required post-processing after line items are priced.
 * @param itemColl the line item collection to post-process
 */
 global void afterPricing(Apttus_Config2.ProductConfiguration.LineItemColl itemColl)
{
 }

 /**
 * Callback after all batches of line items are processed
 * Use the finish method to release state
 */
 global void finish() {
 }
```

```
}
```

Pricing Extension Callback Class

You can use the Pricing Extension Callback Class to define any logic or computation you need to execute before or after the pricing is calculated for the line items. You must define the pre-pricing or post-pricing hooks in the Pricing Extension Callback. The pre-pricing and post-pricing logic are executed separately from the pricing that is calculated when the Sales rep clicks **Submit for Pricing(Async)** button. This allows CPQ to run the pricing without executing custom logic every time.

The pre-pricing and post-pricing logic are executed when the Sales rep clicks **Pre-Price** and **Post-Price** buttons respectively, on the Cart page. If you define **Auto Execute Post-Pricing Step** and **Auto Execute Pre-Pricing Step** settings, the pre-pricing and post-pricing logics are executed automatically when the Sales rep clicks **Submit for Pricing(Async)**. For more information, see Config System Properties.

The following methods are available to you in Pricing Extension Callback Class.

Method	Signature	Description
afterPricing()	*Apttus_Config2.CustomClass.PricingExtensionResult afterPricing(Apttus_Config2.ProductConfiguration, Apttus_Config2.CustomClass.CartHelper)*	You can use this method to execute logic that is required post-processing after line items are priced
beforePricing()	*Apttus_Config2.CustomClass.PricingExtensionResult beforePricing(Apttus_Config2.ProductConfiguration, Apttus_Config2.CustomClass.CartHelper)*	You can use this method to execute logic that is required pre-processing to prepare the cart line items for pricing.
isAfterPricingEnabled()	*Boolean isAfterPricingEnabled()*	You can use this method to determine whether after-pricing should be invoked for the callback.
isBeforePricingEnabled()	*Boolean isBeforePricingEnabled()*	You can use this method to determine whether before-pricing should be invoked for the callback.

Pre-Pricing Logic

You can define logic in beforePricing() method of Pricing Extension callback to execute any computation that drives the pricing of the line item. For example, you can calculate the Total Quantity of reagents across all the locations and use that quantity to determine the tiered price for a line item. You must specify the fields you apply pre-price to in **Custom Pre-Pricing Fields**. For more information, see Config System Properties.

Post-Pricing Logic

You can define logic in afterPricing() method of Pricing Extension Callback to execute any computations that are dependent on the pricing fields like **Net Price** and **Net Extended Price** which are calculated after pricing. For example, if you want to calculate the annual value for a given line item with 3 years of Selling Term and after applying adjustments the Net Price is 3000, then the annual value is 3000/3 = 1000.

Example Code

```
/**
 *  Apttus Config & Pricing
 *  DefaultPricingExtensionCallback
 *
 *  @2019-2020 Apttus Inc. All rights reserved.
 */
global with sharing class SamplePricingExtensionCallback3 implements
 Apttus_Config2.CustomClass.IPricingExtensionCallback
{

    /**
     * Indicates whether the before pricing callback is enabled
     * Use this method to determine whether before pricing should be invoked for the
callback
     * @return <code>true</code> if before pricing callback is enabled, <code>false</
code> otherwise
     */
    global Boolean isBeforePricingEnabled()
    {
        return true;
    }
```

```
    /**
     * Callback before pricing the given cart
     * Use this method to do all required pre-processing to prepare the cart line
items for pricing.
     * @param cart the cart object to pre-process
     * @param helper the cart helper class
     * @return the pricing extension result object
     */
    global Apttus_Config2.CustomClass.PricingExtensionResult
beforePricing(Apttus_Config2.ProductConfiguration cart,
Apttus_Config2.CustomClass.CartHelper helper)
    {
        Id cartId = cart.getConfigSO().Id;
        List<Apttus_Config2__LineItem__c> lineItemsToUpdate = new
 List<Apttus_Config2__LineItem__c>();
        List<Apttus_Config2__LineItem__c> lineItems = [SELECT Id, Name ,

Total_reagent_qty_for_the_site__c ,
                                                       PrePricingCriteriaField__c,
                                                       Apttus_Config2__Quantity__c,
                                                       Apttus_Config2__SyncStatus__c,

Apttus_Config2__PricingStatus__c
                                                       FROM
Apttus_Config2__LineItem__c
                                                       WHERE
Apttus_Config2__ConfigurationId__c = :cartId];

        system.debug ('****************** lineItems:' + lineItems);

        Decimal totalQty = 0;
        for (Apttus_Config2__LineItem__c lineItemSO : lineItems)
        {
            //lineItemSO.Apttus_Config2__Quantity__c = 10;
            totalQty += lineItemSO.Apttus_Config2__Quantity__c +
(lineItemSO.PrePricingCriteriaField__c != null
                                                                ?
lineItemSO.PrePricingCriteriaField__c
                                                                : 0);
            lineItemSO.Apttus_Config2__PricingStatus__c = 'Pending';
            lineItemSO.Apttus_Config2__SyncStatus__c = 'Pre-Pricing Complete';
            lineItemsToUpdate.add(lineItemSO);
        }

        for (Apttus_Config2__LineItem__c lineItemSO : lineItemsToUpdate)
```

```
        {
            lineItemSO.Total_reagent_qty_for_the_site__c = totalQty;
        }

        Database.update (lineItemsToUpdate);

        lineItemsToUpdate = new List<Apttus_Config2__LineItem__c>();

        List<Id> pricingCartIds = helper.getPricingCartsFor(cartId);
        for (Apttus_Config2__LineItem__c pricingItemSO : [SELECT Id, Name
                                        FROM Apttus_Config2__LineItem__c
                                        WHERE Apttus_Config2__ConfigurationId__c
IN :pricingCartIds])
        {

            //pricingItemSO.Apttus_Config2__Quantity__c = 10;
            pricingItemSO.Total_reagent_qty_for_the_site__c = totalQty;
            pricingItemSO.Apttus_Config2__SyncStatus__c = 'Pre-Pricing Complete';
            lineItemsToUpdate.add(pricingItemSO);
        }


        Database.update (lineItemsToUpdate);
        return new Apttus_Config2.CustomClass.PricingExtensionResult();

    }

    /**
     * Indicates whether the after pricing callback is enabled
     * Use this method to determine whether after pricing should be invoked for the
callback
     * @return <code>true</code> if after pricing callback is enabled, <code>false</
code> otherwise
     */
    global Boolean isAfterPricingEnabled() {
        return true;

    }

    /**
     * Callback after pricing the given cart
     * Use this method to do all required post-processing after line items are
priced.
```

```
     * @param cart the cart object to post-process
     * @param helper the cart helper class
     * @return the pricing extension result object
     */
    global Apttus_Config2.CustomClass.PricingExtensionResult
afterPricing(Apttus_Config2.ProductConfiguration cart,
Apttus_Config2.CustomClass.CartHelper helper) {
        Id cartId = cart.getConfigSO().Id;
        List<Apttus_Config2__LineItem__c> lineItemsToUpdate = new
 List<Apttus_Config2__LineItem__c>();

        List<Id> pricingCartIds = helper.getPricingCartsFor(cartId);
        for (Apttus_Config2__LineItem__c lineItemSO : [SELECT Id, Name,
Apttus_Config2__SyncStatus__c
                                        FROM Apttus_Config2__LineItem__c
                                        WHERE Apttus_Config2__ConfigurationId__c
                                                IN :pricingCartIds]) {

            lineItemSO.Apttus_Config2__SyncStatus__c = 'Post-Pricing Complete';
            lineItemSO.Apttus_Config2__Description__c = 'Net Price should be updated
as per the new quantity 10';
            //lineItemSO.Custom_Field__c = 'Post-price';
            lineItemsToUpdate.add(lineItemSO);
        }

        Database.update (lineItemsToUpdate);
        return new Apttus_Config2.CustomClass.PricingExtensionResult();
    }

}
```

Product Attribute Callback Class

Product Attribute Callback allows you to set the default value of the product attributes based on custom logic at run-time. The callback is invoked when you open the Configuration page to select attributes and options for the products.

To use the Product Attribute Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IProductAttributeCallback* interface and register the custom apex class with Product Attribute Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IProductAttributeCallback* interface:

Method	Signature	Description
finish()	*void finish()*	This the last method that is executed after the callback is invoked.
setDefaults()	*void setDefaults(Apttus_Config2.LineItem, Apttus_Config2__ProductAttributeValue__c)*	You can use this method to set the default value of product attributes. This method is invoked only when you enable **Has Attribute** field for the product.
start()	*void start(Apttus_Config2.ProductConfiguration)*	This is the first method that is executed when the callback is invoked.

⚠ You must enable **Has Attribute** field in product details. Otherwise, when the callback is invoked, only the **start()** and **finish()** methods are called and the callback does not set the default values for product attributes.

Example

In the following example, the Product Attribute Callback sets the default attributes of the Car object.

```
/*
* Product Attribute Call Back example.
*
* Configure a proposal in test org, which has PriceList = 'CPQ 202'
* Configure Product = 'Car',
* and verify the default attributes
* prodAttr.Exterior_Color__c = 'Silver'
* prodAttr.Leather_Color__c = 'Brown'
* prodAttr.Inlay__c = 'Walnut'
* prodAttr.Fuel_Type__c = 'Diesel'
* prodAttr.Fuel_Badge__c = 'No'
*/
```

```
global with sharing class SampleProductAttributeCallback implements
 Apttus_Config2.CustomClass.IProductAttributeCallback {

    Id accountId;

    global void start(Apttus_Config2.ProductConfiguration config) {
        system.debug('***ProdAttrCallBack > start() > ' + config);

    }

    global void setDefaults(Apttus_Config2.LineItem lineItem,
Apttus_Config2__ProductAttributeValue__c prodAttr) {
        system.debug('***ProdAttrCallBack > setDefaults() > prodAttr:' + prodAttr +
';<>; lineItem:' + lineItem);

        // lineItem is a wrapper class to the lineItem class which we use in product
configuration.
        // Take line Item from this wrapper class.
        Apttus_Config2__LineItem__c prodLineItem = lineItem.getLineItemSO();

        system.debug('***ProdAttrCallBack > Product Selected: ' +
prodLineItem.Apttus_Config2__Description__c);

        if(prodLineItem.Apttus_Config2__Description__c == '(Demo) Car') {
            prodAttr.Exterior_Color__c = 'Silver';
            prodAttr.Leather_Color__c = 'Brown';
            prodAttr.Inlay__c = 'Walnut';
            prodAttr.Fuel_Type__c = 'Diesel';
            prodAttr.Fuel_Badge__c = 'No';
        }

        if(prodLineItem.Apttus_Config2__Description__c == 'Internet Service') {
            prodAttr.Internet_Speed__c = '25 Mbps';
        }
    }

    global void finish() {
        system.debug('***ProdAttrCallBack > finish()');
    }
}
```

Related Pricing Callback Class

Related Pricing Callback allows you to apply custom logic to calculate the related pricing. This callback is invoked while CPQ is calculating related pricing after the main pricing is calculated.

To use the Related Pricing Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IRelatedPricingCallback* interface and register the custom apex class with Related Pricing Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IRelatedPricingCallback* interface:

Method	Signature	Description
computeBasePrice()	*Apttus_Config2.CustomClass.RelatedPriceResult computeBasePrice(Apttus_Config2.ProductConfiguration, Apttus_Config2.LineItem, List)*	You can use this method to apply custom logic for related pricing calculation and override the default CPQ related pricing logic.

If you have defined Pricing Callback class, Related Pricing Callback is always invoked after the Pricing Callback is executed.

Example

In the following sample code, Related Pricing Callback to calculates the price on related line items.

```
global with sharing class APTS_RelatedPricingCallBack implements
 Apttus_Config2.CustomClass.IRelatedPricingCallback {
    /**
        * Callback to compute the related price for the related line item
        * @param cart the cart object associated with the related line item
        * @param relLineItemMO the related line item to compute the base price for
        * @param breakups the list of price breakup sobjects associated with the
related line item
        * @return the related price result
    */
```

```
    public Apttus_Config2.CustomClass.RelatedPriceResult
computeBasePrice(Apttus_Config2.ProductConfiguration cart, Apttus_Config2.LineItem
relLineItemMO, List<Apttus_Config2__PriceBreakup__c> breakups)
    {
        Apttus_Config2.CustomClass.RelatedPriceResult relPriceResult = new
 Apttus_Config2.CustomClass.RelatedPriceResult();

        //Apttus_Config2__ProductConfiguration__c cartSO =  cart.getConfigSO();
        Apttus_Config2__LineItem__c liSO = relLineItemMO.getLineItemSO();
        system.debug('Net Price-->' + liSO.Apttus_Config2__NetPrice__c + '
ChargeType-->' + liSO.Apttus_Config2__ChargeType__c + ' Base Price-->' +
liSO.Apttus_Config2__BasePrice__c);
        system.debug('breakups-->' + breakups);

        Decimal softwareFeesNetprice = 0;
        if(breakups != null)
        {
            for(Apttus_Config2__PriceBreakup__c b : breakups)
            {
                if(b.Apttus_Config2__BreakupType__c == 'Total' &&
liSO.Apttus_Config2__ChargeType__c == 'Maintenance Fee')
                {
                    softwareFeesNetprice  += b.Apttus_Config2__RelatedNetPrice__c;
                }
            }
        }

        system.debug('softwareFeesNetprice : ' +softwareFeesNetprice+ '
liSO.Maintenance_Level_Value__c ' + liSO.Maintenance_Level_Value__c);
        if(liSO.Apttus_Config2__ChargeType__c == 'Maintenance Fee' &&
liSO.Maintenance_Level_Value__c != null)
        {
            relPriceResult.BasePrice = (softwareFeesNetprice *
liSO.Maintenance_Level_Value__c)/100;
        }
        system.debug('relPriceResult.BasePrice : ' +relPriceResult.BasePrice);
        system.debug('##-- EXIT RELATED PRICING CALLBACK');
        return relPriceResult;
    }
}
```

Asset Line Item Callback Class

Asset Line Item Callback provides you a mechanism to define the criteria based on which you want to filter assets on the Installed Product page. The asset line items on the Installed product page are shown according to the business logic defined in the Asset Line Item Callback class. You can further filter the asset line items on the Installed Products page using the filters on the page. You can also use this callback to validate the termination of an asset on the Asset Termination page.

To use the Asset Line Item Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IAssetLineItemCallback4* interface and register the custom apex class with Asset Line Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IAssetLineItemCallback4* interface:

Method	Signature	Description
finish()	*void finish()*	This the last method that is executed after the callback is invoked.
getAssetSearchScope()	*List getAssetSearchScope()*	You can use this method to return the asset search scope that was set in the start method.
getFilterExpr()	*String getFilterExpr(Apttus_Config2.CustomClass.ActionParams)*	You can use this method to set the search scope.
getQueryFilter()	*String getQueryFilter(Id)*	You can use this method to create and return the query filter.
start()	*void start(Apttus_Config2.ProductConfiguration, String, List)*	This is the first method that is executed when the callback is invoked.
validateAssetTermination()	*Boolean validateAssetTermination(Set, Set, Date)*	You can use this method to validate the termination of asset.

Method	Signature	Description
getAssetTerminationDate()	*Date getAssetTerminationDate()*	You can use this method to fetch the termination date of asset.

⚠ You must update CPQ to the latest version in the Apex class that you create to implement the *Apttus_Config2.CustomClass.IAssetLineItemCallback4* interface. Otherwise, any logic to default date is not executed. Follow the steps below to update the version setting.

1. Go to **Setup** > **Develop** > **Apex Classes**.
2. Open the Apex class you created to implement the Asset Line Item Callback.
3. Click **Edit**. Click **Version Settings**.
4. Find the **Conga Configuration & Pricing** package. In the **Version** dropdown next to the package select the latest version of CPQ. For details about latest packages, refer to *CPQ on Salesforce Release Notes.*
5. Click **Save**.

Examples

You can use Asset Filter Callback in the following scenarios. You can modify the code based on your requirements.

- Defaulting date and custom validation.
- Retrieving assets from multiple accounts.

The following sample code enables you to set a default date and define custom validation.

```
private Apttus_Config2.CustomClass.IAssetLineItemCallback4 callback = null;
private transient Apttus_Config2.ProductConfiguration cart = null;
private List<String> assetSearchScope = null;
private String assetSearchFilter = null;


/**
 * Callback at the beginning of the asset selection call.
 * Use the start method to initialize state
 * @param cart the cart object or null if there is no cart context
 * @param assetSearchFilter the preset static filter used in the asset search or null
if there is no preset filter
 * @param assetSearchScope the list of asset fields to match the search text or empty
to use the default list of fields
```

```
 */
global void start(Apttus_Config2.ProductConfiguration cart, String assetSearchFilter,
List<String> assetSearchScope)
{
    // delegate to the asset callback
    this.cart = cart;
    this.assetSearchFilter = assetSearchFilter;
    this.assetSearchScope = assetSearchScope;
}

global Date getAssetTerminationDate()
{
    return Date.newInstance(2018,01,01);
}

/**
 * Callback to return part of SOQL filter clause
 * This filter is used in listing installed products
 * @param accountId is the context account id
 * @return The query filter is like the following.
 *      Name LIKE 'A%' AND Quantity__c > 100
 *      Id IN ('000123', '000124')
 */
global String getQueryFilter(ID accountId)
{
    // delegate to the asset callback
    system.debug ('*************** AssetLineItemCallbackAdapter:' + accountId);
    return null;

}

global Boolean validateAssetTermination(Set<ID> assetIds, Set<ID> accountIds, Date
eDate)
{
    String nsPrefix = 'Apttus_Config2__';
    Apttus_Config2.CPQStruct.QueryAssetsRequestDO request = new
 Apttus_Config2.CPQStruct.QueryAssetsRequestDO();
    request.AccountIds = new List<Id>(accountIds);
    system.debug ('accountids' + accountIds);
    request.FieldNames = new List<String>();
    request.FieldNames.add(nsPrefix + 'StartDate__c');
    request.FieldNames.add(nsPrefix + 'EndDate__c');
    request.FieldNames.add( 'Name');
```

```
    //added new fields
    request.FieldNames.add(nsPrefix + 'BusinessObjectId__c');
    request.FieldNames.add(nsPrefix + 'ProductId__c');
    request.FieldNames.add(nsPrefix + 'LineNumber__c');
    request.FieldNames.add(nsPrefix + 'PrimaryLineNumber__c');
    request.FieldNames.add(nsPrefix + 'IsPrimaryRampLine__c');
    request.FieldNames.add(nsPrefix + 'IsPrimaryLine__c');
    request.FieldNames.add(nsPrefix +  'PriceGroup__c');
    request.FieldNames.add(nsPrefix +  'LineType__c');
    request.FieldNames.add(nsPrefix +  'ChargeType__c');
    request.FieldNames.add(nsPrefix + 'CurrentContractStartDate__c');

    if (assetIds != null && assetIds.size() >0)
    {
        request.customFilter =  'Id IN  (';
        for (Id assetId : assetIds)
        {
            request.customFilter +=  '\'' + assetId + '\',';
            system.debug ('assetids: ' + assetId);
        }
        request.customFilter = request.customFilter.removeEnd(',');
        request.customFilter += ')';
        system.debug ('request.customFilter ' + request.customFilter);
    }

    Apttus_Config2.CPQStruct.QueryAssetsResponseDO assetResponseDO =
Apttus_Config2.AssetService.getAssetLineItems(request);

    for (Apttus_Config2__AssetLineItem__c asset: assetResponseDO.AssetLineItems)
    {
        if (asset.Apttus_Config2__IsPrimaryRampLine__c &&
asset.Apttus_Config2__IsPrimaryLine__c)
        {
            Apttus_Config2__AssetLineItem__c farthestLineItem =
findFarthestRampLine(asset, assetResponseDO.AssetLineItems);
            if (eDate > farthestLineItem.Apttus_Config2__EndDate__c)
            {
                ApexPages.Message message = new
 ApexPages.Message(ApexPages.Severity.ERROR, 'Termination date : '+eDate + 'should
not be greater than Ramp Farthest Asset End date: ' +
farthestLineItem.Apttus_Config2__EndDate__c );
                ApexPages.addMessage(message);
                return false;
```

```
            }
        }
        else if (asset.Apttus_Config2__IsPrimaryLine__c &&
asset.Apttus_Config2__LineType__c == 'Product/Service')
        {
            if (eDate >= asset.Apttus_Config2__EndDate__c)
            {
                ApexPages.Message message = new
 ApexPages.Message(ApexPages.Severity.ERROR, 'Termination date should not greater
than Asset End Date :Effective Date is ' + eDate );
                ApexPages.addMessage(message);
                return false;

            }
            else if (eDate < asset.Apttus_Config2__StartDate__c)
            {
                ApexPages.Message message = new
 ApexPages.Message(ApexPages.Severity.ERROR, 'Invalid Termination date');
                ApexPages.addMessage(message);
                return false;
            }
        }

       /* if (eDate < asset.Apttus_Config2__CurrentContractStartDate__c) {
            ApexPages.Message message = new
ApexPages.Message(ApexPages.Severity.ERROR, 'Asset cannot be terminated before
Contract start date');
            ApexPages.addMessage(message);
            return false;
        } */
    }
    return true;
}

/**
 * Callback to return the filter expression for the asset query where clause
 * This filter is used in listing installed products
 * @param params the parameters for the method
 * @return the filter expression or null to use the default filter.
 * e.g. Name LIKE 'A%' AND Quantity__c > 100
 *      Id IN ('000123', '000124')
 */
global String getFilterExpr(Apttus_Config2.CustomClass.ActionParams params)
```

```
{
   // dispatch based on the location id parameter
    String clause = '';
    Boolean locationFound = false;
    Boolean accountFound = false;
    Boolean customParamsExist = false;
    String ns = 'Apttus_Config2__';
    List<String> fieldNames = new List<String>();
    fieldNames.add(ns + 'AccountId__c');

    if(params.CustomParams != null && params.CustomParams.size() > 0)
    {
        customParamsExist = true;
        Boolean isFirst = true;
        for(String fieldName: params.CustomParams.keySet())
        {
            if (isFirst)
            {
                isFirst = false;
            }
            else
            {
                clause += ' AND ';
            }

            if (fieldName.toLowerCase().contains('fromdate'))
            {
                clause += fieldName.substring(0, fieldName.toLowerCase().indexOf('fro
mdate')) + ' >= ' + params.CustomParams.get(fieldName);

            }
            else if (fieldName.toLowerCase().contains('todate'))
            {
                clause += fieldName.substring(0, fieldName.toLowerCase().indexOf('tod
ate')) + ' <= ' + params.CustomParams.get(fieldName);
            }
            else
            {
                clause += fieldName + ' IN (' + params.CustomParams.get(fieldName) +
')';
            }
        }
    }
    if(params.LocationIds != null && !params.LocationIds.isEmpty())
    {
```

```
        if (customParamsExist)
        {
            clause += ' AND ';
        }
        Boolean isFirst = true;
        for (ID Id : params.LocationIds)
        {
            if(isFirst)
            {
                isFirst = false;
                locationFound = true;
                clause += ns + 'LocationId__c IN (\'' + Id;
            }
            else
            {
                clause += '\', \'' + Id;
            }
        }
        if (locationFound)
        {
            clause += '\')';
        }
    }

    if(params.AccountIds != null && !params.AccountIds.isEmpty())
    {
        Boolean isFirstSource = true;
        for (String accountAPIName : fieldNames)
        {
            if (isFirstSource)
            {
                isFirstSource = false;
            }
            else
            {
                clause += ' OR ';
            }
            Boolean isFirst = true;
            for (ID Id : params.AccountIds)
            {
                if(isFirst)
                {
                    if (locationFound || customParamsExist)
```

```
                    {
                        clause += ' AND ';
                    }
                    isFirst = false;
                    accountFound = true;
                    clause += accountAPIName + ' IN (\'' + Id;
                }
                else
                {
                    clause += '\', \'' + Id;
                }
            }
            if (accountFound)
            {
                clause += '\')';
            }
        }
    }
    boolean isShowMultipleAccountsEnabled = false;
    //when account filter is not displayed and static asset filter is not present
user cart account id

    if (!accountFound && !isShowMultipleAccountsEnabled && (assetSearchFilter == null
 || assetSearchFilter.trim().length() == 0) )
    {
        if (locationFound || customParamsExist)
        {
            clause += ' AND ';
        }
        accountFound = true;
        Boolean isFirstSource = true;
        for (String accountAPIName : fieldNames)
        {
            if (isFirstSource)
            {
                isFirstSource = false;
                clause += accountAPIName + ' = \'' + params.AccountId + '\' ';
            }
            else
            {
                clause += ' OR ' + accountAPIName + ' = \'' + params.AccountId + '\'
';
            }
        }
    }
```

```
    //append static asset filter
    if (locationFound == true || accountFound == true || customParamsExist == true)
    {
        if (!(assetSearchFilter == null || assetSearchFilter.trim().length() == 0))
        {
            return clause  +  ' AND '  + assetSearchFilter;
        }
        return clause ;
    }
    // account id is also available in the cart
    if (!(assetSearchFilter == null || assetSearchFilter.trim().length() == 0))
    {
        return assetSearchFilter;
    }

    // the sample query filter
    String qryStr = '';
    Boolean isFirstSource = true;
    for (String accountAPIName : fieldNames)
    {
        if (isFirstSource)
        {
            isFirstSource = false;
            qryStr += accountAPIName + ' = \'' + params.accountId + '\' ';
        }
        else
        {
            qryStr += ' OR ' + accountAPIName + ' = \'' + params.accountId + '\' ';
        }
    }
    system.debug ('*************** qryStr'+ qryStr);
    return qryStr;
}

/**
 * Gets the asset search scope
 * @return thr asset search scope or null to use the default asset search scope
 */
global List<String> getAssetSearchScope()
{
    // delegate to the asset callback
    return null;
}
```

```
/**
 * Callback after the filter is used
 * Use the finish method to release state
 */
global void finish()
{
    // delegate to the asset callback
}

private static Apttus_Config2__AssetLineItem__c
findFarthestRampLine(Apttus_Config2__AssetLineItem__c primaryRampLine,
List<Apttus_Config2__AssetLineItem__c> assetLineItems)
{
   Apttus_Config2__AssetLineItem__c farthestAssetLineItem = primaryRampLine;
   if (assetLineItems != null && !assetLineItems.isEmpty()) {
        for (Apttus_Config2__AssetLineItem__c assetLineItem : assetLineItems)
        {
        if (assetLineItem.Apttus_Config2__PriceGroup__c == 'Price Ramp'
            && assetLineItem.Apttus_Config2__BusinessObjectId__c ==
primaryRampLine.Apttus_Config2__BusinessObjectId__c
            && assetLineItem.Apttus_Config2__ProductId__c ==
primaryRampLine.Apttus_Config2__ProductId__c
            && assetLineItem.Apttus_Config2__LineNumber__c.intValue() ==
primaryRampLine.Apttus_Config2__LineNumber__c.intValue()
            && (primaryRampLine.Apttus_Config2__PrimaryLineNumber__c == null
                || (assetLineItem.Apttus_Config2__PrimaryLineNumber__c != null
                    && assetLineItem.Apttus_Config2__PrimaryLineNumber__c.intValue()
== primaryRampLine.Apttus_Config2__PrimaryLineNumber__c.intValue()))
            && assetLineItem.Apttus_Config2__IsPrimaryRampLine__c == false
            && assetLineItem.Apttus_Config2__EndDate__c >
farthestAssetLineItem.Apttus_Config2__EndDate__c)
            {
                farthestAssetLineItem = assetLineItem;
            }
        }
    }
    return farthestAssetLineItem;
}
```

The following sample code enables you to fetch assets from multiple accounts.

```
/*
```

```
 * This class is used in Asset Line Item callback to show asset on Cart page.
 */
global with sharing class CurrentUserFilter_AssetLineItemCallback implements
 Apttus_Config2.CustomClass.IAssetLineItemCallback4
    {

    private List<String> assetSearchScope = null;
    private String assetSearchFilter = null;
    private ID userId = null;

    /**
     * Callback at the beginning of the asset selection call.
     * Use the start method to initialize state
     * @param cart the cart object or null if there is no cart context
     * @param assetSearchFilter the preset static filter used in the asset search or
null if there is no preset filter
     * @param assetSearchScope the list of asset fields to match the search text or
empty to use the default list of fields
     */
    global void start(Apttus_Config2.ProductConfiguration cart, String
assetSearchFilter, List<String> assetSearchScope) {
        this.userId = cart.getConfigSO().CreatedById;
    }

    /**
     * Callback to return part of SOQL filter clause
     * This filter is used in listing installed products
     * @param accountId is the context account id
     * @return The query filter is like the following.
     *         Name LIKE 'A%' AND Quantity__c > 100
     *         Id IN ('000123', '000124')
     */
    global String getQueryFilter(ID accountId) {
        // all Asset Lines created by the current user
        return 'CreatedById = \'' + userId + '\' AND CreatedDate > LAST_MONTH';
    }

    global Boolean validateAssetTermination(Set<ID> assetIds, Set<ID> accountIds,
Date eDate) {
        return true;
    }

    global Date getAssetTerminationDate() {
```

```
        return Date.newInstance(2018,01,01);
    }

    /**
     * Callback to return the filter expression for the asset query where clause
     * This filter is used in listing installed products
     * @param params the parameters for the method
     * @return the filter expression or null to use the default filter.
     * e.g. Name LIKE 'A%' AND Quantity__c > 100
     *         Id IN ('000123', '000124')
     */
    global String getFilterExpr(Apttus_Config2.CustomClass.ActionParams params) {
        return getQueryFilter(null);
    }

    /**
     * Gets the asset search scope
     * @return the asset search scope or null to use the default asset search scope
     */
    global List<String> getAssetSearchScope(){
        return this.assetSearchScope;
    }

    /**
     * Callback after the filter is used
     * Use the finish method to release state
     */
    global void finish() {

    }
}
```

Asset Renewal Custom Callback Class

Asset Renewal Custom Callback allows you to filter the renewal quote assets that are created using the On-Demand job. For example, you can filter assets by **Product Name** as a filter criteria. You can also use this callback to populate a selective set of parameters with corresponding values on a renewal quote (with the same data as the original quote) during renewal automation. If the user has data on the quote header that must be propagated to line items and then be used in pricing, this callback can copy those fields to the renewal quote and pricing will be accurate on the renewal quote generated.

To use the Asset Renewal Custom Callback, you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IAssetRenewalCustomCallback* interface and register the custom apex class with Asset Renewal Custom Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IAssetRenewalCustomCallback* interface:

Method	Signature	Description
getAssetFilterExpr()	*String getAssetFilterExpr()*	You must use this method to append custom filters to the on-demand asset query when the asset line items are fetched from the on-demand renewal quote. If there are no additional filters (other than the standard filters), return null.
getConfigProperties()	*List getConfigProperties(SObject)*	You must use this method to fetch the configuration request parameters when the renewal cart is created. You can also use this method to update the proposal or agreement with default values by passing a proposal or agreement subject.

Example

In the following sample, the assets on the renewal quote are filtered based on the product name.

```
global with sharing class SampleAssetRenewalCustomCallback implements
 Apttus_Config2.CustomClass.IAssetRenewalCustomCallback {

    /**
     * Callback invoked to append asset filter expression to the renew asset line
item query
     * Filter expression should be created using asset line item fields only.
     * @return the filter expression or null to use the default filter.
     */

    global String getAssetFilterExpr() {

        Product2 prod = [SELECT ID from Product2 where Name = 'Standalone2'];
```

```
        return 'Apttus_Config2__ProductId__c = \'' + prod.Id + '\'';

    }

    /**
     * Gets the list of configuration properties for the given business SObject
     * @param bObjectSO the business Sobject to get the configuration properties for
     * @return the list of configuration properties.
     */

    global List<Apttus_Config2.Property> getConfigProperties(SObject bObjectSO) {

        List<Apttus_Config2.Property> configProps = new
 List<Apttus_Config2.Property>();
        configProps.add(new Apttus_Config2.Property('flow', 'LAngFlow'));
        configProps.add(new Apttus_Config2.Property('useAdvancedApproval', 'false'));
        configProps.add(new Apttus_Config2.Property('useDealOptimizer', 'false'));
        return configProps;
    }
}
```

Revalidation Callback Class

Revalidation Callback provides you a mechanism to implement custom revalidations for products based on custom criteria. You can define custom criteria based on which the products are identified by CPQ for revalidation. The callback allows you to define a custom logic to be applied on the line item along with revalidation. You can also define a custom product version when the line item is created.

To use the Revalidation Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IRevalidationCallback* interface and register the custom apex class with Revalidation Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IRevalidationCallback* interface:

Method	Signature	Description
getRevalidations()	*Apttus_Config2.CustomClass.RevalidationResponse getRevalidations(Apttus_Config2.CustomClass.RevalidationRequest)*	You can use this method to determine whether the line item requires revalidation based on a custom logic that you can define. This method is invoked when you reconfigure a save or finalized cart.
setRevalidations()	*void setRevalidations(Apttus_Config2.CustomClass.RevalidationRequest)*	You can use this method write custom logic to apply on line items. This method is invoked when you click **Apply Revalidation** on the Revalidation pop-up.
setVersionNumber()	*void setVersionNumber(Id, List)*	You can use this method to set the version number field of the line item with custom value based on custom logic you define. This method is invoked when you add line items to the cart.

Example

You can use this callback to a scenario like:

- You can have multiple versions for a single product record. This enables you to revalidate a specific version of the product as necessary.
  For example, for product record, you can set the version as X, Y, and any such value for the newly created line item of that product. You can determine which line item qualifies for revalidation and apply custom logic on the line items accordingly.
- You can apply validation based custom fields other than the **Product Version**.
- You can validate changes in pricing fields like **List price** of a given price list item.

The below-given code example checks if the quantity of line items is greater then 10 then triggers the revalidation on the cart, revalidates the line item and resets the quantity of the line items.

```
/**
 *  Apttus Config & Pricing
 *  SampleRevalidationCallback
 *
 *  @2019-2020 Apttus Inc. All rights reserved.
```

```
 */
global with sharing class SampleRevalidationCallback implements
 Apttus_Config2.CustomClass.IRevalidationCallback {
    private static String QUANTITY_TOO_HIGH = 'Quantity Too High';
    /**
     * Callback at the beginning of revalidation check.
     * use this method to get custom revalidations
     * @param request contains lineItemSO and validation info
     * @return map of revalidation category against list of line numbers
     */
    global Apttus_Config2.CustomClass.RevalidationResponse
getRevalidations(Apttus_Config2.CustomClass.RevalidationRequest request){
        Apttus_Config2.CustomClass.RevalidationResponse response = new
 Apttus_Config2.CustomClass.RevalidationResponse();
        Map<String, List<Apttus_Config2__LineItem__c>> lineItemSOsByRevalCategory =
new Map<String, List<Apttus_Config2__LineItem__c>>();

        for (String revalCat : request.lineItemSOsByRevalCategory.keySet()) {
            lineItemSOsByRevalCategory.put(revalCat,
request.lineItemSOsByRevalCategory.get(revalCat));
            for (Apttus_Config2__LineItem__c lineItemSO :
request.lineItemSOsByRevalCategory.get(revalCat)) {
                response.isRevalidationNeeded = true;
            }
        }

        response.lineItemSOsByRevalCategory = lineItemSOsByRevalCategory;
        response.isHardRevalidationNeeded = false;
        List<Apttus_Config2__LineItem__c> customQuantityLines = new
 List<Apttus_Config2__LineItem__c>();

        // Implementing list price change use case
        for (Apttus_Config2__LineItem__c lineItemSO : [select Id,
Apttus_Config2__Quantity__c,
Apttus_Config2__PriceListItemId__r.Apttus_Config2__ListPrice__c,
Apttus_Config2__PrimaryLineNumber__c from Apttus_Config2__LineItem__c where
Apttus_Config2__ConfigurationId__c = :request.cartId]) {
            if(lineItemSO.Apttus_Config2__Quantity__c > 10) {
                customQuantityLines.add(lineItemSO);
            }
        }
        if (customQuantityLines.size() > 0) {
            response.lineItemSOsByRevalCategory.put(QUANTITY_TOO_HIGH,
customQuantityLines);
            response.isRevalidationNeeded = true;
```

```
        }
        return response;
    }

    /**
     * Callback for applying revalidation
     * @param  cartId Id of the cart to revalidate
     * @param  lineNumbers the list of line numbers to be validated
     */
    global void setRevalidations(Apttus_Config2.CustomClass.RevalidationRequest
request) {
        List<Apttus_Config2__LineItem__c> lineItemsSOsToUpdate = new
 List<Apttus_Config2__LineItem__c>();
        for (String revalCategory : request.lineItemSOsByRevalCategory.keySet()) {
            for (Apttus_Config2__LineItem__c lineItemSO :
request.lineItemSOsByRevalCategory.get(revalCategory)) {

                if (revalCategory == QUANTITY_TOO_HIGH) {
                    lineItemSO.Apttus_Config2__Quantity__c = 1.0;
                    lineItemSO.Apttus_Config2__PricingStatus__c = 'Pending';
                    lineItemsSOsToUpdate.add(lineItemSO);
                }
            }
        }
        System.debug('**'+lineItemsSOsToUpdate);
        if (lineItemsSOsToUpdate.size() > 0) {
            System.debug('>>'+lineItemsSOsToUpdate);
            update lineItemsSOsToUpdate;
        }
    }

    /**
     * Callback to set custom version number
     * @param  cartId Id of the cart
     * @param  lineIntemSOs the list of line Item sobjects to set custom version
     */
    global void setVersionNumber(ID cartId, List<Apttus_Config2__LineItem__c>
lineItemSOs){
        /*for (Apttus_Config2__LineItem__c lineItemSO : lineItemSOs) {
            lineItemSO.Apttus_Config2__ProductVersion__c = 33.0;
        }*/
    }
}
```

Action Params Callback Class

Action Params Callback provides you a mechanism to create action parameters for a cart object. You can use the action parameters created using the callback in the following actions in CPQ:

- Submitting the cart for approval
- Generating document
- Analyzing the deal
- Retrieving Advanced Approval URL for selected line item

To use the Action Params Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IActionParamsCallback* interface and register the custom apex class with Action Params Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IActionParamsCallback* interface.

Method	Signature	Description
createActionParams	*Apttus_Config2.CustomClass.ActionParams createActionParams(Apttus_Config2.ProductConfiguration)*	You can use this method to create action parameters for the cart object. This method is invoked when you: • Submit the cart for approval • Retrieve Advanced Approval URL for selected line item • Perform the analyze deal action • Perform the generate doc action

The following table lists the response parameters of the *Apttus_Config2.CustomClass.IActionParamsCallback.*

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
AccountId	Id	The ID of the Account.

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
AccountIds	Set<Id>	The set of Account ID you want to use to filter the result. CPQ ignores the *AccountId* field if this field is used.
ActionName	String	The name of the action.
ActivateOrder	Boolean	Indicates whether to activate the order or not.
ApprovalCtxType	String	The Context Type of Approval.
ApprovalMode	String	The Mode of Approval
ApprovalReason	String	The reason of Approval.
ApprovalType	String	The type of Approval.
AssetLineItems	List<Apttus_Config2__AssetLineItem__c>	The list of asset line items
BundleId	Id	The IDs of the bundle product.
CartIds	List<IDs>	The list of cart IDs.
CartSyncMode	Apttus_Config2.CustomClass.SyncMode	The Sync Mode of the cart. For example, *enum SyncMode {SYNC, FINALIZE}*.
ConfigurationId	Id	The ID of the configuration in the proposal.
CurrentState	String	The current state
CustomParams	Map<String, String>	The custom parameter that you have defined.
FinalizeClass	String	The callback class that is executed when you finalize the cart.

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
Flow	String	The flow used in the configuration
IsAngular	Boolean	Indicates whether the user interface is Angular.
IsDraft	Boolean	Indicated whether the proposal is in draft stage.
LaunchState	String	The launch state.
LineItemIds	Set<Id>	The set of line item IDs.
LineNumber	Decimal	The line number of the products.
LocationIds	Set<Id>	The set of location IDs.
Method	String	The method of configuration.
Mode	String	The mode of the configuration.
OrderLineItems	List<Apttus_Config2__OrderLineItem__c>	The list of order line items.
OriginalOrderSO	Apttus_Config2__Order__c	The Original Order SObject
OutputFormat	String	The output format of the generated document. You can use the following values: • *PDF* • *DOC* • *RTF*
ProductIDs	List<IDs>	The list of product IDs.
ProtectionLevel	String	The protection level for the parameters you created.

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
RequestId	Id	The request ID of the configuration.
ReturnId	Id	The record ID of the business object.
ReturnPage	String	The return page of the business object.
SessionId	String	The ID of the session.
SessionUrl	String	The URL of the session.
TemplateName	String	The name of the template in the proposal.

Example

The following sample code enables you to create parameters.

```
global with sharing class DefaultActionParamsCallback implements
 Apttus_Config2.CustomClass.IActionParamsCallback
{
    /**
     * Callback to create action parameters for the given cart object
     * @param cart the cart object
     * @return the action parameters object
     */
    global Apttus_Config2.CustomClass.ActionParams
createActionParams(Apttus_Config2.ProductConfiguration cart)
    {

        // create a test configuration request
        Apttus_Config2.TempObject__c requestSO = new
 ConfigRequestTest().createTempObject('Quote');

        // action parameters
        Apttus_Config2.CustomClass.ActionParams params = new
 Apttus_Config2.CustomClass.ActionParams();

        params.ActionName = 'CustomAction';
        params.ApprovalType = Apttus_Config2.CustomClass.APPROVALTYPE_STANDARD;
```

```
        params.ApprovalCtxType = Apttus_Config2.CustomClass.APPROVALCTX_HEADER;
        params.ApprovalMode = Apttus_Config2.CustomClass.APPROVALMODE_PREVIEW;
        params.ApprovalReason = 'Request for Approval';
        params.RequestId = requestSO.Id;
        params.Method = null;
        params.Mode = null;
        params.Flow = 'Default';
        params.LineNumber = 1;
        params.TemplateName = 'Test Template';
        params.ProtectionLevel = 'Read only';
        params.OutputFormat = 'PDF';
        params.IsDraft = true;
        params.SessionId = UserInfo.getSessionId();
        params.SessionUrl = 'https://login.salesforce.com'; // Use https://
test.salesforce.com for sandbox.
        params.ReturnId = '01tA0000001qEcJ'; // The id of the bussiness object record
        params.ReturnPage = 'MyCustomPage';
        params.CustomParams.put('CartId', cart.getId());

        return params;

    }
}
```

Action Invoker Callback Class

Action Invoker Callback provides you a mechanism to implement custom navigation for custom actions. You can define logic for a custom page to redirect the Sales rep to.

To use the Action Invoker Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IActionInvokerCallback* interface and register the custom apex class with Action Invoker Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IActionInvokerCallback* interface.

Method	Signature	Description
invokeAction	*System.PageReference invokeAction(Apttus_Config2.ProductConfiguration, Apttus_Config2.CustomClass.ActionParams)*	You can use this method to execute the custom navigation upon the Sales rep's click of the custom action.

Example

The following sample code is a skeletal structure of custom callback class that implements the *IActionInvokerCallback* interface. You can write your own logic and return navigation for the custom action defined in Display Action Setting for the specific flow.

```
global with sharing class DefaultActionInvokerCallback implements
 Apttus_Config2.CustomClass.IActionInvokerCallback {
   /**
    * Callback to invoke the custom action
    * @param cart the cart object
    * @param params the parameters for the action
    * @return the result page reference or null if there is no page
    */
     global ApexPages.PageReference invokeAction(ProductConfiguration cart,
Apttus_Config2.CustomClass.ActionParams params) {
            // do nothing
            return null;
        }
}
```

Action Callback Class

Action Callback provides you a mechanism to apply custom actions on different actions available on the cart. For example, you can apply custom action before deleting or copying a bundle, finalizing a cart, submitting a cart for approval. You can directly use methods of the callback to apply the custom action or use Web Services to implement the callback.

To use the Action Callback you must create a custom Apex class that implements one of the following interfaces and you must register the custom class with Action Callback Class. You must write your custom logic in the custom apex class.

Interface	Description
IActionCallback2	This interface provides a mechanism to implement custom actions on cart finalization, submission for approval using Web Services.
IActionCallback3	This interface provides a mechanism to implement custom actions before delete and copy actions on bundles.

This section describes the interfaces of Action Callback:

- IActionCallback2 Interface
- IActionCallback3 Interface

IActionCallback2 Interface

IActionCallback2 class provides a mechanism to apply custom actions on cart finalization, submission for approval using the below Web Services.

- *ActionInvokerWebService.invokeAfterFinalizeCart*
- *ActionInvokerWebService.invokeSyncCart*
- *ActionInvokerWebService.invokeSyncCartAsync*
- *ActionInvokerWebService.invokeFinalizeCart*
- *ActionInvokerWebService.invokeFinalizeCartAsync*
- *ActionInvokerWebService.invokeAfterSyncCart*
- *ActionInvokerWebService.invokeAfterSyncCartAsync*
- *ActionInvokerWebService.invokeAfterFinalizeCartAsync*
- *ActionInvokerWebService.invokeSubmitApproval*
- *ActionInvokerWebService.invokeGenerateDoc*

Action callback executes the custom logic from Web Services on the following aspects of CPQ:

- Finalizing the cart
- Synchronizing the cart
- Submitting the cart for approval
- Generating documents

The following methods are available in the *Apttus_Config2.CustomClass.IActionCallback2* interface:

Method	Signature	Description
afterFinalizeCart()	*Boolean afterFinalizeCart(Id)*	You can use this method to perform post-finalization tasks on the cart. This method is invoked when you execute the following Web Services: • *ActionInvokerWebService.invokeAfterFinalizeCart()* • *ActionInvokerWebService.invokeAfterFinalizeCartAsync()*
finalizeCart()	*Boolean finalizeCart(Id)*	You can use this method to finalize the cart. This method is invoked when you execute the following Web Services: • *ActionInvokerWebService.invokeFinalizeCart()*
generateDoc()	*Id generateDoc(Id, Apttus_Config2.CustomClass.ActionParams)*	You can use this method to generate documents for the given cart. This method is invoked when you generate documented the following Web Service: • *ActionInvokerWebService.invokeGenerateDoc()*
submitApproval()	*Boolean submitApproval(Id, Apttus_Config2.CustomClass.ActionParams)*	You can use this method to submit the cart for approval. This method is invoked when you submit the cart for approval using following the Web Service: • *ActionInvokerWebService.invokeSubmitApproval()*

Method	Signature	Description
afterSyncCart()	*Boolean afterSyncCart(Id)*	You can use this method to perform post-synchronization tasks on the cart. This method is invoked when the cart is synchronized using the below Web Services: • *ActionInvokerWebService.invokeSyncCart()* • *ActionInvokerWebService.invokeAfterSyncCartAsync()*
syncCart()	*Boolean syncCart(Id)*	Tou can use this method to perform to synchronize the cart. This method is invoked when the cart is synchronized synchronously or asynchronously using the below Web Services. • *ActionInvokerWebService.invokeSyncCart()* • *ActionInvokerWebService.invokeSyncCartAsync()*

The following table lists the response parameters of the *Apttus_Config2.CustomClass.IActionParamsCallback.*

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
AccountId	Id	The ID of the Account.
AccountIds	Set<Id>	The set of Account ID you want to use to filter the result. CPQ ignores the *AccountId* field if this field is used.
ActionName	String	The name of the action.
ActivateOrder	Boolean	Indicates whether to activate the order or not.

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
ApprovalCtxType	String	The Context Type of Approval.
ApprovalMode	String	The Mode of Approval
ApprovalReason	String	The reason of Approval.
ApprovalType	String	The type of Approval.
AssetLineItems	List<Apttus_Config2__AssetLineItem__c>	The list of asset line items
BundleId	Id	The IDs of the bundle product.
CartIds	List<IDs>	The list of cart IDs.
CartSyncMode	Apttus_Config2.CustomClass.SyncMode	The Sync Mode of the cart. For example, *enum SyncMode {SYNC, FINALIZE}*.
ConfigurationId	Id	The ID of the configuration in the proposal.
CurrentState	String	The current state
CustomParams	Map<String, String>	The custom parameter that you have defined.
FinalizeClass	String	The callback class that is executed when you finalize the cart.
Flow	String	The flow used in the configuration
IsAngular	Boolean	Indicates whether the user interface is Angular.
IsDraft	Boolean	Indicated whether the proposal is in draft stage.

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
LaunchState	String	The launch state.
LineItemIds	Set<Id>	The set of line item IDs.
LineNumber	Decimal	The line number of the products.
LocationIds	Set<Id>	The set of location IDs.
Method	String	The method of configuration.
Mode	String	The mode of the configuration.
OrderLineItems	List<Apttus_Config2__OrderLineItem__c>	The list of order line items.
OriginalOrderSO	Apttus_Config2__Order__c	The Original Order SObject
OutputFormat	String	The output format of the generated document. You can use the following values: • *PDF* • *DOC* • *RTF*
ProductIDs	List<IDs>	The list of product IDs.
ProtectionLevel	String	The protection level for the parameters you created.
RequestId	Id	The request ID of the configuration.
ReturnId	Id	The record ID of the business object.
ReturnPage	String	The return page of the business object.
SessionId	String	The ID of the session.

Response Parameters:Apttus_Config2.CustomClass.ActionParams		
Name	**Type**	**Description**
SessionUrl	String	The URL of the session.
TemplateName	String	The name of the template in the proposal.

Example

The sample code below enables you to send out the status of the cart to the proposal owner as callback action after finalizing the cart. You can perform post-finalization by calling the Web Service *Apttus_Config2. ActionInvokerWebService.invokeAfterFinalizeCart()* passing the custom callback class name and the cart ID as parameters.

```
1  global with sharing class Apttus_ActionCallback implements
    Apttus_Config2.CustomClass.IActionCallback2
2  {
3      global static Boolean finalizeCart(ID cartId)
4      {
5          // callback logic to finalize the cart.  This logic will be
   executed when we call the webservice
   Apttus_Config2.ActionInvokerWebService.invokeFinalizeCart
6          return true;
7      }
8
9      global static Boolean afterFinalizeCart(ID cartId)
10     {
11         // get the owner of the proposal
12         Apttus_Config2__ProductConfiguration__c proposalSO = [SELECT
   Name,Apttus_Config2__Proposald__r.owner.id
13                                                               FROM
   Apttus_Config2__ProductConfiguration__c
14                                                               WHERE Id
   =:cartId LIMIT 1];
15         // get the email ID of the proposal owner
16         Id ownerID = proposalSO.Apttus_Config2__Proposald__r.owner.id;
17         User userSO = [SELECT email FROM User WHERE Id =: ownerID LIMIT 1]
   ;
18
19         // send finalize action
```

```
20          Messaging.SingleEmailMessage mail = new
 Messaging.SingleEmailMessage();
21          mail.setSenderDisplayName('Salesforce Support'); //Sender's
Display name
22          mail.setSaveAsActivity(true);
23          mail.setSubject('Cart - ' + proposalSO.Name + ' Finalized');
24          mail.setPlainTextBody('Please be informed that the cart is
finalized.');
25          mail.setToAddresses(new List <String> {userSO.email});
26          Messaging.sendEmail(new Messaging.SingleEmailmessage[] {mail});
27
28          return true;
29
30      }
31
32      global static Boolean submitApproval(ID cartId,
Apttus_Config2.CustomClass.ActionParams params)
33      {
34          // callback logic to submit the cart for approval. This method
will be executed when we call the WebService
Apttus_Config2.ActionInvokerWebService.invokeSubmitApproval
35          return true;
36      }
37
38      global static ID generateDoc(ID cartId,
Apttus_Config2.CustomClass.ActionParams params)
39      {
40          // callback logic to generate the document for the given the cart.
This method will be executed when we call the webserice
Apttus_Config2.ActionInvokerWebService.invokeGenerateDoc()
41          return null;
42      }
43
44      global static Boolean syncCart(ID cartId)
45      {
46          // callback logic to synchronize the cart for the given cartId.
This method will be executed when we call the webserice
Apttus_Config2.ActionInvokerWebService.invokeSyncCart or
Apttus_Config2.ActionInvokerWebService.invokeSyncCartAsync
47          return true;
48      }
49      global static Boolean afterSyncCart(ID cartId)
50      {
51          // callback logic to perform post-synchronization tasks for the
given cartId. This method will be executed when we call the webserice
```

```
        Apttus_Config2.ActionInvokerWebService.invokeAfterSyncCart or
        Apttus_Config2.ActionInvokerWebService.invokeAfterSyncCartAsync
52          return true;
53      }
54  }
```

IActionCallback3 Interface

IActionCallback3 class provides a mechanism to apply custom actions before delete and copy actions on bundles.

The following methods are available in the *Apttus_Config2.CustomClass.IActionCallback3* interface:

Method	Signature	Description
beforeCopyBundleLineItems()	*Apttus_Config2.CustomClass.ActionCallbackResponse beforeCopyBundleLineItems(Apttus_Config2.CustomClass.ActionCallbackRequest)*	You can use this method to perform tasks to be executed before a bundle is copied. This method is invoked when the Sales Rep Clones bundle or standalone product.
beforeDeleteBundleLineItems()	*Apttus_Config2.CustomClass.ActionCallbackResponse beforeDeleteBundleLineItems(Apttus_Config2.CustomClass.ActionCallbackRequest)*	You can use this method to perform tasks to be executed before performing the following actions: • Abandoning the configuration • creating, updating, and deleting line items

Request Parameters: Apttus_Config2.CustomClass.ActionCallbackRequest

Name	Type	Description
BundleLineItemIds	List	The list IDs of line items.
CartId	Id	The ID of the configuration

Response Parameters: Apttus_Config2.CustomClass.ActionCallbackResponse		
Name	**Type**	**Description**
errorMessages	List<String>	List of error messages if they occur.
IsSuccess	Boolean	Indicates whether the action was successful.

Example

The sample code below allows you to use **beforeDeleteBundleLineItems()** to validate the line items based on custom logic written in the method. You can allow the deletion of a line only when the following conditions are fulfilled:

- The number of line items to be deleted is not more than the limit defined in the code.
- The line items are not read-only and any approval request is not pending.
- The line items are not in collaboration.

```
1  global with sharing class APTS_ActionCallback3 implements
    Apttus_Config2.CustomClass.IActionCallback3
2  {
3      private static final String strEMPTY = '';
4      public static final Integer MAX_LI_TO_DELETE = 2;
5      public static final String BUSINESS_OBJECT_TYPE_PROPOSAL = 'Proposal';
6      public static final String PRODCONFIG_STATUS_PENDING_APPROVAL =
   'Pending Approval';
7      public static final String COLLAB_STATUS_COMPLETED = 'Completed';
8      public static final String COLLAB_STATUS_ACCEPTED = 'Accepted';
9      public static final String DEL_LIMIT_EXCEED_ERROR_MESSAGE = 'Deletion
   of more than \''+
10
   MAX_LI_TO_DELETE +
11                                                              '\' line
   items not supported. Please select fewer line items to delete';
12
13     public static final String PARENT_UC_ERROR_MESSAGE = 'Parent cart
   under collaboration, can not delete';
14
15     global static Apttus_Config2.CustomClass.ActionCallbackResponse
   beforeDeleteBundleLineItems(Apttus_Config2.CustomClass.ActionCallbackReque
   st CallBackReq)
16     {
17         Apttus_Config2.CustomClass.ActionCallbackResponse result = new
    Apttus_Config2.CustomClass.ActionCallbackResponse();
```

```
18          List<Id> lstLIToDel = CallBackReq.BundleLineItemIds;
19
20          // check the deletion limit.
21          if(lstLIToDel != null && lstLIToDel.size() > MAX_LI_TO_DELETE)
22          {
23              result.isSuccess= false;
24              result.errorMessages.add(DEL_LIMIT_EXCEED_ERROR_MESSAGE);
25              System.debug(LoggingLevel.ERROR,
DEL_LIMIT_EXCEED_ERROR_MESSAGE);
26              return result;
27          }
28
29          // Check for collaboration status.
30          String collabRequestStatus;
31          Boolean isCollabComplete;
32
33          for (Apttus_Config2__LineItem__c oLI : [
34              SELECT
Apttus_Config2__CollaborationRequestId__r.Apttus_Config2__Status__c,
35                     Apttus_Config2__CollaborationRequestId__c,
36
Apttus_Config2__ConfigurationId__r.Apttus_Config2__Status__c,
37
Apttus_Config2__ConfigurationId__r.Apttus_Config2__BusinessObjectType__c,
38                     Apttus_Config2__IsReadOnly__c
39              FROM Apttus_Config2__LineItem__c
40              WHERE Id IN :lstLIToDel]) {
41
42              collabRequestStatus =
(oLI.Apttus_Config2__CollaborationRequestId__c != NULL) ?
43
oLI.Apttus_Config2__CollaborationRequestId__r.Apttus_Config2__Status__c :
44                                      strEMPTY ;
45              isCollabComplete = (String.isNotBlank(collabRequestStatus) &&
46                                  !(collabRequestStatus ==
COLLAB_STATUS_COMPLETED || COLLAB_STATUS_ACCEPTED ==
collabRequestStatus)) ?
47                                      true :
48                                      false;
49
50              if(BUSINESS_OBJECT_TYPE_PROPOSAL ==
oLI.Apttus_Config2__ConfigurationId__r.Apttus_Config2__BusinessObjectType_
_c
```

```
51                && (oLI.Apttus_Config2__IsReadOnly__c ||
52                    isCollabComplete ||
53                    PRODCONFIG_STATUS_PENDING_APPROVAL ==
oLI.Apttus_Config2__ConfigurationId__r.Apttus_Config2__Status__c)){
54
55                 result.isSuccess= false;
56                 result.errorMessages.add(PARENT_UC_ERROR_MESSAGE);
57                 System.debug(LoggingLevel.ERROR, PARENT_UC_ERROR_MESSAGE);
58                 break;
59             }
60         }
61         return result;
62     }
63
64     global static Apttus_Config2.CustomClass.ActionCallbackResponse
beforeCopyBundleLineItems(Apttus_Config2.CustomClass.ActionCallbackRequest
callbackReq)
65     {
66             return new
 Apttus_Config2.CustomClass.ActionCallbackResponse();
67     }
68
69 }
```

Cart Approval Callback Class

Cart Approval Callback drives the Approvals functionality on the Cart page. You cannot use this callback to implement customization on CPQ. You can only use this callback to enable or disable the Approvals functionality. For more information, see Setting the Config Custom Classes.

Deal Optimizer Callback Class

Deal Optimizer Callback drives the Deal Maximizer functionality on the Cart page. You cannot use this callback to implement customization on CPQ. You can only use this callback to enable or disable the Deal Maximizer functionality. For more information, see Initiate and Integrate Deal Guidance.

Adjustment Spread Callback Class

Adjustment Spread Callback provides you a mechanism to restrict adjustment spread amount on option line items.

To use the Adjustment Spread Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IAdjustmentSpreadCallback* interface and register the custom apex class with Adjustment Spread Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IAdjustmentSpreadCallback* interface:

Method	Signature	Description
finish()	*void finish()*	This the last method that is executed after the callback is invoked.
spreadBundleAdjustment()	*List spreadBundleAdjustment(Apttus_Config2.LineItem, List)*	You can use this method to restrict the adjustment spread amount on options from the Bundle line adjustment.
spreadGroupAdjustment()	*List spreadGroupAdjustment(Apttus_Config2.SummaryGroup, List, Map>)*	You can use this method to restrict the adjustment spread amount on options from the summary group adjustment.
start()	*void start(Apttus_Config2.ProductConfiguration)*	This is the first method that is executed when the callback is invoked.

Example

The given code example implements both methods to restrict the adjustment spread values of option line items.

```
1  global with sharing class RG_AdjustmentSpreadCallback implements
    Apttus_Config2.CustomClass.IAdjustmentSpreadCallback
2  {
```

```
3
4     private static Apttus_Config2__ProductConfiguration__c config;
5     global static void start(Apttus_config2.ProductConfiguration cart)
6     {
7        config = cart.getConfigSO();
8     }
9
10     global static List<Apttus_config2.LineItem>
spreadGroupAdjustment(Apttus_config2.SummaryGroup parentGroup,
List<Apttus_config2.LineItem> childLineItems, Map<ID,
List<Apttus_config2.LineItem>> optionItemsByBundleId)
11     {
12
13         List<Apttus_config2.LineItem> lines;
14         if(config.Price_list_name__c== 'Auto Pricing Call Back Pricelist2')
15         {
16                 ID lineId = childLineItems.get(0).getLineItemSO().ID;
17                 lines = optionItemsByBundleId.get(lineId);
18                 for (Apttus_config2.LineItem optionLineItem : lines)
19                 {
20
21                     //Below line use to restrict adjustment spread amount
on option line items.
22
optionLineItem.getLineItemSO().Apttus_Config2__AdjustmentAmount__c =
7.99999;
23                 }
24         }
25         return lines;
26     }
27
28
29 global static List<Apttus_config2.LineItem>
spreadBundleAdjustment(Apttus_config2.LineItem parentItem,
List<Apttus_config2.LineItem> optionItems)
30     {
31       for(Apttus_Config2.LineItem OI: optionItems)
32       {
33         if(config.Price_list_name__c== 'Auto Pricing Call Back Pricelist')
34         {
35           Apttus_Config2__LineItem__c item = OI.getLineItemSO();
36
37           //Below line use to restrict adjustment spread amount on option
line items.
38           item.Apttus_Config2__AdjustmentAmount__c = 8.99999;
```

```
39            }
40          }
41          return optionItems;
42        }
43    }
```

Lifecycle Callback Class

Lifecycle Callback provides you a mechanism to execute custom actions at the following stages in the quote process. class

- after a proposal is cloned
- after an agreement is created from a proposal
- creates a new proposal from the given account
- creates a new proposal from the given opportunity
- clones the given proposal
- creates a new agreement from the given proposal
- after a proposal is created from an account
- after a proposal is created from an opportunity

To use the Lifecycle Callback you must create a custom Apex class that implements the *Apttus_Proposal.CustomClass.IQuoteLifecycleCallback2* interface and register the custom apex class with Lifecycle Callback Class. You must write your custom logic in the custom apex class.

> ⓘ You must register the custom apex class for this callback in **Lifecycle Callback Class** in **Proposal Custom Classes** instead of **Config Custom Classes**.

The following methods are available in the *Apttus_Proposal.CustomClass.IQuoteLifecycleCallback2* interface:

Method	Signature	Description
afterClone()	*void afterClone(Apttus_Proposal__Proposal__c, Apttus_Proposal__Proposal__c)*	You can use this method to execute custom logic after the proposal is cloned. CPQ does not invoke this method when you clone any proposal using the standard Salesforce **Clone** button. You must create a custom button or a formula field on the Proposal object to invoke this method after cloning a proposal.
afterCreateAgreement()	*void afterCreateAgreement(Apttus_Proposal__Proposal__c, Apttus__APTS_Agreement__c)*	You can use this method to execute custom logic after the agreement is created from proposal.
afterCreateFromAccount()	*void afterCreateFromAccount(Apttus_Proposal__Proposal__c)*	You can use this method to execute custom logic after the proposal is created from account. CPQ only invokes this method when you execute the **Apttus_Proposal.ProposalWebService.afterCreateFromAccount** API.

Method	Signature	Description
afterCreateFromOpportunity()	*void afterCreateFromOpportunity(Apttus_Proposal__Proposal__c)*	You can use this method to execute custom logic after the proposal is created from opportunity. CPQ only invokes this method when you execute the **Apttus_Proposal.ProposalWebService.afterCreateFromOpportunity** API.
afterComplete()	*void afterComplete(Apttus_Proposal__Proposal__c, Apttus_Proposal.CustomClass.ActionType)*	You can use this method to execute custom logic after proposal lifecycle is completed.
afterComplete()	*void afterComplete(Apttus__APTS_Agreement__c, Apttus_Proposal.CustomClass.ActionType)*	You can use this method to execute custom logic after agreement lifecycle is completed.

Example

You can use this callback to a scenario like:

- After you create a new proposal from Opportunity, values of **Bill To Account** and **Ship To Account** fields should be automatically set from the same fields that are present in the associated account.
- When you clone a proposal, you should have information about the proposal from which the new proposal was cloned from.

The below code implements the first example mentioned above.

```
/*
* Quote Life Cycle Call Back - Exmaple 1
* This is use case of executing Quote Life Cycle Call Back.
* Method - afterCreateFromOpportunity() of this class will be called whenever we go
to opportunity and click on Create Quote/Proposal button.
```

```
* Here, We will set BillToAccount and ShipToAccount fields of newly created proposal
by taking info of account from opporunity.
*/
global with sharing class APTS_PS_QuoteLifecycleCallback_UC1
    implements Apttus_Proposal.CustomClass.IQuoteLifecycleCallback2
{

    /**
     * Callback invoked after a quote/proposal is created from an account
     * @param quoteSO the new quote sobject
     */
    global void afterCreateFromAccount(Apttus_Proposal__Proposal__c quoteSO) {
    }

    /**
     * Callback invoked after a quote/proposal is created from an opportunity
     * @param quoteSO the new quote sobject
     */
    global void afterCreateFromOpportunity(Apttus_Proposal__Proposal__c quoteSO) {
        /* USECASE# 1 */

        String billTOAcc = quoteSO.Apttus_QPConfig__BillToAccountId__c;
        String shipTOAcc = quoteSO.Apttus_QPConfig__ShipToAccountId__c;
        String accId = quoteSO.Apttus_Proposal__Account__c;


        if(billTOAcc == null && shipTOAcc == null) {
            billTOAcc = accId;
            shipTOAcc = accId;
        } else if(billTOAcc == null) {
            shipTOAcc = billTOAcc;
        } else if(shipTOAcc == null) {
            billTOAcc = shipTOAcc;
        }


        quoteSO.Apttus_QPConfig__BillToAccountId__c = billTOAcc;
        quoteSO.Apttus_QPConfig__ShipToAccountId__c = shipTOAcc;

    }

    /**
     * Callback invoked after an agreement is created from a quote/proposal
     * @param quoteSO the quote sobject
     * @param agreementSO the new agreement sobject
```

```
     */
    global void afterCreateAgreement(Apttus_Proposal__Proposal__c quoteSO,
Apttus__APTS_Agreement__c agreementSO) {
    }

    /**
     * Callback invoked after a quote/proposal is cloned
     * @param originalSO the original quote sobject
     * @param cloneSO the clone quote sobject
     */
    global void afterClone(Apttus_Proposal__Proposal__c originalSO,
Apttus_Proposal__Proposal__c cloneSO) {
    }

     /**
     * Callback invoked before a quote/proposal is cloned
     * @param originalSO the original quote sobject
     * @param cloneSO the clone quote sobject
     */
    global void beforeClone(Apttus_Proposal__Proposal__c originalSO,
Apttus_Proposal__Proposal__c cloneSO) {
    }

    /**
     * Callback invoked after the completion of a lifecycle action
     * @param quoteSO the quote sobject
     * @PARAM actionType the lifecycle action type
     */
    global void afterComplete(Apttus_Proposal__Proposal__c quoteSO,
Apttus_Proposal.CustomClass.ActionType actionType) {
   }

    /**
     * Callback invoked after the completion of a lifecycle action
     * @param agreementSO the agreement sobject
     * @PARAM actionType the lifecycle action type
     */
    global void afterComplete(Apttus__APTS_Agreement__c agreementSO,
Apttus_Proposal.CustomClass.ActionType actionType) {
    }
}
```

Tax Callback Class

Tax Callback Class provides you the mechanism to communicate with a Tax Engine of your choice for tax calculation on invoice generation. You need to provide details like Invoice Line Item or a Credit Memo Line Item, Handback, Tax Address, Tax Code, and Taxable Amount to the callback. You must register the Tax Callback Class in Config Custom Classes in Custom Setting.

For more information, see Tax Callback Class. You can find information about how to setup the callback, method description, the working of the callback, and sample code in that topic.

Formula Callback Class

This callback has been deprecated. You must use the expression builder to create and use formulas.

Metadata Callback Class

This callback has been deprecated. Salesforce platform has been optimized with a better cache handle. This callback is no longer needed to maintain the metadata.

Recommendation Callback Class

Recommendation Callback provides you a mechanism to define products in the recommended products lists in addition to the products defined by the recommendation type constraint rules. This callback is invoked when constraint rules are executed on the Catalog page if the setting **Show Recommended Products Cart View** is enabled in Config Select Products Settings. After execution, Sales Rep sees the recommended products in the Recommendation popup.

To use the Recommendation Callback you must create a custom Apex class that implements the *Apttus_Config2.CustomClass.IRecommendationCallback* interface and register the custom apex class with Recommendation Callback Class. You must write your custom logic in the custom apex class.

The following methods are available in the *Apttus_Config2.CustomClass.IRecommendationCallback* interface:

Method	Signature	Description
getRecommendedProducts()	*List getRecommendedProducts(Apttus_Config2.ProductConfiguration)*	You can use this method to define and recommend products. You must take into consideration the effective price list of the Quote. If the recommended product is not part of the price list in effecting for the quote, Sales Rep is not allowed to use those products in any operation.

Example

The given code example defines a product for recommendation.

```
global with sharing class SampleRecommendationCallback implements
 Apttus_Config2.CustomClass.IRecommendationCallback
{

    global List<Apttus_Config2.CustomClass.RecommendationInfo>
getRecommendedProducts(Apttus_Config2.ProductConfiguration cart)
    {
        Apttus_Config2.CustomClass.RecommendationInfo recommendedProduct =
new Apttus_Config2.CustomClass.RecommendationInfo('a0S0B000004JrYF',
'Relevance');

        // product ID, relevance

        return new
 List<Apttus_Config2.CustomClass.RecommendationInfo>{recommendedProduct};
    }
}
```

Advanced Approval Callback Class

This callback has been deprecated. Conga recommends not define any customization for approvals.

Configuring Admin Settings

Admin Settings are system properties that have different values and you can use them for different purposes. You must configure admin settings to meet the business objectives of your implementation along with Custom Settings.

To create admin settings

1. Click  and click **Admin**. All the standard Admin Settings are displayed on the Admin home page. Based on your organizations' requirements, you can add or create new Admin settings.
2. To create a new Admin Setting, click **New**.
3. Enter **Name**, **Value**, and **Code** for the Admin Setting and click **Save**. Your new Admin Setting is saved and added.

Admin Settings in CPQ

The following content provides information about various admin Settings available in CPQ along with their values and purposes.

APTS_AsyncMergeCall

You can use this setting to enable asynchronous document generation for large templates. A notice is displayed when the Sales rep performs the document generation after a number of seconds/milliseconds determined by the property **APTS_MergeCallTimeOut** or **Merge Time Callout Millis** Comply System Property , and permits the Sales rep to perform other operations while the document generates in the background.

Name	*APTS_AsyncMergeCall*
Value	• *True* • *False*
Code	Leave the field blank

APTS_AsyncMergeEmail

You can use this setting to send a notification email to the Sales rep initiating the asynchronous document generation after generation is complete.

> Conga recommends to enable both **APTS_AsyncMergeCall** and **APTS_AsyncMergeEmail**.

Name	*APTS_AsyncMergeEmail*
Value	• *True* • *False*
Code	Leave the field blank

APTS_DefaultEmailContactName

You can use this setting to define the default email contact name.

Name	*APTS_DefaultEmailContactName*
Value	Enter a value to define the default email contact name.
Code	Leave the field blank

APTS_DisableCartActivityHistory

You can use this setting to disable the generation of activity history product configuration. When enabled, CPQ does not add entries to the **Activity History** related list, when you create, finalize, or reconfigure a product configuration.

Name	*APTS_DisableCartActivityHistory*

Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_DisableCartAsyncNotification

You can use the Admin Setting **APTS_DisableCartAsyncNotification** to turn off the email notifications you receive after completion of any async operation such as cart finalization and pricing in every type of quote or agreement. In the Split quote or agreement, the notification for cart splitting is also disabled along with finalization and pricing. If you enabled the Progress Tracker on the quote or agreement page to monitor the progress of all the async operations, email notifications are redundant and you can use this feature to avoid them.

Name	*APTS_DisableCartAsyncNotification*
Value	*true*
Code	Leave the field blank

APTS_DisableConstraintRulesOnFinalize

You can use this setting to disable the execution of server-side constraint rules to optimize performance upon cart finalization.

ⓘ This feature is not supported with the Replacement type Constraint Rule.

Name	*APTS_DisableConstraintRulesOnFinalize*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_DisableCumulativeQuantityAcrossCart

You can use this setting to disable CPQ from applying the cumulative range matrix on multiple line items of the same product.

Name	*APTS_DisableCumulativeQuantityAcrossCart*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_DisablePLITimeZoneAdj

You can disable the time zone adjustment on the **Effective Date** and **Expiration Date** fields in the Price List Item. By default, the time on the **Effective Date** and **Expiration Date** fields is adjusted based on the Time Zone of the Locale in the org. If you disable the adjustment, CPQ does not change the time you manually configured in the **Effective Date** and **Expiration Date** fields. Refer to Adding Additional Details to a Price List Item for information about the Price List Item fields.

Name	*APTS_DisablePLITimeZoneAdj*
Value	The following are the valid values: • *true*: disables time zone adjustment • *false*: enables time zone adjustment
Code	Leave the field blank

APTS_DisableRolldownOptimization

You can use this setting to optimize the system behavior while using custom logic in the methods invoked in the Roll Down mode.

Name	*APTS_DisableRolldownOptimization*

Value	The following are the valid values: • *True*: the method in Rolldown mode is invoked when the Sales rep clicks **Got To Pricing** or **Reprice** after changing the quantity or applying summary level discount. • *False*: the method in Rolldown mode is only invoked when the Sales rep clicks **Reprice** after applying summary level discount.
Code	Leave the field blank

APTS_EnableBlockingCallsOnServiceCatalog

Use this admin setting to display a spinner on the Service Catalog upon the click of the **Add to Cart** or **Update** buttons. When this admin setting is set to true, CPQ displays the spinner to ensure that the creation of related line items is completed for service products before you navigate away to a different page or click another button.

When this admin setting is set to false, if you quickly navigate away from the Service Catalog after clicking **Add to Cart**, CPQ does not create related line items for service products.

Name	*APTS_EnableBlockingCallsOnServiceCatalog*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_EnableInstallBaseFilteringAcrossAccounts

You can use this setting along with Asset Line Item Callback to display assets from multiple accounts, which are related to an opportunity, on the Installed Products page. For more information to configure the callback, see Displaying Assets from Multiple Accounts on the Installed Products Page.

Name	APTS_EnableInstallBaseFilteringAcrossAccounts

Value	Enter one of the following valid values: • *True*: CPQ displays assets from multiple accounts. • *False*: This is the default value. CPQ displays assets from a single account related to the quote.
Code	Leave the field blank

APTS_EnableMultiBenefitItems

You can use this setting to control the ability of the Sales rep to apply promotions to multiple line items of the benefit product (Y). By default, this setting is disabled and CPQ applies the promotion only to the first line item of the benefit product (Y). When you enable this setting, CPQ applies the promotion to all line items of the benefit product (Y).

Name	*APTS_EnableMultiBenefitItems*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_ExcludeOptionalProducts

You can use this setting to exclude options, bundle, and sub-bundle that are marked optional from the min/max criteria of the option group. When the Sales rep marks options or bundles as optional using the **Is Optional** checkbox, they are not considered in the **Min Options**, **Max Options**, **Min Total Quantity**, and **Max Total Quantity** criteria.

Name	*APTS_ExcludeOptionalProducts*
Value	Enter true to enable the exclusion.
Code	Leave the field blank

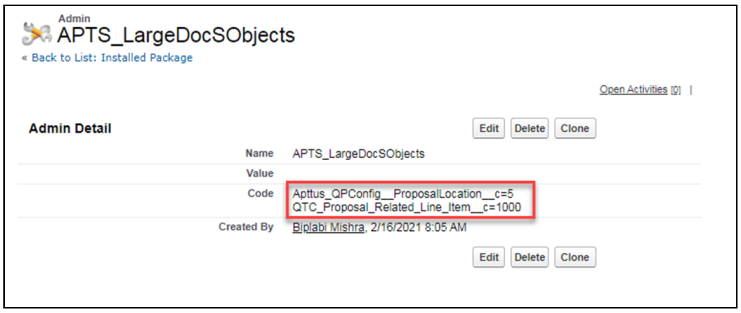
APTS_IncentivePricingBatchSize

You can use this setting to process the promotions in batches for a cart with a large number of line items. The batch processing is only supported for Buy X Get X and Buy X Get Y promotions with Auto Apply? enabled. You can use this feature for both regular and async pricing.

Name	*APTS_IncentivePricingBatchSize*
Value	Enter the number of line items to be processed in a batch. ⓘ You must define the batch size based on the complexity of the cart and the number of incentives applied on the line items.
Code	Leave the field blank

APTS_LargeDocSObjects

You can use this setting to define objects as Large Objects for document generation. You can also define a batch size for each of these objects. By default, CPQ considers Proposal Line Item and Agreement Line Item as Large Objects for document generation.



Name	*APTS_LargeDocSObjects*
Value	Leave the field blank

Code	Use the format Auto Applied *<Object_API_name=<Batch Size>>* to defined the objects. You can add more objects separated by a new line. Decide the batch size based on the complexity of the cart and template. For example, *Apttus_QPConfig__ProposalLocation__c=5* *QTC_Proposal_Related_Line_Item_c=1000*

APTS_LoadProductRelatedAttributesOnly

You can use this setting to enable the optimized retrieval of attributes on the cart page. When this setting is enabled, CPQ only retrieves the attributes in the cart that are associated with the product added to the cart. By default, all the attributes from Product Attribute Value, Product Attribute Value Ext, Product Attribute Value Ext 2, Product Attribute Value Ext 3, and custom Product Attribute Value Ext objects are loaded on the Cart page.

Name	*APTS_LoadProductRelatedAttributesOnly*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_MergeCallTimeout

You can use this setting to display a timeout message whenever document generation exceeds 60 seconds.

Name	*APTS_MergeCallTimeout*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_MS_EnableNameSort

You can use this setting to specify whether the related list should be sorted by the name field in the generated document. When enabled the related lists will be printed in the merge documents in the order of the name field of the related list.

Name	*APTS_MS_EnableNameSort*
Value	Enter one of the following valid values: • *True* • *False*
Code	N/A

APTS_NoISOCurrencyFieldObjects

You can use this setting to add support for custom object generation for objects that do not have a CurrencyIsoCode field (e.g., "EventRelation," "Task," "TaskRelation"). Separate Object names using a comma or new line. The setting is only applicable to templates in multi-currency org.

Name	*APTS_NoISOCurrencyFieldObjects*
Value	XML
Code	<Object API name>

APTS_NoNameFieldObjects

You can use this setting to add support for custom object generation for Objects that do not have a Name field. (e.g., "EventRelation," "Task," "TaskRelation") Separate Object names using a comma or new line.

Name	APTS_NoNameFieldObjects
Value	XML
Code	<Object API name>

APTS_Password

You can use this setting to create a password for protecting Microsoft Word agreements that are generated by the application. It also allows the application to password protect generated Microsoft Word documents.

Name	APTS_Password
Value	<password>
Code	Leave the field blank.

APTS_PricingServiceUrl

You must use this setting to set the Turbo Engines URLs. For more information, see Setting Up the TurboPricing Endpoint URLs.

APTS_PriceListItemCustomFields

You use this setting to auto-retrieve the custom fields in the **onPriceItemSet()** method in the Pricing Callback Class. For more information on Pricing Callback methods, see Pricing Callback Class.

Name	*APTS_PriceListItemCustomFields*
Value	Leave the field blank.
Code	Enter the API names of the custom field of the Price List Item separated by a comma or on a new line.

APTS_ProposalConfig

You can use this setting to enable pdf security on the generated Quote/ Proposal document. Before defining this setting you must configure **Enable PDF Security** and **PDF Owner Password**.

Name	*APTS_ProposalConfig*
Value	Enter the value *XML*.

Code	Enter the following code:
	
	`<ProposalConfig>`
	`<PDFSecurityDefault>`
	`<CanPrint>true</CanPrint>`
	`<CanCopy>true</CanCopy>`
	`<CanChange>false</CanChange>`
	`<CanAddNotes>true</CanAddNotes>`
	`<CanFillFields>true</CanFillFields>`
	`<CanAssemble>false</CanAssemble>`
	`</PDFSecurityDefault>`
	`</ProposalConfig>`

You can also use this setting to define the line and option sequence.

Name	*APTS_ProposalConfig*
Value	Enter the value *XML*.

Code	Enter the following code:
	
	`<ProposalConfig>`
	`<SortSpec>`
	`<SortObjects>`
	`<SortObject>`
	`<Name>Apttus_Proposal__Proposal_Line_Item__c</Name>`
	`<SortFields>`
	`<SortField>`
	`<Name>SequenceLine__c</Name>`
	`</SortField>`
	`<SortField>`
	`<Name>SequenceOption__c</Name>`
	`</SortField>`
	`</SortFields>`
	`</SortObject>`
	`</SortObjects>`
	`</SortSpec>`
	`</ProposalConfig>`
	
	Depending upon your business object (Quote/Proposal or Agreement), you can enable this for *ProposlConfig* or *ComplyConfig* respectively.

APTS_RollUpOptionNetAdjustment

You can use this setting to display the adjustments on options at the bundle and summary totals level.

Name	*APTS_RollUpOptionNetAdjustment*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_SortDocumentsDesc

You can use this admin setting to sort the documents generated on the Doc Gen page, in descending order of the creation date by defining it as true. If this setting is not configured or set to false, the documents are sorted in ascending order of creation date, by default.

Name	*APTS_SortDocumentsDesc*
Value	Enter one of the following valid values: • *True* • *False*
Code	Leave the field blank

APTS_UpdateViewProductClasses

You can use this setting to execute the Category Maintenance batch job for the changes you make modifications to existing records instead of executing the batch job on the entire category definition.

Name	*APTS_UpdateViewProductClasses*
Value	Leave the field blank
Code	Enter the Product Classification records of the newly added products. ⓘ These records are created after you associate products to the category. In case, you want to execute a maintenance batch job for the entire definition, clear the **Code** field, and then execute the Category Maintenance batch Job.

APTS_UpdateViewApiBatchSize

You can use this setting to specify the batch size for the Category Maintenance batch job when the job is executed using the exposed global Apex method CPQWebService.updateHierarchyViews(). You must specify a batch size value only when

you notice limit exceptions in batch jobs because of complex categories with a lot of hierarchy levels. In such cases, you must decrease the batch size to execute the batch jobs successfully.

Name	*APTS_UpdateViewApiBatchSize*
Value	Enter a value between 1 and 2000. The default batch size for the Category Maintenance batch job set in this global method is 2000.
Code	Leave the field blank

Frequently Asked Questions

Why do I get the 'An internal server error has occurred' message when I click Configure Products, Go to Pricing, or Reprice?

This is a Salesforce error. Contact Salesforce Support for further assistance.

Why can't I approve requests using email notifications?

1. Make sure that the email id saved in the email reply template on the visual force page is the same as the one saved in **Custom Settings** > **Email Services**.
2. Make sure that the Approver is a registered User with a valid Conga license and then complete the following procedures.
    a. Click **Develop** > **Email Services** to create a new email service using ApprovalEmailHandler and generate an email address.
    b. Click **Develop** > **Custom Settings** > Approval System Properties and enter values for the fields described in the following table.

Field	Value
Enable Email Approval Response	Select the check box.
Email Approval Service Address	Enter the approver's email address.

    c. On the Visual force email template (that is sent during approval submission) and enter the email address you generated in step 2.a, in the **Reply To** attribute of the template.
    d. Make sure that the template has the following fields.
        i. **Approval Request ID**
        ii. **Approval User ID**

e. If you use a Custom VF Email Template, make sure the Conga standard Merge fields (Approval ID and User ID) are NOT defined in Table format in code in Custom VF template or associated component. If these merge fields are in a Table format then Email to Approval will not work.

> **Note**
>
> Conga does not support the system fields (Approval ID and User ID) in tabular format in the Custom VF email template code or the associated component code.

To resolve this, you must remove the Conga standard Merge fields (**Approval ID** and **User ID**) from table and insert them separately in the following format.

Approval Request ID: {!relatedTo.ID}
Approval User ID: {!relatedTo.Apttus_Approval__Assigned_To_Id__c}

How can I resolve a hung page when I try to check-in or check-out any document on Conga Author?

1. Make sure your profile has permission to Read, Create, Edit, and Delete the Template Object.
2. Make sure your profile has access to the following Visual force pages.
   a. Apttus.BrowseTemplates
   b. Apttus.BrowseAgreements
3. Log in with your credentials and paste the following link in the browser's URL.
   https://<instance>.salesforce.com/apex/Apttus__BrowseTemplates?callerMode=Checkout

Why do I get the 'An internal server error has occurred' message when I click Configure Products, Go to Pricing, or Reprice?Why does my sandbox instance redirect to my production instance whenever I refresh the page?

This is a Salesforce error. Contact Salesforce Support for further assistance.

Why do I get the 'An internal server error has occurred' message when I click Configure Products, Go to Pricing, or Reprice?

When you refresh the sandbox, the app copies all the data and components from the production instance to the sandbox instance.

The following table lists the applications and corresponding custom settings you must change or verify to resolve this issue.

> **Note**

The instructions in the following table direct you to change settings in your sandbox instance. If you face the same problem on your production instance (switching to the sandbox instance when you refresh), use the same procedure but be sure to enter the URL for the production instance.

App	Settings
CPQ	1. Click **Setup** > **App Setup** > **Develop** > **Custom Settings** > **Config System Properties** > **Edit System Properties.** 2. Change the Production instance URL to the sandbox instance URL.
CLM	1. Click **Setup** > **App Setup** > **Develop** > **Custom Settings** > **Comply System Properties** > **Edit System Properties.** 2. Change the Production instance URL to the sandbox instance URL.
Approval Management	1. Click **Setup** > **App Setup** > **Develop** > **Custom Settings** > **Approval System Properties** > **Edit System Properties.** 2. Change the Production instance URL to the sandbox instance URL.

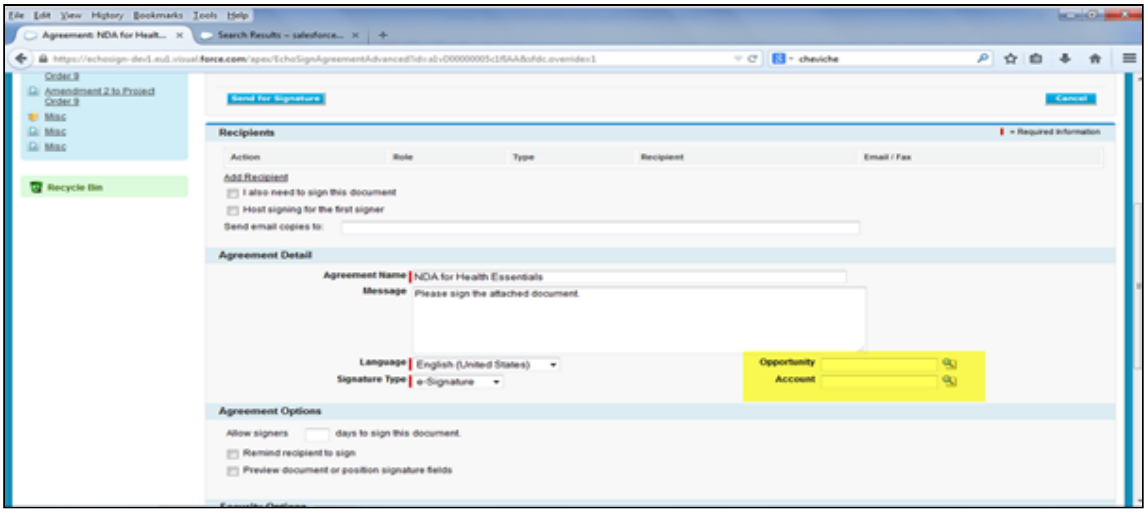
What Object references should I set to the instance of an Object so I can log in to X-author for contracts?

If you have this question, you receive the following message when you try to log in to X-Author for Contracts.

This error is not because there are Object permission issues in CPQ, in spite of the wording, this is a Time Zone error. To resolve this issue, change the time zone in the user profile settings to match the time zone in the Org settings.

How do I get the EchoSign Send page to auto-populate the Account and Opportunity fields?

The Account and Opportunity fields on the EchoSign Send page are lookup fields and the Conga Integration Adaptor cannot auto-populate these fields.



To auto-populate these fields, employ one of the following work-around.

1. Create a custom trigger to auto-populate the **Account** and **Opportunity** fields from the Agreement. Because these are lookup fields, you cannot use workflow rules.
2. Click **Custom Settings** > **EchosignSettings** > **Edit**.
3. On the EchoSign settings page, uncheck the following check boxes.
    - **Disable Opportunity Lookup**
    - **Disable Account Lookup**
4. Click **Save.**

How do I enter values for custom fields on the Price List Item page?

The Price List Item detail page is a managed, custom visual force page where you cannot enter values for custom fields.

To be able to enter values on the Price List Item detail page, you must create a button of type URL that sales users can access to define values for custom fields. Use the following URL when you create this button.

```
{!URLFOR($Action.Apttus_Config2__PriceListItem__c.Edit,
Apttus_Config2__PriceListItem__c.Id,
```

```
[retURL=URLFOR($Action.Apttus_Config2__PriceListItem__c.View,Apttus_Config2__PriceLis
tItem__c.Id)],true)}
```

How is the Base Price calculated when I renew or change an asset with custom pricing?

You must use custom pricing for an asset if you want to override the defined current price of the asset.

For example, you are a Sales Representative for a software company and you receive a request to renew a software subscription. The price of the original order is $100.00. The current price of the product is 150.00. You can process the renewal for the original price if you enable Custom Pricing.

If you have enabled Custom Pricing for an asset, CPQ uses the Net Price of the asset to calculate the asset price. The formula is ***Net Price/Quantity*Term***.

When I create any template, why do I see the following error?

```
Error: Script Thrown Exception
```

Your profile does not have Edit permissions and so you do not have access to the following custom fields for the Agreement Object.

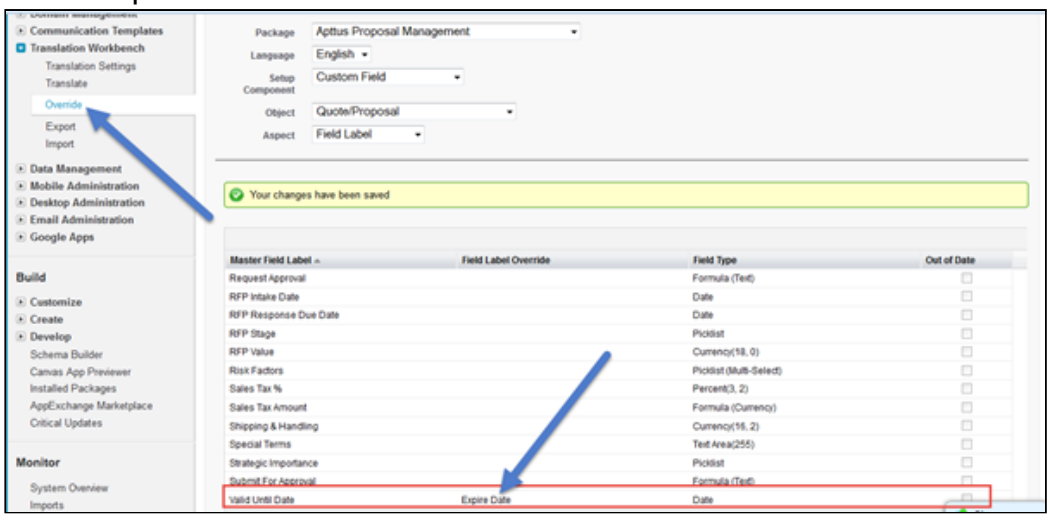
- Status Category
- Status
- Workflow Trigger Generated Agreement

Ask your system administrator to enable Edit permissions for the Agreement Object.

How do I change the label for an Conga Managed field?

To change the label of a Conga Managed field,

1. Click **Setup** > **Translation Workbench** > **Override**.



2. Double click in the **Field Label Override** column to enter a new value.
3. Click **Save**.

How do I clone a User Record?

Cloning a User record is not a Salesforce function. You must create a custom link along with formula to clone a user record.

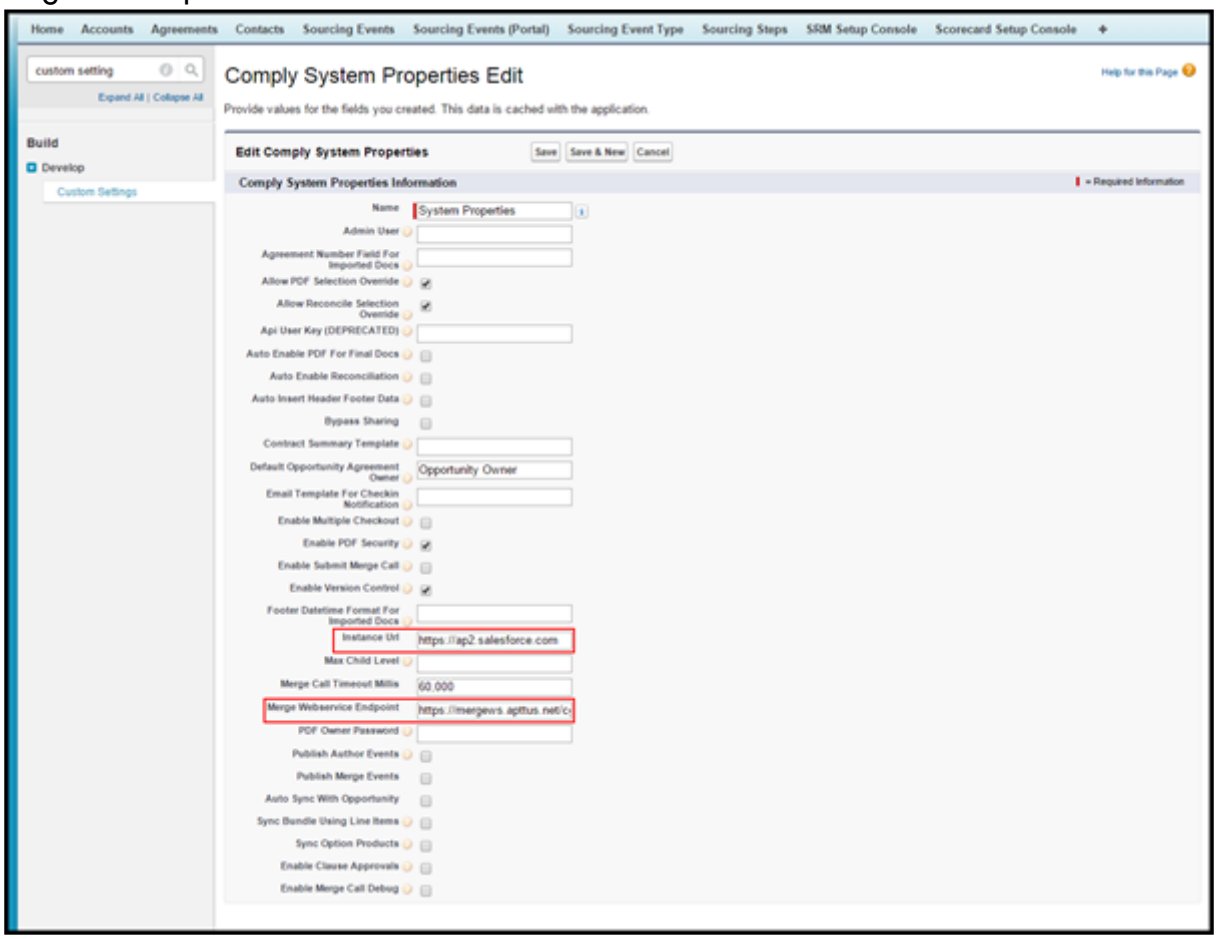
Create a custom link with the following formula.

```
/{!User.Id}/e?clone=1&retURL=%2F{!User.Id}&name_firstName=&name_lastName=&Alias=&Em
```

How do I add the Conga Generate document functionality on any custom object?

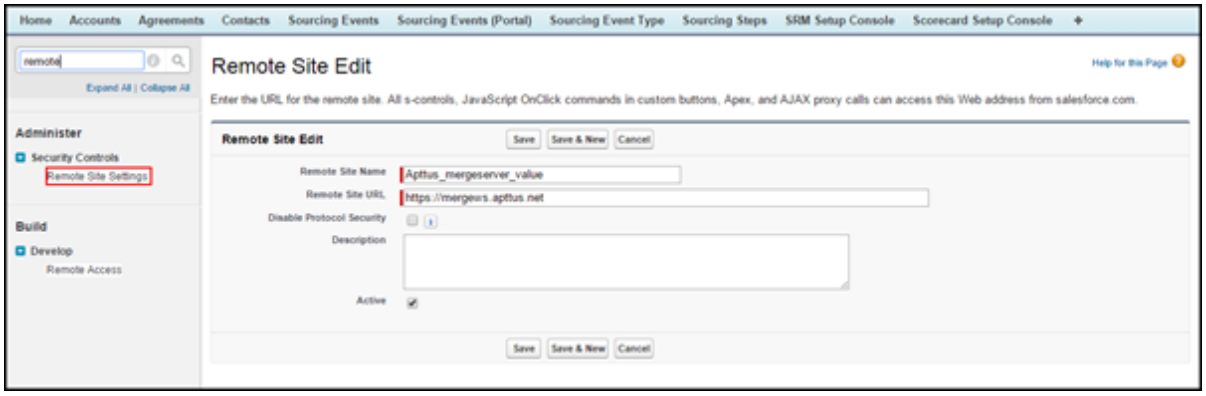
You can add the Conga Generate document functionality on any custom object by creating a formula on the custom object **Edit** .

1. Click **Setup > Build > Develop > Custom Settings > Comply System Properties > System Properties > Edit**.



2. Define values for the following fields.
    - **Merge Webservice Endpoint**
    - **Instance URL**

3. Click **Save**.
4. Click **Setup** > **Administer** > **Security Controls** > **Remote Site Settings** > **New Remote Site**.



5. Define values for the following fields.
   - **Remote Site Name**
   - **Remote Site URL**(your merge server URL)
6. Create a **Record Type** (at least on record type. For example, **Rec Type**) on the custom object, where you want to add the Generate document functionality.
7. Add that record type to the Agreement Types field (*Apttus__APTS_Template__c.Apttus__Agreement_Types__c*) as a picklist value.
8. Add that record type to the Type field (*Apttus__APTS_Template__c. Apttus__Type__c*) as a picklist value.
9. Create a Formula field for the **Generate** button on your custom object with the below expression.

```
("/apex/Apttus__DocumentGenerate?id="&Id&"&action=Generate_Document&templateTyp
e=RecTyp",IMAGE("/resource/Apttus__Button_Generate","Generate"),"_self")
```

10. Click **Save**.

When I submit the agreement by clicking Submit Request button, why do I get the following error?

```
Error: Missing Target ObjectId with Template
```

You don't have default contact access to the System Administrator Record.There is an Admin Entry created and defined as System Administrator as a Default Email Contact Name.

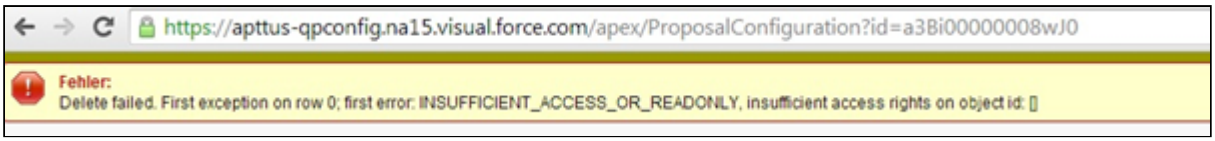
APTS_DefaultEmailContactName: Symantec Admin

Whenever you submit the Agreement record, the system looks for System Administrator contact. But, the OWD settings is defined as Private for Contact object, so this contact record doesn't have access to the concerned user. So the concerned user must have the System Administrator contact access to submit the records.

> **Note**
>
> **The System Administrator has all record access automatically. So, Admin should not get any issue, but the users don't have access to all records.**

Your System Administrator must provide you with access to **System Administrator** record sharing.

When I click the Configure Products button, why do I get the following error?



You will need **Modify All** permission on Product Configurations object.

The Quote Owner or System Admin will be able to configure any products for a quote. If any other user other than quote owner will try to do the same, the system will now the user to configure the Quote and will get the above error.

Ask your System Administrator to provide you with **Modify All** permissions on Product Configurations object.

When I run the Bundle Maintenance, why do I get the following error?

```
Error: First error: Insert failed. First exception on row 0; first error:
NUMBER_OUTSIDE_VALID_RANGE, Depth: value outside of valid range on numeric field: 10:
[Apttus_Config2__Depth__c]
```

You get this error when you've already ran bundle maintenance earlier, and in the middle you abort this process. In our system, we maintain the maintenance jobs status in an object called Batch Job (Apttus_Config2__BatchJob__c). Once the status is stuck in Queued stage, the actual batch job is aborted, and you will face this problem.

Resolution:

When you receive the above error, after running Bundle Maintenance, perform the following steps to resolve records with product components where the option group is null in the **Product Option Component** object.

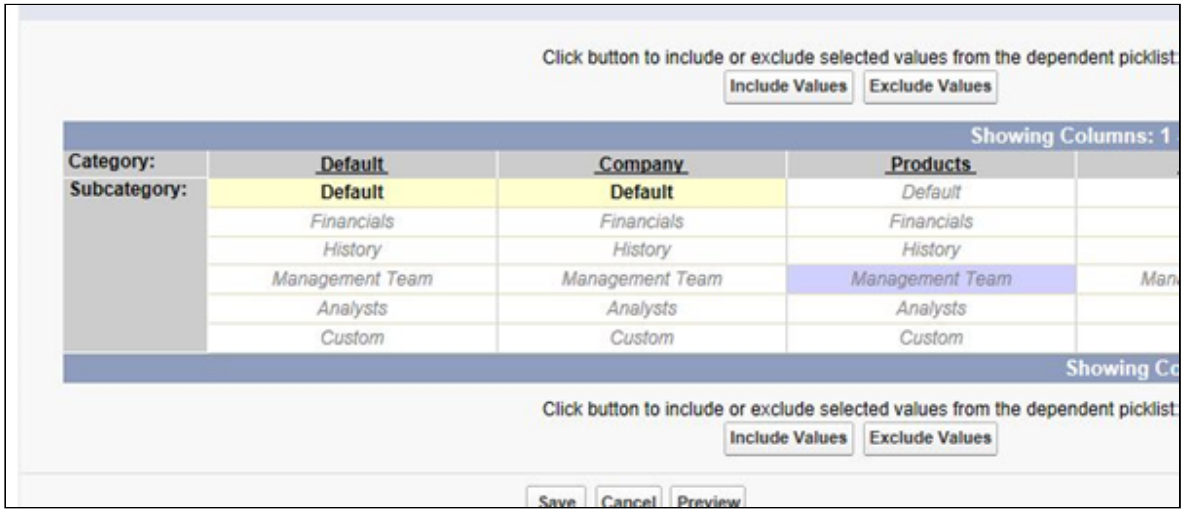
Run the following Query in the Developer Console. (a Salesforce System Administrator should be utilized or informed when performing this operation).

```
SELECT name, id, Apttus_Config2__ComponentProductId__c,
Apttus_Config2__RelationshipType__c,Apttus_Config2__ParentProductId__c from
Apttus_Config2__ProductOptionComponent__c where
Apttus_Config2__ProductOptionGroupId__c = NULL
```

You will receive a list of returned IDs from the above Query. You must identify and delete the Product Component records without any option group assigned. Now, the Bundle Maintenance will execute properly.

How do I include the values in Subcategory field which are lost while saving a template in XAuthor V6.0?

1. Go to **Setup** > **Create** > **Objects.**
2. Search for **Template** object and click to open it.
3. Go to **Custom Fields and Relationships**.
4. Click **Category.**
5. Go to **Field Dependencies** and edit **Subcategory** to see the following screen.



Conga Contact Support

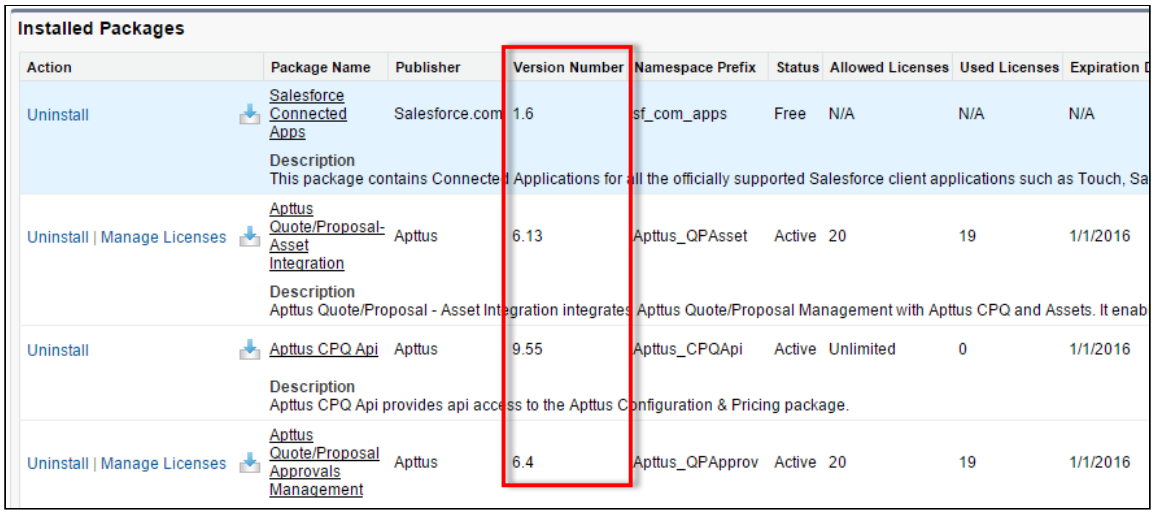
If you experience an issue with an Conga product and need help, you can contact Conga Support. Before you contact Conga support, prepare a brief description of the problem you are experiencing. Additionally, to enable us to resolve your problem at the earliest, provide the following important information:

- What is the environment in which you are experiencing the problem: Sandbox or Production?
- How many users are affected?

Which product versions are installed?

To determine version numbers:

1. Go to **Setup > App Setup > Installed Packages**.
2. In the Installed Packages section, all the installed packages are displayed. You can find the version numbers in the Version Number column.



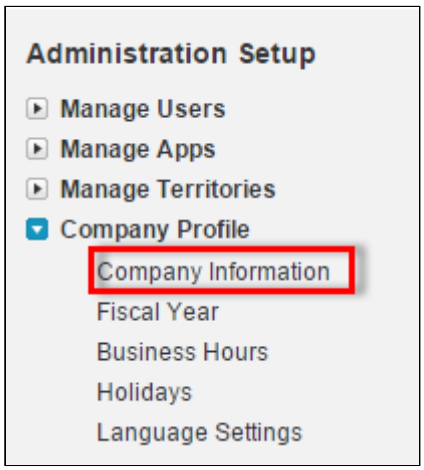
Installed Packages

Action	Package Name	Publisher	Version Number	Namespace Prefix	Status	Allowed Licenses	Used Licenses	Expiration D
Uninstall	Salesforce Connected Apps	Salesforce.com	1.6	sf_com_apps	Free	N/A	N/A	N/A
	Description: This package contains Connected Applications for all the officially supported Salesforce client applications such as Touch, Sa							
Uninstall | Manage Licenses	Apttus Quote/Proposal-Asset Integration	Apttus	6.13	Apttus_QPAsset	Active	20	19	1/1/2016
	Description: Apttus Quote/Proposal - Asset Integration integrates Apttus Quote/Proposal Management with Apttus CPQ and Assets. It enab							
Uninstall	Apttus CPQ Api	Apttus	9.55	Apttus_CPQApi	Active	Unlimited	0	1/1/2016
	Description: Apttus CPQ Api provides api access to the Apttus Configuration & Pricing package.							
Uninstall | Manage Licenses	Apttus Quote/Proposal Approvals Management	Apttus	6.4	Apttus_QPApprov	Active	20	19	1/1/2016

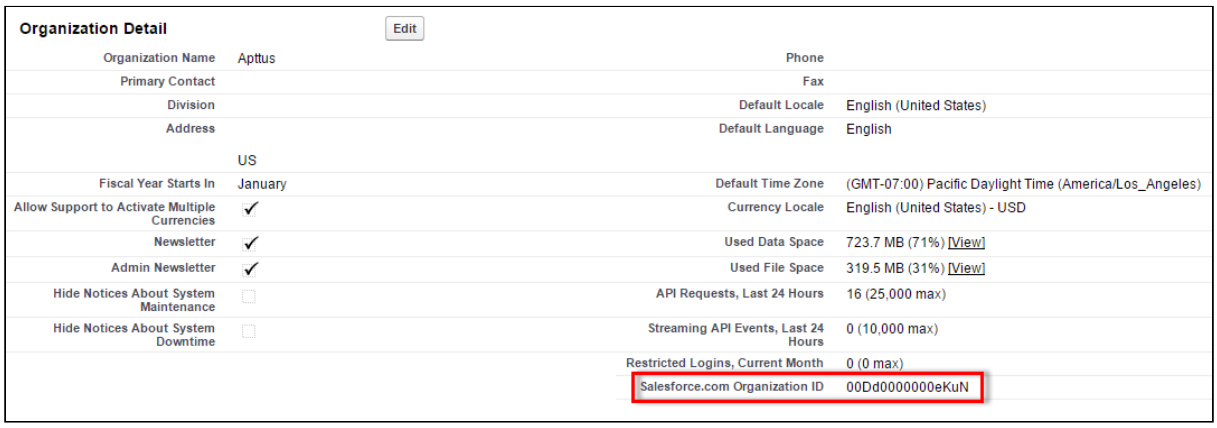
What is your Salesforce.com Organization ID?

To determine the Salesforce.com organization ID:

1. Go to **Setup > Administration Setup > Company Profile > Company Information**.



2. From the Organization Detail pane, provide the Salesforce.com Organization ID.



If you are having issues generating documents, what is your merge server end point?

To find the merge server end point:

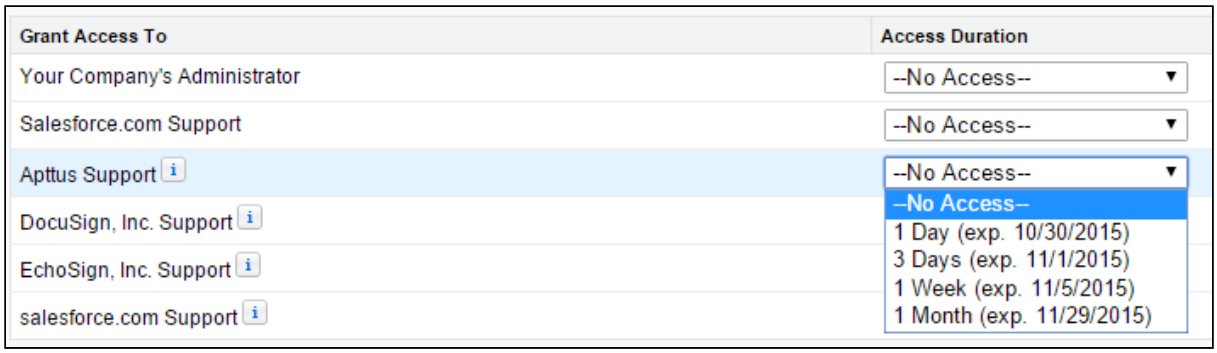
1. Go to **Setup > App Setup > Develop > Custom Settings**.
2. Click **Manage** for Comply System Properties.
3. Click **System Properties**.
4. The Merge Webservice Endpoint field displays the setting. The *https://mergeserver.apttus.net:9876* portion of the setting is what will be helpful to customer support.

Grant Login Access of the affected user and an administrator.

To grant login access:

1. Go to **Setup > Personal Setup > My Personal Information > Grant Login Access**.

2. From the Apttus Support picklist, select an option for access duration.



3. Click **Save**.

System Fields

All fields in managed objects are system fields, except for explicit override mechanisms that CPQ provides through user input or APIs.

The following are some examples of the fields for which CPQ accepts user input through the out-of-the-box CPQ UI or globally exposed APIs:

- Price List
- Quantity
- Term
- Start Date and End Date
- Coupon Codes
- Adjustments

- Base Price Override
- Product Attributes

> Except those fields that are available to be modified using the out-of-the-box CPQ UI and APIs, all other managed fields are system fields. Therefore, you must not manipulate them because of the risk of data integrity issues.

In case you must manipulate a particular system field value because of business requirements, you must contact Conga Support to evaluate if one of the following options is possible:

- Is there any existing out-of-the-box mechanism to do that?
- If not, explore the possibility of CPQ providing such a mechanism.
- If both options above are not possible, Conga Support must get approval from the CPQ Product team so that the Product team can track such fields and maintain backward compatibility when developing an enhancement or fixing any issue related to such system fields.

CPQ for Users

This section explains how to use Conga Configuration, Pricing, and Quoting (CPQ) to manage your organization's and your customers' configuring, pricing, and quoting requirements.

Topic	Description
What's Covered	This section walks the Sales Representative through the entire process of configuring, pricing, and quoting. It provides conceptual information, step-by-step instructions, and use cases for the features provided by Conga CPQ.
Primary Audience	• Sales Representative • Collaborator (such as a product owner, manager, factory manager)
IT Environment	Refer to the latest *Conga CPQ Release Notes* for information on System Requirements and Supported Platforms.
Updates	For a comprehensive list of updates to this section for each release, see the What's New in CPQ Documentation topic.
Other Resources	Refer to *Conga CPQ Release Notes* for information on system requirements and supported platforms, new features and enhancements, resolved issues, and known issues for a specific release.

This section describes the following tasks:

- Create quotes/proposals
- Configure products from the catalog
- Price products and apply adjustments
- Work with cart
- Finalize quotes/proposals
- Manage assets
- Manage services

Before using CPQ, you must be familiar with the following:

- Basic knowledge of Salesforce
- Salesforce and Conga terms and definitions

Select one of the following topics for more information:

- Getting Started

Getting Started

This section explains the tasks to be performed to get started with the CPQ.

About System Requirements

CPQ is a web-based, on-demand application that is accessed through a standard web browser through the Internet. Before you install and start using CPQ, you must ensure the minimum system requirements. For more information, refer to the latest *Conga CPQ Release Notes*.

Logging on to CPQ

Log in to your Salesforce.com org to access CPQ.

> Do not use the Back button on your browser when using CPQ.

Before you log in to CPQ, make sure you meet the following criteria.

- You have installed all of the required CPQ module packages.
- You have administrative privileges.
- You have login credentials provided by Conga.

To log in to CPQ

1. Go to http://www.salesforce.com/.
   Or
   If your organization is using a sandbox or test environment to access CPQ (for example, if you are doing user acceptance testing), go to http://test.salesforce.com/ instead.
2. From the toolbar at the top of the page, click **Login**. The login page opens.
3. Enter your user name and password, and click **Log in**.
4. Navigate to **Apttus Proposal Management** to access the proposals.
   - In Salesforce Classic: Click the App Menu
   - In Salesforce Lightning Experience: Click the App Launcher.

Now that you have logged in, know more About CPQ User Interface.

About CPQ User Interface

The CPQ Home page is the starting point of various functions in the application.

The home page consists of the following components:

> The Home page view depends on user settings.

Feature/Section	Description
Search	This feature enables you to quickly search for various items like Quote/Proposals, Templates, Accounts, Contacts, and more. You can further search within the results for specific objects using the Advanced Search feature. A search bar is available at the top of every window within the application, backed by an easy-to-use search engine. Based on the keyword you enter, the system searches against all key elements and retrieves quotes or proposals.

Feature/Section	Description
Create New	This button is located on the left-hand side of the homepage. This button provides an easy way to initiate new items like Quote/ Proposal, Opportunity, Template, Account, Report, and more.
Tabs	These are individual tabs that provide you a broader view of specific items like Proposals, Templates, Reports, and more.
Dashboard	Dashboards give you a real-time snapshot of corporate metrics and key performance indicators. A dashboard is a group of different charts (or components) that graphically display your custom report data.
My Tasks	A quick summary of tasks that have been assigned to you.
Items To Approve	A quick summary of items that have been assigned to you for approval.
Setup	This link is located on the top right-hand corner of the homepage. The link redirects you to the Force.com home page using which you can perform actions, such as view, customize, add, or edit objects and fields.
Salesforce menu	The drop-down menu provides you with the list of Conga modules that you can use for configuring, pricing, and quoting.

The CPQ Home page enables you to access quoting information in a variety of ways; Searching, Reporting, Dashboards, and Views.

- Creating Views
  Views are a very powerful, and easy way to create a virtual file cabinet of your proposals. You can create views based on dates, values, products, any data field, or a combination of fields and also have specific fields displayed. Views are a customized way to display the exact information you are looking for.
- About Reports
  Salesforce delivers standard report folders containing reports for each record type. For example, Accounts, Opportunities, Leads, and more. These reports are used as a starting point for your organization's reporting efforts. Salesforce allows all users to run, customize, and create reports. For more information on setting up and managing

reports within Conga, see salesforce.com information related to creating and customizing reports.

- About Dashboards
  Dashboards organize key statistics in easy to understand, bottom-line format, allowing you to gain the insight you need to fulfill your business needs.

Creating Views

Views are a very powerful, and an easy way to create a virtual file cabinet of your proposals. You can create views based on dates, values, products, any data field, or combination of fields and also have specific fields displayed. Views are a customized way to display the exact information you are looking for.

To create views

1. From the Salesforce drop-down menu, select **Apttus Proposal Management** and go to the **Proposals** tab.
2. Click the **Create New View** link.
3. In the **Enter View Name** section, enter a mandatory **View Name** and a **View Unique Name**.
4. In the **Specify Filter Criteria** section, specify the filter options to narrow down to your preferred view for proposals. For example, you can choose to view only those proposals for a specific Account.
5. In the **Select Fields to Display** section, from Available Fields, select one or more options and click the icon to move to Selected Fields.
6. In the **Restrict Visibility** section, select one of the following:
   - **Visible only to me**
   - **Visible to all users** (Includes partner and customer portal users)
   - **Visible to certain groups of users**
7. To save the view, click **Save**.

About Dashboards

Dashboards organize key statistics in easy to understand, bottom-line format, allowing you to gain the insight you need to fulfill your business requirements.

CPQ provides you with a set of out-of-the-box dashboards. Navigate to the **Dashboards** tab to customize the page by designating which dashboard type you prefer to see from a list. For more information, see Working with Dashboards.

The following table lists the out-of-the-box CPQ dashboards available to you:

Folder	Dashboard Name	Description
Agreement Dashboards	Contract Performance Dashboard	Displays the following graphs: • Dollars under contract • Contracts by status
	Executive Dashboard	Displays the following graphs: • Dollars under contract • Contracts by stage
	Legal Performance Dashboard	Displays the following graphs: • Agreement by Status • Standard vs non-standard agreements.
	Total Contract Value by Account	Shows the total contract value by account.
Proposal Dashboards	Proposal Pipeline Dashboards	Displays graph about proposal and proposal value by stage.
	Revenue Dashboard	Displays Revenue Management dashboard.

About Reports

CPQ provides you with a set of out-of-the-box reports. These reports are used as a starting point for your organization's reporting efforts. You can also run, customize, and create reports. You can access the reports through the **Reports** tab. CPQ provides standard reports based on the most common reporting requirements.

The following table lists the out-of-the-box CPQ reports available to you:

Folder	Report Name	Description
Agreement Reports	Agreement Expiration - This Quarter	List all the agreements expiring in the current fiscal quarter.
	Agreement Expiration - This Year	List all the agreements expiring in the current fiscal year.
	Agreement Search	Easily find your agreements.
	Agreement Value Report	List all the agreements contract value along with their Account.
	Agreement by Stage	List the agreement and their stage.
	Agreement of High Value due Renewal	Lists the date the high value agreements are due for renew.
	Agreement with Non-Standard Terms	List all non-standard agreements.
	Contract Activity Report	List the recent contract activity report.
	Contracts Expiring This Year by Value	Lists the contract expiring this year with value.
	Contracts with non-standard language	Lists all agreement with non-standard language
	High Risk Contracts	Lists agreement with high risk.
Proposal Reports	Proposals by Stage	Lists the proposal based on the Approval Stage field. The report displays the account and the grand total of the proposal.
	Proposals Outstanding	Lists the proposals that are not closed to the present date.

Folder	Report Name	Description
	Proposals Value by Stage	Lists the proposal based on the Approval Stage field. The report displays the account and the grand total of the proposal.
	Proposals with Line Items and Products	Lists the details in a proposal including proposal header details, line items details with product, pricing, and line item fields.
	RFP Content Update Report This Quarter	Lists the proposal where RFP content was updated in the current quarter
	RFP Response Content	Lists the RFP content and the owner of the content.
	RFPs by Value	Lists the proposals with RFP values
Proposal Line Items	Quote and Quote Lines	Lists proposals and proposal line item details
	Quote Details by bundle	Lists proposals and proposal line item details grouped by bundles

You can create new reports by either starting from scratch or taking a report containing the desired results, modifying it, and changing the report's name using the Save As feature. For more information on setting up and managing reports within Conga, see Create a custom report.

About Reports (Backup)

CPQ provides you with a set of out-of-the-box reports. These reports are used as a starting point for your organization's reporting efforts. You can also run, customize, and create reports. You can access the reports through the **Reports** tab. CPQ provides standard reports based on the most common reporting requirements.

The following table lists the out-of-the-box CPQ reports available to you:

Folder	Report Name	Description
Agreement Reports	Agreement Expiration - This Quarter	List all the agreements expiring in the current fiscal quarter.
	Agreement Expiration - This Year	List all the agreements expiring in the current fiscal year.
	Agreement Search	Easily find your agreements.
	Agreement Value Report	List all the agreements contract value along with their Account.
	Agreement by Stage	List the agreement and their stage.
	Agreement of High Value due Renewal	Lists the date the high value agreements are due for renew.
	Agreement with Non-Standard Terms	List all non-standard agreements.
	Contract Activity Report	List the recent contract activity report.
	Contracts Expiring This Year by Value	Lists the contract expiring this year with value.
	Contracts with non-standard language	Lists all agreement with non-standard language
	High Risk Contracts	Lists agreement with high risk.
Proposal Reports	Proposals by Stage	Lists the proposal based on the Approval Stage field. The report displays the account and the grand total of the proposal.
	Proposals Outstanding	Lists the proposals that are not closed to the present date.

Folder	Report Name	Description
	Proposals Value by Stage	Lists the proposal based on the Approval Stage field. The report displays the account and the grand total of the proposal.
	Proposals with Line Items and Products	Lists the details in a proposal including proposal header details, line items details with product, pricing, and line item fields.
	RFP Content Update Report This Quarter	Lists the proposal where RFP content was updated in the current quarter
	RFP Response Content	Lists the RFP content and the owner of the content.
	RFPs by Value	Lists the proposals with RFP values
Proposal Line Items	Quote and Quote Lines	Lists proposals and proposal line item details
	Quote Details by bundle	Lists proposals and proposal line item details grouped by bundles

You can create new reports by either starting from scratch or taking a report containing the desired results, modifying it, and changing the report's name using the Save As feature. For more information on setting up and managing reports within Conga, see Create a custom report.

About Dashboards (Backup)

Dashboards organize key statistics in easy to understand, bottom-line format, allowing you to gain the insight you need to fulfill your business requirements.

CPQ provides you with a set of out-of-the-box dashboards. Navigate to the **Dashboards** tab to customize the page by designating which dashboard type you prefer to see from a list. For more information, see Working with Dashboards.

The following table lists the out-of-the-box CPQ dashboards available to you:

Folder	Dashboard Name	Description
Agreement Dashboards	Contract Performance Dashboard	Displays the following graphs: • Dollars under contract • Contracts by status
	Executive Dashboard	Displays the following graphs: • Dollars under contract • Contracts by stage
	Legal Performance Dashboard	Displays the following graphs: • Agreement by Status • Standard vs non-standard agreements.
	Total Contract Value by Account	Shows the total contract value by account.
Proposal Dashboards	Proposal Pipeline Dashboards	Displays graph about proposal and proposal value by stage.
	Revenue Dashboard	Displays Revenue Management dashboard.

Creating Quotes

- Creating Opportunities in Salesforce
  The CPQ process begins as soon as you create an opportunity.
- Creating Quotes from Opportunities
  CPQ provides you the capability to enter necessary information via a user-friendly interface to prepare a quote/proposal request or a draft.
- Cloning Existing Quotes
  You can clone the existing Quote/Proposal using the Clone button on the Quote detail page.
- Creating Quick Quotes
  You can use the Quick Quote mode to quickly select numerous products from the product selection page directly from an opportunity with just one click.
- Including Product Footnotes in Proposal Documents
  Product footnotes enable you to dynamically include static *footnote* content in your quote/proposal documents, based on the products that are included in the quote.

Notes:

Displaying Quote Name in the Application Browser Tab
CPQ shows a quote or proposal name for ease of navigation through the browser tabs.

- It shows the Quote/Proposal name when you are on the Catalog page, Installed Products page, Attribute/Options page, Cart page, Analyze Quote page, and Proposal Document Generation page.
- It shows the Collaboration ID in the browser tab for a Quote Collaboration Request.

Displaying CPQ Quote ID in the Application Browser Tab
When you create a quote/proposal, CPQ used to autogenerate a Salesforce Quote ID. To improve the user experience you can see a new CPQ specific Quote ID in the browser tab. On the following pages, CPQ shows the browser tab name according to the name of the quote/proposal: Catalog, Installed Products, Options or Attributes, and Analyze Quote.
Displaying Quote/Proposal Names instead of Numbers
CPQ displays the name of the Quote/Proposal adjacent to its label at the top left corner of the page. If the proposal name is too long then you can see up to 26 characters and then an ellipsis to expand the full name. While if you hover over, you can see the whole text.
You can see the proposal names on the following pages: Catalog, Installed Products, Renew, Terminate, Swap, Service Catalog, Attribute or Options, Cart Grid, Service Cart, Analyze Quote, and Compare Products.
Obsolete Quote Header Fields
CPQ does not use the following fields in any feature. They are displayed on the quote header but they are not functional.

- **Grand Total**
- **Net Amount**
- **Unit Price**

Creating Opportunities in Salesforce

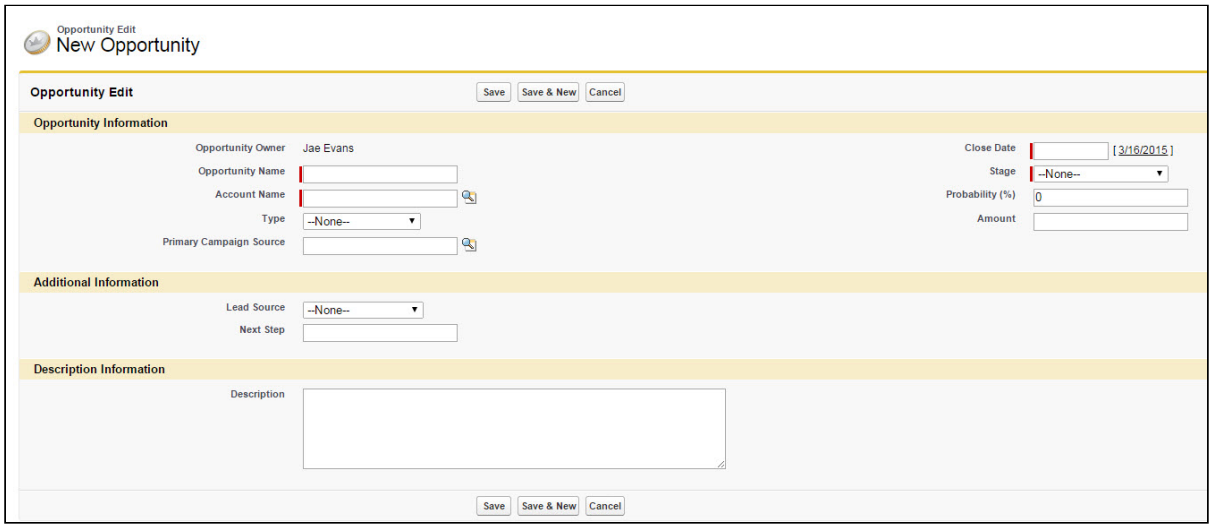
Opportunities are the qualified contacts or accounts that you have already talked to and have entered into your sales cycle. Adding opportunities to Salesforce builds your pipeline and increases your sales forecast. You will use the Salesforce Opportunities tab to create and find your opportunities.

The CPQ process begins as soon as you create an opportunity.

> ⓘ Opportunities are standard Salesforce objects, not custom objects. For more information on opportunities, see Salesforce documentation.

To create an opportunity

1. Log in to Salesforce.com.
2. Navigate to the **Opportunities** tab, and click **New**.



3. Enter the **Opportunity Name, Account Name, Close Date,** and **Stage**.
4. Enter additional information as required.
5. Click **Save**.

You have created an opportunity. For complete information about creating opportunities in Salesforce, refer to Create Opportunities and Opportunity Fields in the Salesforce documentation.

Creating Quotes from Opportunities

You can use the user-friendly interface to enter the necessary information and prepare a quote/proposal request or draft. To complete the quote/proposal, you must enter details relevant to a proposal. Some of the fields identified with (red vertical line) are required fields, without which you cannot save the proposal record. If the required details are not entered or incorrect details are entered, the system displays validation rule errors at the top of the page highlighted in red.

To create a quote/proposal

You must be logged into Salesforce.com. A series of colored tabs are aligned along the top navigation area.

1. From the Salesforce.com application menu, select **Apttus Proposal Management**.
2. Go to the Proposal tab and click **New.**
   - OR -
   Go to the Opportunities tab and click the **Create Quote/Proposal** button.

   > If you have installed CPQ for the first time, use **Edit Layout** to drag the button to the layout.

3. From Record Type of new record, select the desired record type, and click **Continue.** You can add any type of Quote/Proposal template (Budgetary, Final, and more). The out-of-the-box system only has the Proposal and RFP type. If quote/proposal types are not set up or if you have access to only one quote/proposal type, skip to the next step.
4. Type a mandatory **Proposal Name** and enter a mandatory Opportunity.
5. Enter a Price List and an Account.
   The Price List determines the categories and products you see in the configuration. Account is auto-populated if the proposal is created from an opportunity.
6. Select a value for **QTC Profile**. The following table describes the values of the field:

Value	Description
Regular	This is the default value of the field, the cart pricing and finalization are processed in sync mode by default. You can override this behavior by using URL parameters.
Split	Select this to enable Smart Cart flow for the quote. When you select this, the cart pricing and the finalization are processed asynchronously.

Value	Description
Enterprise	Select this to process the cart finalization asynchronously for a quote with upto 2000 line items. The administrator must add *Enterprise* as a picklist value in the **QTC Profile** field in the objects Quote/ Proposal, Product Configuration, and Agreement for CPQ to display the value on the quote. For more information, see Enabling Enterprise as a QTC Profile Value.

7. An Opportunity may have many proposals or quotes. To set this record as a primary quote, select the **Primary** checkbox.
8. In **Primary Contact**, enter the primary contact person name for the account.
9. Select the **Single Transaction Adjustment** checkbox to control the adjustment of the billing based on the context in which the discount is applied. When you select this checkbox on a proposal, billing schedules are adjusted from the start of the contract term. For more information, see Calculating Billing Amount based on the Contract Term and the Single Transaction Adjustment Flag.
10. From the **Intent** drop-down, select what you want to achieve after the quote is accepted. The supported values are:

Intent	Purpose
Order	CPQ creates a regular flow, that is, order and assets. If you do not select any value from the **Intent** drop-down, CPQ considers *Order* as the default value.
Contract	CPQ does not create order and assets. Instead, CPQ will create an agreement as a related list to the proposal with Intent = Contract. You must use this option when cart configuration must be driven by agreements. • Auto-created related list Agreement (to proposal with Intent = Contract) will have auto-finalized configuration. • Auto-created related list Agreement (to proposal with intent = Contract) will have agreement line items while having the same price and adjustments carried in quote with Intent = Contract.

Intent	Purpose
Price Agreement	CPQ does not create order and assets. Instead, CPQ will create a price agreement, which will be used for defining contract pricing for future quoting. Since the intent is price agreement, your agreement price rules will be generated, and also a new price list will be generated for you containing price list items related to each one of the line items in your cart related to proposal. The Net Unit Price (after all adjustments) of products on the quote with Intent = Price Agreement will be applied as list price on products during the new quote if the original quote number (earlier used as Intent = Price Agreement) is applied as the Contract Number on the new quote.

11. In the **Purchase Identifier** field, enter a unique identifier.

> A purchase identifier is populated to cumulate the quantity, coterminate assets during quantity changes, bind various asset streams together, or add quantity upsell to the original renewal quote during renewal automation. This purchase identifier floats from quote to cart line items > proposal line items > order line items > asset line items. This purchase identifier is used to calculate the total quantity that spans multiple streams during the increment scenarios, on the Installed Products page (of a renewal quote).
> For example, you enter PI-01 in the **Purchase Identifier** field and the account has the following assets:
> - Asset A
> - Asset A
> - Asset B
>
> Then the total quantity of both Asset A and Asset A = 2, and the total quantity of Asset B = 1. For more information, see Managing Asset Increment with Coterminate Lines.

12. From Approval Stage, select the stage of the proposal. The approval stage determines the actions available to you for a quote/proposal. For detailed information on various approval stages and the actions available, refer to Glossary of Approval Stages.
13. To enable the final quote or proposal generation from the current proposal, select **Ready to Generate**. The **Ready to Generate** checkbox is selected by default. If it is not selected, the **Generate** button does not appear in the proposal.

14. To attach and send the proposal in an email to a configured account, select Ready to Present. The Ready to Present checkbox is selected by default. If it is not selected, the Present button does not appear in the proposal.

    > Before saving the quote/proposal, ensure that you have set the **Pricing Date** that falls in between PLI expiration and effective dates. Otherwise, Quote cannot pick the PLI with a pricing date from an expired PLI.

15. Click **Save**.

You have created the draft proposal which you can use for searching and reporting. Various new related lists are enabled on the quote/proposal page. To know more, refer to Glossary.

Navigate to the proposal you just created and proceed with product selection.

> **Create quotes from opportunities inheriting the related line items**
>
> You can create a draft quote with the selected service product and its related line items (asset line items) and then synchronize it with the opportunity. After clicking the synchronize with opportunity, you can also cascade the related equipment lines to the opportunity and forecast the service revenue in conjunction with the opportunity.
> Similarly, you can create a renewal quote that carries over the related line items to the quote or cart.

Cloning Existing Quotes

You can deep clone the existing quote using the **Clone (with line items)** button on the Quote detail page. Cloning a quote copy's all the information from the existing quote, including line items and proposal header information asynchronously. You can deep clone a quote when you need multiple draft options of the same quote or need to make changes (for example, extend the quote expiry date) to a finalized quote before sending it out to the customer.

You can clone the quote in the following product configuration statuses:

- Pending Approval
- Ready for Finalize
- Approval Required
- Saved
- Finalized

> ⓘ Quotes with **QTC Profile** defined as *Split* are cloned regardless of approval status as Smart Cart does not support cart approval.

The new cloned quote retails the information based on the configuration status and **QTC Profile** defined. The following table lists the information copied upon cloning the quote:

QTC Profile	Product Configuration Status	Information Retained in the Cloned Quote
• Regular • None	Finalized	• Proposal header details • Proposal Line Items
	All statuses except Finalize	• Proposal header details • Configuration Line Items
• Enterprise • Split	All statuses	• Proposal header details • Configuration Line Items

The following information is not retained from the parent quote:

- Activity history
- Notes and Attachments

Cloning is processed asynchronously, CPQ tracks the different states of the process using the Async Progress Tracker.

To clone an existing quote

1. Navigate to your existing quote.
2. Click the **Clone (with line items)** button.
3. Fill in the required details in the cloned quote and click **Save**.

When you clone a quote with its line items, a new quote is generated with a unique quote number whose **Status** is *Draft,* and **Configurations** status is *Saved*.

If an approval trigger is set on the existing quote, the approval status and trigger are not copied to the cloned quote. You must re-trigger approval on the cloned quote. You can use the custom setting **Clone With Approval** to alter the approval status of a cloned quote. For information, see Clone Quote with its existing configuration.

Creating Quick Quotes

The Quick Quote feature allows you to create a simple quote from opportunity in fewer clicks. You can directly go to the Catalog page from Opportunity, configure products, and finalize the cart without having to create a proposal in turn reducing the number of clicks. Quick Quote feature is especially meant for creating budgetary quotes where you can begin the quoting process by populating the quote header with limited values.

After you complete the configuration on the Quick Quote you created from the opportunity, you are redirected to the Quote detail page.

When you create a quick quote, the following fields are populated based on the button's URL:

- Opportunity ID
- Price List
- Account
- Expected Start Date
- Expected End Date

The proposal name is set the same as the opportunity name. In addition to that, the custom fields that are common between both opportunity and quote are also populated, these custom fields are defined with the same API name.

The administrator must create the **Create Quick Quote** button, refer to Configuring Quick Quote Mode.

To create a quick quote

1. Navigate to the Opportunities tab, select an existing opportunity, and click **Create Quick Quote**. The Catalog page is displayed with products from the price list associated with that opportunity.
2. Search and configure products, add them to the cart, and finalize the cart.

After you finalize the cart, you are redirected to the Quote Details page of the quote created when you click **Create Quick Quote**. You are also redirected to the Quote Details page if you save, abandon, or close the cart. Your quote is ready using the Quick Quote mode.

Including Product Footnotes in Proposal Documents

Product footnotes enable you to dynamically include static *footnote* content in your quote/proposal documents, based on the products that are included in the quote.

To include product footnotes in Proposal documents

1. Login to X-Author Contracts, select the **X-Author Templates** tab and check out (or create a new) a proposal document template.
2. Place the cursor in the template where you want the footnotes and click **Insert Merge Fields**.
3. Select Lookup Fields, and then select **Product Footnote Fields** > **Footnote** > **Text** and click **Insert Field**. The merge field is entered into the document template.
4. Click **Check-In** and ensure that Type is **Proposal** and that the selected Agreement Types also match those of the proposals you will use the template with.

When you Finalize your shopping cart and subsequently Preview or Generate the proposal document, you should see the text associated with the product footnotes – for the products included in the Line Items – displayed in the document.

Using the Catalog

The following features are available on the product catalog page.

- Searching Products from the Catalog
- Using Refine Your Search
- Comparing Products on the Catalog
- Using Guided Selling

Searching Products from the Catalog

The Catalog page enables you to select and configure products. This page allows multiple ways to search and choose the product you are looking for. You can select products by using the **Browse Catalog list**, **Browse**, or the **Search bar**. When using the search bar, you can search for any product regardless of categories. However, the products must have their

respective Price List associated with the Quote. If Smart Search is enabled in your org, typeahead results available while searching. You can search for products by:

- Product Code
- Product Family
- Product Description
- Product Configuration Type
- Category Names
- Location

In the search bar, enter the desired product and click **Search**.

> As per the Salesforce limitation, the Text Search in the **Search bar** does not support wildcards at the beginning of the search term. For example, there are 3 products in your Catalog, namely Cat1_Product1, Cat1_Product2 and Cat2_Product4. If your Search Term is **Product1*, the search returns 0 result and if your search term is *Cat1**, the search returns 2 results (Cat1_Product1 and Cat1_Product2).

Based on your selection from any of the above-mentioned sections, the products are displayed in the Product Catalog section in the center. You can use the Refine Your Search feature to narrow the list of products. For more information, see About Refine Your Search.

Searching Product or Category

You can search for a product by doing any of the following:

- From the **Browse Catalog** list, select a category hierarchy you want to view.
- From the Browse section, navigate to the category hierarchy for the desired product.
- In the Search bar, type a category or product name. When you enter the name of a product you can see the typeahead results if the product name has a match. Click **Search**. The result is displayed in alphabetical order of Product Name unless Smart Search is enabled. The products are sorted based on the weight configuration of the Smart Search. The products with higher weight are displayed first.

  > If Smart Search is enabled in your org, note the following:
  > - Typeahead results are displayed only for products and not categories.
  > - The typeahead is case sensitive.
  > - Ensure that your search term does not contain hyphen (-).
  > - Typeahead results are not displayed if the remote server is not functional.

- Sort the products in the Catalog page using the **Sort** drop-down. Click the sort icon () to see the available fields for sorting in the dropdown. You can select to sort the product in either ascending or descending order of the predefined fields. The following options are available to you in the **Sort** drop-down on the Catalog page by default. Your administrator can define more fields in Config System Properties:
  - *Product Name Ascending*
  - *Product Name Descending*

    > The administrator can configure product sort to add more fields. For more information, see Configuring Product Sort. The **Sort** drop-down is hidden if Smart Search is enabled.

Based on your selection, the products are displayed in the Product Catalog section. If you have Refine Your Search configured, all the filter fields related to the selected category and Search bar are displayed dynamically in the Refine Your Search section. The system displays the products you search using the Search box, Refine Your Search, and Browse Catalog feature collectively.

Proposal Locations are defined to differentiate the products as per your customer locations. Depending on the Account and Proposal Locations defined for your Quote/Proposal, the system displays products on the catalog page:

- If no account is selected on Quote/Proposal, the system does not filter any product.
- If an account is associated with your Quote/Proposal, the system return displays account locations (in the drop-down list) that are associated to the Account.
- If no proposal locations are defined, the system filters the products based on the account (previous 2 points).
- If you have defined proposal locations on your Quote/Proposal, the system displays only proposal locations in the drop-down list.
- If you add a product to the cart after you defined a location, any product that is auto-included through a constraint rule inherits the location you defined for the condition product.

You can further filter the product list by selecting the filter values from the **Refine Your Search** section. These filter fields change dynamically based on your Category and Search to further filter the products.

> If you perform another search again, all your previous search selections are reset.

The product is added to the Selected Products panel on the right. If a product is configurable, you can configure a product with multiple options or attributes.

Using Refine Your Search

When you are selecting items from the Product Catalog, **Refine Your Search** enables you to narrow the list of products to those that match your search filter questions.

Search filters are associated with categories, so that the *Refine Your Search* questions may change when you select different product categories. The questions are also found in the product record, which is how the Product list gets refined. When you select an answer to a question and that answer matches the answer on the product record, that product is included in the filtered Product list. For more information, see About Refine Your Search..

> For a product to be displayed on the Catalog page, it must match all of the selected Refine Your Search criteria.

To refine your Search

You must have configured Custom fields for each search question.

1. Select a Quote/Proposal and click **Configure Products**.
2. On the Catalog page, search for a product. Based on your selected category, all the products are listed in the Product Catalog section.
3. The **Refine Your Search** window displays the search filter questions. Select a value to answer the questions that best match your requirements, and based on your selection, the list of products are further filtered to narrow down the list.

The displayed products list will refresh to only display those products which match the search criteria. Products must match all criteria to be displayed.

Additional selections you make in the Refine Your Search window will further filter the products list.

Comparing Products on the Catalog

Product comparison allows you to compare features of multiple products on the catalog page enabling you to narrow down to products that best fit your customer's needs.

To compare multiple products

1. On the Catalog page, select more than one product to compare.
2. Click **Compare Products**.

A Product Compare page appears listing the selected products and its features in the tabular form. You can show/hide the products to display on the comparison page.

- The system indicates the number of products that you have chosen to compare inside parenthesis on the **Compare** button. This helps you keep a counter of the number of products chosen for comparison on different pages on the catalog page. After product selection, the system also indicates how many products you have chosen for comparison, as *'n' selected*. The products can be chosen from different categories also.
- You can remove a product from the selection by clicking  icon adjacent to each product.
- The products are listed in the Compare Products selection dialog box based on the order of your selection and not in an alphabetical order.

To go back to the catalog page, click **Return to Catalog**.

The **Return to Catalog** button now takes you back to the last category that you had browsed for product selection. For example, you had selected Product 1 from Category A and Product 2 from Category B. Once you are on the Product Comparison page, clicking **Return to Catalog** takes you to Category B and the catalog page (containing all the categories).

Using Guided Selling

This feature has been deprecated since Spring '21 release.

Adding Products

The Quote creation process involves product selection, configuring attributes, configuring options, and pricing the products. Pricing can be done only after you have added products to the shopping cart. The Quote/Proposal detail page is the gateway to select and configure your products, and later price and finalize them to generate a Quote/Proposal. When you configure products, Product Configuration is created on the Quote/Proposal detail page that holds the details about the product you configure on the cart including the options and attribute configuration and pricing configuration.

The Quote creation process comprises the following four steps:

- Select: This stage of the process enables you to select products from the Catalog page. You can select and add Standalone and Bundle products directly to the shopping cart or choose to configure products that have attributes or options

associated with it. While doing this, you can also view the Product Summary by clicking a specific product, which lists the Product Code, Product Description, and whether a product has attributes and options associated with it.
- Configure: On the Catalog page, if you choose to configure products, you can select attributes from the Attributes page or select options from the Options page.
- Price: After you are done with your product selection or configuration, you can price them on the Shopping Cart page.
- Finalize: Once your cart is complete, you can finalize the cart and your quote is ready to be generated.

Depending on your profile or any other business use case set for your organization, when you click **Configure Products**, you may view different versions of the following:

- Catalog page, Attributes page, Options page, Cart page, and Assets page.
- Action buttons like **Next**, **Reprice**, **Save**, and more.
- Columns on the Cart page.

> If your administrator implemented contract pricing, use the **Configure Products (Contract Pricing)** to add products to your quote. For more information, see Managing Contract Pricing.

This section describes the following topics:

Configuring Products from the Catalog

If you have attributes and/or options associated to your products, you can configure them from the Catalog page.

Configure Product Attributes - You can select and configure attributes for your products from the Attributes page.

Configure Options for bundle product - You can select and configure options for your products from the Configuration page.

If the system is configured such that you have to include one or more products along with your regular product selection, you may view the following possibilities on the Catalog page and Configuration page:

- If there is only one product, it is auto-included automatically and you will see only the **Add to Cart** button.
- If there is an option to select only one product from a product group, you will see a radio button next to each product and the **Add to Cart** button.
- If there is an option to select multiple products, depending on the minimum/maximum number of products set for your organization, you will see a checkbox next to each product and the **Add to Cart** button. This enables you to select multiple products at once.

⚠
- If no option products are visible due to Visibility rules configuration or different price lists, and the administrator have set the minimum options as zero, the system considers this configuration as invalid. Ensure that your administrator has set the right configuration for the Min/Max option for an option group.
- CPQ does not display a message when you hover over an option product which is disabled on the configuration page. As a part of notification and error handling, CPQ provides a defined Rule Messaging for exclusion rules. If you run the exclusion rules successfully, no hover over message is displayed, unlike in the case of inclusion rules where such messages are helpful for you to make a decision about adding products to the cart.

You can see the following buttons on the Configuration page:

Action	Description
Validate	Validates that the Catalog is configured correctly, that is, there is no validation or other rule-related errors in catalog configurations. This button ensures that the constraint and pricing rules are processed for your products. While the configurations are validated, the **Go to Pricing** button is disabled on the Configuration page. After the configurations are validated, the **Go to Pricing** button is enabled. After you click **Go to Pricing**, the Cart page is displayed.

Action	Description
Go to Pricing	Calculates the price of products on the Configuration or Catalog page, and navigates you to the Cart page. When your administrator has enabled the **Bypass Shopping Cart** setting, this button is hidden on the Configuration and Catalog pages. However, you can still finalize the quote using the Mini cart.
Add More Products	Returns to the Catalog page to add more products.
Installed Products	Navigates to the Installed Products page. You can select products from a list of purchased products that are associated with the account, which is associated with the proposal. This enables you to select products directly that are previously purchased by the account.
Update Price	Calculates the price of a bundle and its options on the Configuration page. This button is enabled whenever you change field values on the Configuration page and is disabled once the pricing is updated.
Abandon	Abandons the current cart (deletes the cart).
Close	Closes the cart without saving any changes to it.
Edit	The **Edit** button in the menu under the proposal name enables you to edit the Quote details.
View	The **View** button in the menu under the proposal name enables you to view the Quote details.
Confirm Option Selections	Initiates calculation of pricing, expression, and execution of rules on the Configuration page. Click this button after you configure all the options and attributes. This button is only displayed if the administrator enables it.

To configure product attributes

1. Follow one of the following ways to find a product on the Catalog page:
    - From the **Browse Catalog** list, select a category hierarchy you want to view.

- From the Browse section, navigate to the category hierarchy for the desired product.
- In the Search bar, type a category or product name and click **Search**.

2. From the Product Catalog section, to configure attributes for a product, click the **Configure** button.
3. Configure the required attributes. You can search for attributes using the search bar next to the bundle name. Type full or partial text and click **Next** or **Previous** or press *Enter* to search the attribute. The search result is highlighted under the respective tab.

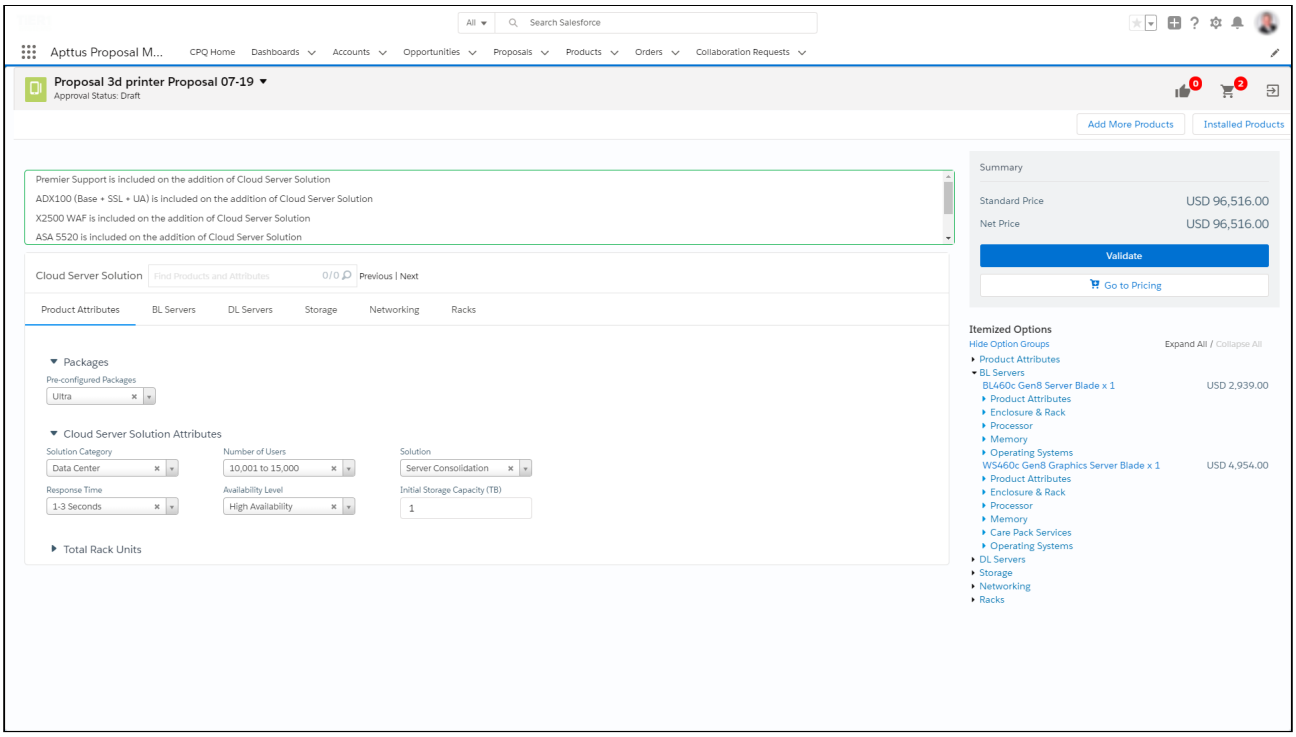
   > The values of attributes can depend upon the selection of other attributes. The visibility of values in the attributes is based on the dependent attributes. If you select a certain value in an attribute, only the relevant values are displayed in the dependent attribute. You must remove or select a different value in the attribute to see different values in the dependent attributes.
   > If you want to see all the values of an attribute after the values are filtered, click the cross icon ( ). Regardless of the values of dependent attributes, all the values are displayed and when you select a different value, the dependent attributes are filtered based on your selection. For example, the values of Attribute B depend on the selection of Attribute A. When you select a value in Attribute A, only relevant values are displayed and all other values are hidden on Attribute B. If you change the value of Attribute B, the value of Attribute A changes automatically.
   > While changing attributes, you must wait until the constraint rule execution is complete. If you keep changing the attribute value multiple times and switching between the Product Attributes tab and the Product Options tab on the Configuration page, the Product Options tab disappears intermittently.

4. The following actions are available for you once you select the attributes.
   - Click **Next**. This button appears in case of bundled products where you may need to configure or add additional products.
   - Click **Add More Products**, to add additional products.
   - Click **Remove Item**, to remove a product.
   - Click **Go to Pricing** to price the product.
   - Click **Validate** to verify your configuration.

   The values for an attribute can be driven by the numeric expression builder configured for the selected product. While the constraint and pricing rules are processing in the system, the **Go to Pricing** button is disabled on the Configuration page.

Based on your selected attributes, the system saves the product in your shopping cart.

- On the Product Attribute Group Detail page, you can set the Two or Three Column Display for the product attributes in order to view the configured attribute groups in two or three-column format.
- CPQ displays a red border on the left when attributes are set required in the UI.



Attributes can derive the price for a product from an attribute without additional configurations to the Attribute Group for that attribute.

Consider a product Internet Plan, with three attributes - Internet Speed, Data Limit, and Package Price. On configuring the product from the cart page, the user enters the value for Internet Speed and Data Limit. Create an attribute Package Price, for which the value is calculated using the formula Internet Speed * Data Limit using field expression builder. The Package Price attribute value will not be shown to the user on the ProductAttributeDetail, ConfigureBundle, and ConfigureOptions pages. The user can choose to derive the Base Price of the product Internet Plan from the Package Price Attribute Value using the Default Price From field from the Default tab of the Price List Item.

The attributes calculated using formula fields are hidden from the following pages:

- Apttus_Config2__ProductAttributeDetail3- Attributes for Bundle or Standalone products
- Product Attribute Detail page
- Page Configure Bundle page - Attributes for Options
- Select Config Options Detail View- Attributes for Options shown in line with the option

- Configure Option page.

To configure bundle products

1. You can select a product by doing any of the following:
    - From the Browse Catalog list, select a category hierarchy you want to view.
    - From the Browse section, navigate to the category hierarchy for the desired product.
    - In the Search bar, type a category or product name and click **Search**.
2. From the Product Catalog section, to configure options for a product, click **Configure** next to it. The selected product is displayed in the Selected Products section on the right and the options page is displayed.
3. Select the options for your bundle. You can search for products using the search bar next to the bundle name. Type full or partial text and click **Next** or **Previous** or press *Enter* to search the product in various option group tabs. The search result is highlighted under the respective tab.
4. The following actions are available for you once you select the options.
    - Click **Add More Products**, to add additional products.
    - Click **Remove Item**, to remove a product.
    - Click **Go to Pricing** to price the product.
    - Click **Save** to save your selected product for that quote.
    - Select **Optional** to consider a particular line item as an optional one.

    The values in the **Quantity** field for options can be driven by the numeric expression builder configured for the selected option.
    The **Save** button is used in case you want to save your configuration before proceeding to the attributes and cart page. For example, if you change the **Quantity** of the options associated with your bundle product, you might want to save the configuration before moving to the cart page.

> The administrator must enable the search bar on the Configuration page. For more information, see Configuring Config Page Settings.
> Hidden options and attributes are not displayed in the search result.

Configuring Optional Line Items

You can mark bundles, sub-bundles, and options in the bundle as optional. This helps you to give customers a choice and recommend the product to them as part of the configuration. You can present these optional products separately in a proposal document or keep them

as a part of the configuration. Click **is Optional** checkbox next to a product to mark them optional.
Whenever you mark any product as optional, the price of the product is not included in the total. If you mark a bundle as optional, the price of that bundle is not included in the total and if you mark an option in a bundle as optional, the price of that option is not included in the bundle price.

Keep in mind the following when you mark options or bundles as optional.

- When you mark a bundle as optional or clear the **is Optional** checkbox, the action is cascaded to all options and sub-bundles associated with that bundle. That is, if you mark a bundle as optional, the options and sub-bundles are also marked as optional. Though, you are allowed to edit the **is Optional** checkbox for the associated options and sub-bundles.
- You can edit **is Optional** checkbox for options or sub-bundles added because of rules.
- You can edit **is Optional** checkbox for options or sub-bundles that are marked as required only if the parent bundle is marked optional.
- You cannot edit **is Optional** checkbox for options in the radio button group, unless the parent bundle is marked optional.
- The options for which you selected **is Optional** checkbox are not considered in the Option Group Min/Max validation. You must select other options in that Option Group that are not marked optional to satisfy the Min/Max validation.
- The options in an optional bundle which is marked by the **is Optional** checkbox, are not considered in the Option Group Min/Max validation regardless of the **is Optional** value for the individual options themselves. The details like amount and quantity are not considered in the validation.
- You can edit **is Optional** checkbox on the Cart page as well.

  > Optional products are not considered in the Min/Max criteria only if **Exclude Options Products** was enabled by the administrator in Config System Properties. For more information, see Excluding Optional Product in Min/Max Criteria.

The **Validate** button ensures that the constraint and pricing rules are processed for your products. While the configurations are validated, the **Go to Pricing** button is disabled on the Configuration page. Once validated, the **Go to Pricing** button is enabled.

After you click **Go to Pricing**, the Cart page is displayed.

You can clone a bundle within the same quote, make modifications, and use it as another bundle.

This feature enables you to clone a bundle with attributes, options, and pricing, make desired changes and save it as another bundle.

Min-Max Messaging: On the product configuration page, when you make a selection of options in the desired Option Group, CPQ displays the min-max messages guiding you about how many options can be selected. For example, If you want to select X number of options where options can be X to X+n (n=1,2,3...). Then you will see an appropriate message for selecting those options from the desired Option Group. Optional products are not considered in the Min/Max criteria if **Exclude Options Products** is enabled in Config System Properties.

Horizontal Scroll for Option Groups Tab: The Product Configuration page provides a horizontal scroll bar when you have many option groups. You can see a single line of tabs for option groups and for menus with many option groups, you can also see the submenus with option groups being available for selection.

Increased Threshold for Bundle Maintenance Job: You can run Bundle Maintenance Job without failure when CPQ has a large number of options validated with the following scenarios.

- 10,000 options groups for the entire system
- 10,000 options in a given option group
- 2,000 option groups for a given bundle with an average of 3 to 5 options each group

To clone a product

You must have an existing product in the cart.

1. On the Cart page, select products using the checkbox.
2. Click the Copy icon (). The product line items are cloned on the cart page.
3. You can configure the products and make the desired changes.

If you clone bundles with complex structure multiple times, CPQ may encounter *STRING_TOO_LONG* or *Apex CPU Time Limit Exceeded* error. The errors may also be encountered if the administrator has defined Option Filter Callback.

You have created another bundle from an existing bundle. You can proceed to pricing.

To lock an option value

A lock icon appears when the quantity of the option is derived using a numeric expression.You can click the lock icon next to the option, to ensure that any updates to the field from which its value is derived do not affect the value set the first time. For example,

you can lock the quantity field for an option that is populated based on multiple attribute values using the field expression builder, once you enter the attribute value on the Attribute Detail page and click **Next**, the value for quantity is auto-populated.

1. Select the option and click the Lock icon.
2. Navigate back to the Attribute detail page and edit the attribute value and click **Next**. Notice that the value of the Option does not change even though the attribute value has changed.

- If you skip any mandatory configurations, the status of your configuration is shown as pending with a red exclamation mark on the Cart page. You must complete the pending configurations in order to finalize the Cart.

When a product is added by a rule that populates the AddedByRuleInfo__c field on the line item with a value, CPQ does the following:

- Clears the **is Optional** checkbox if **is Required** checkbox for the line item is selected at the Option Group level.

In both cases above, you can edit the **is Optional** checkbox.

To select products from prompt

The prompt appears on the Catalog page when you add a product to the cart that cannot be sold alone, but other specific products must be sold with that product. When you add such product, CPQ requires you to select a minimum or a maximum number of products from the prompt to add to the cart. The product listed in the prompt is based on the inclusion constraint rule defined by the Administrator.

There are two types of prompt that appear on the Catalog page.

- With checkboxes: In the prompt with checkboxes, select the checkbox against the product you want to add to the cart and click **Add to Cart**. The **Add to Cart** button is not enabled until you select at least a minimum number of products and the remaining products are disabled when you have selected a maximum number of products. The prompt is closed when you click **Add to Cart**.
- Without checkboxes: In the prompt without checkboxes, you need to click **Add to Cart** button against the products. After you click **Add to Cart**, the product is removed from the prompt. When you select less than the maximum value, you have to manually close the prompt because the prompt closes automatically only when the maximum number is reached.

After you add the product to the cart, the counter on the mini-cart icon at the top right-hand side corner is updated. The number of products in the cart is reflected on the counter.

If there is a recommendation constraint rule defined on any product you added to the cart, the counter on the recommendation icon is also updated.

About Favorite Configurations on the Cart

As a Sales Rep, you can use a saved configuration to add products to the cart and also save an already configured cart and use the same configuration multiple times. These configurations are saved using Save as Favorite feature. Favorite configuration enhances the product configuration and quoting process by providing the user the ability to create, share, and import favorite configurations or designs.

The saved configuration also includes the products added through constraint rules and default options that are auto-included. You can add a favorite to the cart same as you would add any other product from the catalog. You can add multiple saved configurations to the cart.

The favorite configuration can be private or public. A private configuration is saved to your library and is only visible to you. However, public configurations are displayed on the Catalog page of all other Sales Reps as well.

You can also categorize the saved configuration using the filters provided on the Save as Favorite pop-up on the Cart page. You can use these filters to search for the saved configuration on the Catalog page similar to how you search for products using Refine Your Search.

As a Sales rep, you can use a saved configuration to add products to the cart and also save an already configured cart to the same configuration multiple times. These configurations are saved using Save as Favorite feature. Favorite configuration enhances the product configuration and quoting process by providing the user the ability to create, share, and import favorite configurations or designs. The favorite configuration can be private or public.

> The administrator must configure the Save as a Favorite feature. For more information, see Configuring Save as Favorite. The visibility of the functionalities provided in this feature depends on the permissions defined by the administrator for your user or profile.

Vertical industries in the quote to cash domain require a capability of favorite configurations. This feature is useful for the following scenarios:

- Speed up quoting process – you can save frequently configured solution, which is complex in nature for its re-use at a later time.
- Dynamic bundles – as a product manager, you have the ability to create, save and import a saved configuration as a whole.

CPQ provides the following functionalities in the Save as Favorite feature

Saving your Favorite Configurations

A button called **Save as Favorite** is available on the Cart to save the configuration for a particular Quote/Proposal. Select the products you want to save in the configuration and click **Save as Favorite**, a pop-up is displayed. You can customize the fields displayed this dialog box by using Favorite Configuration settings. The following table explains the different fields inside the pop-up.

Field	Description
Name	Enter a suitable name for your favorite configuration. This name is displayed as a record under your Favorites category.
Scope	Choose an option for the visibility of your configuration record. The available values are, *Public* and *Private*. If you set this to *Private*, this record is visible to only the owner of the record. If you set this to *Public*, this record is accessible to all the users. The default value is *Private*. The visibility of this field depends on the configuration defined by your administrator.
Description	Enter details about your favorite configuration.
Favorite Filters	Select values from the picklists to categorize the favorite configuration. You can select the values as you desire or leave the default values of the picklists that are displayed. Your admin must pre-define the picklists that are displayed on the dialogue box in the **FavoriteFilters** setting.

These favorite configurations are saved on Price Lists and can be used across multiple quotes.

The favorite configuration records are visible on the Catalog page only after you save your favorite configuration record and reconfigure the Cart.

Reusing your Favorite Configurations

As a Sales Rep, you can add a saved configuration to the quote. You can also add multiple configurations. When you add a saved configuration to the quote, a copy of the original design is created. This is useful in case if you want to edit the design on the quote without impacting the original design.

After you have saved your favorite configurations for a particular Price List and have associated this Price List with your Quote/Proposal, a separate category, named **Favorites**, containing your favorite configurations is displayed on the Catalog page. CPQ creates a clone of the favorite configuration records every time you associate the Price List with a Quote/Proposal. Ensure that the primary charge type is same on both the price lists.

> You can set your favorite category to be displayed first when you have multiple offerings, using a custom setting **Show Favorite as First Category**. For more information, see Sequencing the Favorite Category on the Catalog Page.

Note that the rules (Constraint rules, Validation rules, and more) are carried forward in the cloned favorite configuration records.

You can search the favorite configuration records just like the normal products on the Catalog page using **All Favorites** option from the Search drop-down list. You can also use **Refine Your Search** to search for the desired saved configuration. While importing a favorite, you can also control whether you want to include discounts to the cart or not. The price of the favorite product is derived from the Price List associated with the Quote. You can only import a Favorite Configuration when your administrator has defined appropriate access for your profile.

For every favorite configuration record, Add to Cart action button along with the drop-down list containing **Exclude Adjustments**, **Delete** and **Publish** buttons is displayed on the Catalog page.

- **Add to Cart** – this will simply put all the cloned Line Items (including Options, Attributes, and manual adjustments) of your favorite configuration record in the Cart.
- **Exclude Adjustments** – this will fetch details of all the Line Items (including Options and Attributes, excluding manual adjustments) and add them to the Cart. The manual adjustments that you have applied to the original favorite configuration record are not carried forward to the cloned favorite configuration record.
- **Delete** – this will delete a favorite configuration record. This button is dependent on the privacy defined for your favorite configurations. As a user, you can view all the

public configuration records and can delete only those records for which you are the record creator or owner.

- **Publish** – this will publish your favorite configuration record to another Price Lists. From the **Choose Price Lists** dialog box, you must select the price lists to which you want to publish your favorite configuration record. You can publish a favorite configuration record using the **Publish** button on your favorite configuration record in Salesforce also. Once published, this record shows up under your **Favorites** category of the Quote/Proposals that use the price lists containing these favorite configuration records.

You can control the fields displayed in the pop-up dialog box when you click the Product Name on the Catalog page. Similar to custom settings defined for normal products, you can create a separate record under **Object Summary Settings** for the favorite configuration record.

Name the new record as *Favorite Configuration* and select *FavoriteConfiguration__c* as the **Object Name**. For more information, see Configuring Object Summary Settings.

> **Multiple Adjustments Using Favorite Configuration**
> You can configure and add a product to the cart and then apply price adjustments to it. Save this product configuration as Favorite on the cart. Hence, next time you can leverage the favorite configuration to add multiple adjustments to products in the cart. This helps you save time and expedite the process of applying price adjustments using the Favorites on the Product Catalog page.

Providing Ability to Add Multiple Favorites with a Single Click

CPQ provides you an ability to add multiple favorite product configurations to the cart with a single click. On the Catalog page, if you click **Add to cart** for any of the Favorites, CPQ disables the **Add to cart** button for all the available favorites and enables them back when you complete pricing operation.

About Quote Lifecycle Collaboration

As a Sales Rep, you can request different users to collaborate on a configuration using the Quote Collaboration feature. The feature helps you to work with different users, called the Collaborators, and get their input on a configuration or pricing, merge the changes with the original configuration and finalize a single quote. Quote Collaboration allows you to assign a part of the configuration to Collaborator that you need help with, which helps you

to limit the exposure of the entire configuration. You can use SalesForce Chatter to communicate with the Collaborator.

The following roles are participating in a Quote Collaboration:

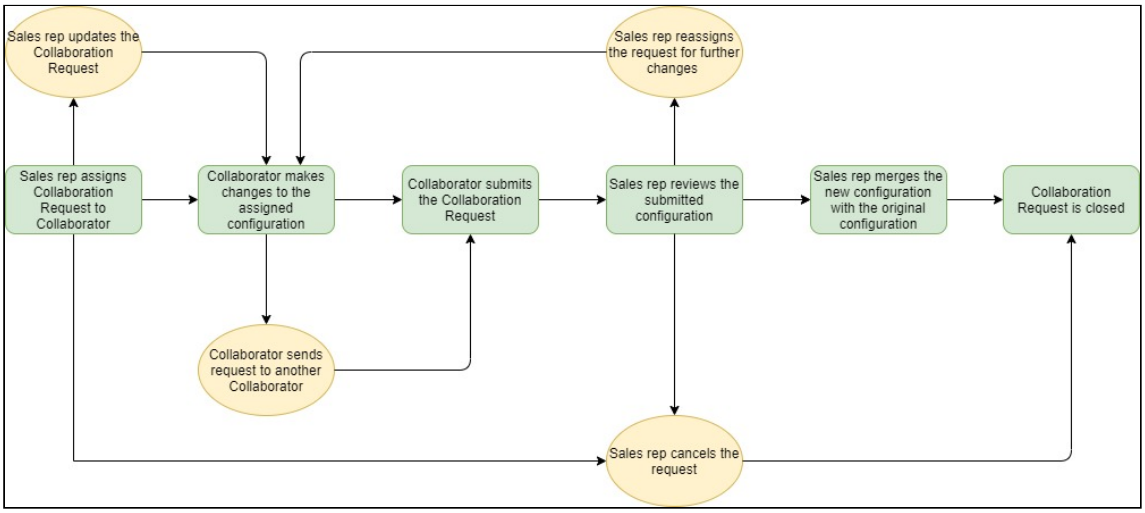
Role	Description
Sales Rep	The Sales Rep initiates a collaboration process. Typically a Sales Rep is a member of the sales team and has a customer-facing role. The Sales Rep is in charge of sending out a single final quote to the customer.
Collaborator	A Collaborator is a product owner, manager, factory manager, or another user who has the knowledge about a particular configuration. Another Sales Rep can also be a Collaborator.

The following table describes the types of collaboration you can use:

Type	Description
Peer-To-Peer	In the Peer-To-Peer collaboration type, users from the same business units can collaborate on a configuration. In peer-to-peer collaboration type is selected, the entire line item is sent over in the collaboration request.
Multi-tier	In the Multi-tier collaboration type, users from different business units can collaborate on a configuration. When you use the multi-tier collaboration type, only specific fields in the line items are sent in the collaboration request. Ideal for situations for which creating a cost structure is necessary when using transfer price mapping or any other collaboration scenario that requires a specific update of certain fields.

Process flow

The basic flow of the quote collaboration from initiating the collaboration request to its closure is described below:



- To initiate quote collaboration, you must assign a collaboration request to a Collaborator on the Cart page. Quote Collaboration feature allows you to assign multiple collaboration requests from a single configuration to a collaborator or assign different collaboration requests to different Collaborators. You can assign a complete line item or specific fields of a line item.
- Once you have assigned the request, the Collaborator can work on the configuration and submit the configuration back to you for review.
- You can monitor and administer the progress of collaboration requests on the Quote Collaboration pop-up once you have assigned them.
- You can review the changes the made after the collaborator submits the request.
- If the changes are not acceptable, you can send the request back to the Collaborator or reassign the request to someone else. You can also reject the request completely if the submitted configuration is not acceptable.
- If you accept the changes you must merge the configuration submitted by the collaborator with the original configuration in the cart.
- After the collaboration request is merged, the request is considered closed. However, you can initiate a new request on the same line items again.

The Collaborator can also share the collaboration request with other Collaborators, which creates a sequential chain of requests. For example, you assign a collaboration request to Collaborator A, who in turn further assigns the same collaboration request to Collaborator B. The sequence here is Sales rep > Collaborator A > Collaborator B. To close sequential collaboration requests they must be merged in the reverse sequence that they were assigned. That is, Collaborator B must submit the request first, then Collaborator A and in the end, the Sales rep must accept the changes.

You must close a collaboration request to finalize the Cart or send the cart for approval. The **Finalize** and **Submit for Approval** buttons are disabled on the Cart page when the collaboration requests are not accepted. This prevents you from finalizing or approving the

cart prematurely. A warning message is displayed on the Cart page when collaboration requests are not closed.

When you initiate a collaboration request, the Collaboration Status changes as the request progresses. The following table describes the different status of the collaboration request:

Collaboration Status	Description
Assigned	The Sales rep has assigned the configuration request to the collaborator. This status is shown with an orange exclamation mark when the sales rep assigns the request to the collaborator.
Submitted	The Sales rep has submitted the configuration request to the collaborator.
Completed	Collaborator has submitted the configuration back to the sales rep.
Accepted	The Sales rep has accepted the configuration and has merged with the parent cart.
Cancelled	The Sales rep has cancelled the collaboration request.
Abandoned	The Sales rep has abandoned the cart.
Deleted	The Sales rep has deleted the line item from the cart.

The collaboration status depends upon the access permissions defined by the administrator. The actions available on the Collaboration pop-up also differ based on permissions. For more information on how the access permissions defined by the administrator impact the collaboration status and the actions on the popup, see Enabling Quote Collaboration in the Org.

The following topics describe the various aspect of Quote Collaboration flow:

- Assigning the Configuration to the Collaborator or Queue
- Working on the Configuration Request
- Merging the Configurations in the Parent Cart

Assigning the Configuration to the Collaborator or Queue

As a Sales Rep, you can assign a bundle as well as a standalone product to the collaborator. You can assign a collaboration request to a User or a Queue from the Cart page. The Users and members of the Queue receive an email notification about the request. After assigning the collaboration request, a record is created in the Product Configuration section on the Quote Detail page with the status as *In Collaboration*.

> The administrator must define proper **Read**, **Create**, **Edit**, and **Delete** permission to give the user access to the Quote Collaboration feature. For more information on permissions, see Enabling Quote Collaboration in the Org.

To assign the collaboration request

1. On the Cart page, you can send one or multiple line items simultaneously:
   a. To assign a single line item: Click the Collaboration icon () for the corresponding line item.

      > CPQ disables the collaboration icon () next to an auto-included primary line item.

   b. To assign multiple line items: Select the line items you want to send and click the menu icon ( ) and click the **Product Collaboration** button.

      > CPQ does not support miscellaneous and primary line items auto-included through constraint rules for Quote Collaboration. CPQ disables the **Product Collaboration** button if any of the primary line items you selected is auto-included.

2. When you click either the collaboration icon or the **Product Collaboration** button, Quote Collaboration pop-up is displayed. Under the **Assignment Details** tab, specify the following information.

Field Name	Description
Owner ID	Select *User* or *Queue* to whom you want to assign the configuration. In the **Select User** dropdown, search and select the user or queue. ⓘ Only the users with **Edit**, **Delete** or **Modify All** access are displayed in the drop-down list.
Collaboration Type	• *Peer-To-Peer*: Select this to get users from the same business units to collaborate on a configuration. In peer-to-peer collaboration type is selected, the entire line item is sent over in the collaboration request. • *Multi-Tier*: Select this to get users from different business units to collaborate on a configuration. When you use the multi-tier collaboration type, only specific fields in the line items are sent in the collaboration request. Ideal for situations for which creating a cost structure is necessary when using transfer price mapping or any other collaboration scenario that requires a specific update of certain fields.
Specifications	Specify the information that you want to convey to the collaborator in this text box.
Start Date	Specify the start date by which the collaborator should start the configuration.
Completion Due Date	Specify the completion date by which the collaborator should submit his configuration.
Priority	Specify the urgency of this request.
Grant Access to Merge Cart (Read-Only)	Select this check box to allow the collaborator to view the entire cart in a read-only mode. This can be useful when you want the collaborator to see the entire cart to gain a better understanding of the overall offering. This functionality is yet not implemented.
Attachments	Attach the additional documents or images.

> Do not click **Cancel** on the Quote Collaboration pop-up just to close the pop-up. Use the cross icon(X) at the top of the pop-up to close. The **Cancel** button cancels the collaboration request itself and the collaboration record is deleted.

3. Click **Notify Collaborator(s)** to send the notification email to the collaborator.

Take into consideration the following while assigning the collaboration request:

- You cannot assign a line item to multiple collaborators. For example, you assigned a line item to Collaborator A. Now, if you select multiple line items and assign them to Collaborator B, including the one already assigned to Collaborator A, CPQ skips the one assigned to Collaborator A and assigns the remaining line items to Collaborator B.
- You cannot finalize or approve the Cart until you close or cancel all the collaboration requests. If the collaboration is sequential, all the collaboration requests must be closed.

After the collaboration request is assigned, the name of the assignee is displayed below the collaboration icon ( ). Click the assignee's name to open the Quote Collaboration pop-up. The following functionalities are available to you on the pop-up:

- The status of the collaboration request at the top of the pop-up.
- The line items assigned to the collaborator under the tab **Assigned Products**.
- You can update the collaboration request and send that to the collaborators using the **Update** button. For example, you can change the user or queue and reassign the request to someone else, you can change the priority, or change the specification.
- You can communicate with the collaborator using the Salesforce Chatter under the tab **Chatter**.
- You can cancel the request altogether, thereby revoking the Collaborator's access to the line items completely. Canceling the collaboration request deletes all the records on the Quote Details page and the Collaboration Request tab, that are created for the collaboration request. Canceling also resets the product configuration status from *In Collaboration* to *Saved*.

You can add products to an existing collaboration request using a global method. For more information, see Adding Products to a Collaboration Request.

Once you have assigned the collaboration request, the collaborator can start working on the assigned configurations as explained in the Working on the Configuration Request section.

Working on the Configuration Request

As a Collaborator, you will receive a notification when the Sales Rep assigns a collaboration request to you. If the Sales Rep assigns you the collaboration as User, you can directly configure the collaboration request. Whereas, if the collaboration request is assigned to you as a member of the Queue, you must claim ownership of the collaboration request. After you have been assigned the request, you can start working on the configuration. When you finish working on the configuration, you must submit the configuration back to the Sales Rep.

You can access the collaboration request in the Collaboration Request tab in All Tabs. The Collaboration Request Detail page holds all the information about the configuration sent by the Sales Rep such as Collaboration Status, Specifications, Priority. You can open the configuration in the Cart using the **Configure Product** button.

> Unless you own a particular Collaboration Request, you cannot see the Configure Products button on the detail layout. Even if the Collaboration Request is assigned to a Queue and the currently logged-in User is a member of that Queue, he does not see the **Configure Products** button unless he accepts the collaboration request.

Taking ownership of the collaboration request assigned to a Queue

1. Go to **All Tabs** > **Collaboration Requests**.
2. From **View** drop-down, select the appropriate Queue and Click **Go!**.
3. Click the **Collaboration Request Id** to view the details of the configuration request.
4. In the **Collaboration Request Detail** page, use the **Owner** field to change the ownership from Queue name to the Collaborator username.
5. In the **Collaboration Request Detail** page, click **Configure Products** button under **Action** column to work on the configuration request.

To work on the collaboration request

You can work on the collaboration request sent to you by opening the configuration in a Cart. Follow the steps below:

1. Click the link provided in the notification email that you received
   or
   Go to **All Tabs** or click All Tabs icon(+)> **Collaboration Requests**.

2. Click Collaboration Request Id to access the specific collaboration request.
3. Click the **Configure Products** button on the Collaboration Request Detail page to start working on the configuration assigned to you. You are redirected to the cart page to perform the configuration tasks.
4. Update the fields of the line item as you desire.
    a. You can add, delete, and replace products in the cart including product for which the Sales Rep initiated the collaboration request.
    b. You can further request another user or queue to collaborate on the configuration assigned to you and initiate a sequential collaboration.

    > If you initiate a sequential collaboration, you cannot submit the configuration until you close the collaboration request you initiated. Otherwise, the Sales Rep is not able to finalize the cart or send it for approval.
    > For example, a Sales Rep assigns a collaboration request to you, which you further assign to Collaborator B. Collaborator B's completed configuration must be merged with your configuration, and your completed configuration must then be merged with the Sales rep's configuration.

5. Click **Submit For Merge** to submit the completed configuration back to the Sales Rep.

    > When the Sales Rep deletes a line item associated with a work-in-progress collaboration request, the corresponding request gets deleted automatically. Hence, you will not be able cannot access and continue working on the collaboration request even by clicking on the email link.

Once you submit the completed collaboration request, the Sales Rep can review and merge the configurations as explained in the Merging the Configurations in the Parent Cart section.

Merging the Configurations in the Parent Cart

As a Sales Rep, you will receive a notification when the collaborator submits the completed configuration. The collaboration icon displays a yellow exclamation mark after Collaborator submits the configuration. You can review the changes before proceeding to merge them with the original configuration. When you merge the configuration the line item in the original configuration is overwritten by line item in the Collaborator's configuration. If the changes made to the configuration submitted by the Collaboration are not acceptable, you may reject the request for collaboration request by cancelling it.

If the Collaborator has replaced the line item from the configuration, the old line item is deleted from your cart and a new line item is created in its place when you merge the collaboration request. If the Collaborator added a new product, a new line item is created on the Cart. You can delete a line item after the collaboration request is closed, the copy of configuration the Collaborator has from the collaboration request is updated accordingly.

After all the collaboration request is closed, you can finalize the cart.

To merge the collaboration request

1. Click the collaboration icon () for the line item the Collaborator worked on to open the Quote Collaboration pop-up.
2. Review the changes under the **Assigned Products** and **Completed Configuration** on the pop-up. Click **More Details** to view the configuration in read-only mode.
3. Perform any one of the following actions:
    a. Click **Merge** to accept the configuration submitted by the collaborator and to merge the line item with the original configuration.
    b. In case if further collaboration is required, you can update the fields in the configuration, and on the Quote Collaboration pop-up click **Update** to assign the collaboration task back to the collaborator. This is useful in a scenario when you are negotiating a transfer price or you need to further request additional changes in the configuration submitted by the collaborator.
    c. You can change the user and assign the collaboration request to someone else.
    d. If the changes are not acceptable, click **Cancel** to cancel the collaboration request.

As a Sales Rep, you have the ability to make changes to the configuration request details and you can re-submit the same configuration to the same collaborator or to someone else, without losing previously merged configuration. When you change the configuration request owner and re-assigns the same configuration to someone else, the previous collaborator receives the cancellation notification email.

> When you update a **Completed** or **Merged** configuration, the status of such collaboration request reverts to **Submitted**, and the Collaborator receives an email notification.

While using Cost and Profitability functionality with Quote Collaboration, if the collaborator and the Sales Rep are from different business units and the collaborator sends a price which becomes one of the cost elements of the Sales Users cart and this price is negotiated, you must use multi-tier as the collaboration type. In Peer-to-peer

collaboration, the only difference is that the merge copies over entire line items rather than merging field values based on transfer price map and merge field set.

Consider that a parent cart has a product and that product is already collaborated by the Collaborator. Originally collaborated line item should get merged as per the transfer price map and merge fieldset. The child cart now has one line item containing the product. If the Quote Collaborator added the same line item and performs a merge operation, the line item would be overridden during the merge operation. After the merge operation, if two line items have the same product, and the collaborator performs a merge operation, the line items are saved as individual records.

Working with the Mini-Cart

The Mini-Cart provides you a brief overview of all products that you configured from the product catalog. When you click the cart icon () on the toolbar, the following functionalities are available for you on the Mini-Cart.

- The wrench icon ( ) next to the product name navigates to go the Configuration page of the bundle product or a product with attributes.
- To remove a selected product, click the delete icon ( ) on the Mini-Cart.
- The **Net Price** is calculated based on the product configuration. The **Grand Total** is the sum of the net price of individual products.
- The **View Cart** button navigates to the Cart page.
- Click **Finalize** to finalize the configuration and go to the quote/proposal page. If there are any validation errors due to the configuration of products, constraint rules, or pricing rules, the Confirm Finalization window is displayed prompting you to confirm whether you want to finalize the cart, with warning messages.
    - Click **OK**. The cart is finalized.
    - Click **Cancel** to resolve the errors.

Working with the Cart

This chapter describes how to work with the Cart.

- Viewing the Cart in Grid View
- Managing Cart Views
- Applying Advanced Filters on the Cart
- Managing Mass Update of Product Fields
- About Cart Locking for Concurrent Access
- About Smart Cart

Viewing the Cart in Grid View

The shopping cart has been redesigned to provide better usability and accessibility to the sales representative.

Following are the key changes that you can see in the shopping cart:

- **Displaying cart level actions on top-right corner:** A system displays the primary cart level actions on the top right corner of the page with **Finalize** action as a default. The other user actions such as **Reprice**, **Apply Promotion**, **Analyze Quote**, **Submit for Approvals**, and so on are made available based on the user scenarios. The user actions that are grayed out show an info icon next to them, providing a reason for non-availability for selection. You can customize the primary and secondary buttons. For more information, see Configuring Display Actions Settings.
- Action buttons, such as **Add More Products**, **Installed Products, Reprice**, **Finalize**, **Submit for Approval,** and more are available as picklist values.
- Mass actions, such as **Delete Product**, **Save as Favorite**, **Mass Update**, **Product Collaboration** at the line item level are available as picklist values. The Copy action is available at the cart header level only, which you can use to copy a single line item or perform mass update. Any change using Mass Update performs the calculation on the client-side for faster execution.
- Easy and type-ahead search functionality on the top of line item table.
- A Quick view of the **Grand Total** amount on the top-right corner. Also, **Show Totals**, **Total Adjustments** are displayed on the page header. The right arrow next to Grand Total shows you the detailed calculations at the line item level.
- If there are two custom fields on Summary Group and one custom field (field A) is dependent on another custom field (field B, where you provide your input) for its value, even if you pass a value in field B and reprice the cart, CPQ does not recalculate field A.
- The **Edit** and **View** buttons in the menu under the proposal name enable you to edit and view the Quote details respectively.
- Bundle cart line items with expand/collapse functionality. The administrator can disable the functionality at the product level.
- A spinner on the cart header to indicate the progress of any rules/adjustments that are being executed in the background. If CPQ encounters errors in executing the rules, warnings and error messages are displayed at the top of the Cart page. In the case of multiple constraint rules errors, a number list of messages is displayed.
- Infinite scrolling for line items avoiding the need for pagination.

- Column headers are always available on the top while scrolling. Non-numeric fields are center-aligned and numeric fields are right-aligned. You can adjust the width of the columns by dragging the right-hand side border. However, if you refresh the page or relaunch the cart the column widths are set to default.
- Editable fields and actions on hover.
- Support for different data types for columns, such as Text, Picklist, Lookup, Date, Hyperlink Formula, Hyperlink Image Formula, Image, Formula, and more.

  > By default, CPQ pre-populates values in lookup type fields on the Cart page. If the administrator has not enabled the **Enable Quick View** setting, you must search for values by clicking the search icon next to the lookup field on the Cart page. For information refer to Configuring Custom Settings.

- **Support for Deal Guidance:** For a product that you configured on the cart page, the system displays a Pricing Guidance pop-up for a customer deal. On the Guidance column when you hover your mouse on a green circle, it pops up a screen with some guidance parameters.

  > On the Cart page, the Pricing Guidance pop-up does not display values for the **Price From** and **Price To** columns if the deal guidance rule has a custom measure.

- **Ability to apply mass adjustments for options without applying adjustments on bundle:** You can apply the mass update on the option and bundle products separately. A new check box design has been implemented that depicts how you can choose to apply the mass adjustments on bundle, option or both.
  On the bundle level, when you click for the consequent times and click **Mass Update**, you see the following behavior:

Click number	Check box design	Behavior
First	Solid square box 	The mass updates are applied only to the bundle product. Sub-bundle products also behave as bundle products.
Second	Minus sign 	The mass updates are applied only to the option products.

Click number	Check box design	Behavior
Third	Checked sign 	The mass updates are applied to both, the bundle and option products.
Fourth	Empty check boxes	The mass updates are not applied.

- Support for long Product Name, Column text, and charge names
- **Providing client-side computation facility**:
    - On Cart Grid with custom setting **Enable Auto Reprice** set to ON, when you change any Quantity on cart line item, system updates the pricing fields such as Extended Price, Net Price for the specific row.
    - Calculate totals (by charges and frequency) and grand totals
    - Calculate subtotals for category
- **Providing bundle rollup for cost and price**: Rollup line gets updated when the individual line's price gets changed and any discount is applied on the summary line when any individual lines get updated.
- **Support for removal of price list items for the existing quotes:** Whenever you have a PLI removed from an active quote, the system notifies you and removes the item automatically.
- **Eliminating the additional pricing spinner:** When the pricing has been completed on the catalog or config page, the system does not show spinner on the cart grid page.
- **Recalculating the end date of one-time fee:** Based on the start date and the selling term, the end date for a one-time price type is re-computed when you delete it in the cart.
- **Calculating the selling term of recurring fee:** When Price Type = Recurring, CPQ calculates the selling term on the cart as follows:
    - If the cart line item has start date and end date (but the Default Selling Term is not defined on the PLI), CPQ calculates the selling term based on the start date and end date.
    - If the Default Selling Term is defined on the PLI (but the cart line item does not have start date and end date), CPQ calculates the selling term based on the default selling term.
    - If the default selling term is defined on the PLI and the cart line item also has start date and end date, CPQ gives precedence to the start and end dates, and calculates the selling term accordingly.

- Showing sub totals and totals sections: The Cart Grid UI shows subtotals and totals section at the bottom of the cart page based on the setting called **Enable Contextual Totals**. For more information, see Config System Properties.
- **Providing an indicator for line item approval:** If you make updates to a quote/ proposal, the system displays an indicator "Approval Required" for the line item on the cart page.
- **Reordering of primary lines on the cart:** You can drag and drop the primary line items on the cart using the hand icon that helps you sort the products.
- **Cancelling the order line items**: While amending an in-flight order, you can cancel one or more order line items from the cart that are not yet fulfilled (orders with pending and in-fulfillment status).
- **Improved User Experience for Adjustment Amount field**:

  CPQ displays a blue bar when you click or hover over the **Adjustment Amount** field on the line item or **Totals** section in the cart page. This feature improves the user experience by highlighting the **Adjustment Amount** field in order to make it noticeable during an edit mode.
- **Three Columns in the left section of the Cart Grid**
  - The left side of cart grid is limited to three line items fields.
  - When you create a new cart view and try to drag the fourth field into the third section, it moves that field to the middle section.
  - When you create a new cart view as an admin, you can see the same behavior as a general user creating a cart view.
- **Hiding Secondary Action Buttons upon Hard Revalidate**: Upon hard revalidation of the quote by changing the product version, CPQ hides the secondary action buttons on the cart that are part of Display Action Settings. For example, Add More Products, Installed Products, Add Miscellaneous Items, Reprice, Finalize, and so on.

- On the configuration page at the **Summary** section, the **Hide Option Groups** link is disabled when no product option is selected.
  For example, while you are configuring a quote, you do not want to include option groups that are associated with the product. In that case, you want to finalize the quote and add the product on the cart without any options. Hence, the Product Summary section needs to hide or disable the Hide Option Groups link, which is irrelevant in this scenario.
  Suppose you are providing a quote of an internet plan to a customer which has no installation fee and router charges for the festival season. In that case, you can configure this product without the options, namely Installation Fee and Router Charges.

You can resize the first column and change the sequence of the columns (by dragging the line items vertically). For more information, see Configuring Display Columns Settings.

Please keep in mind the following pointers for icons/action buttons when using the new grid view:

- Copy, Remove and Product Collaboration – these icons/action buttons are available when all the selected line items are primary line items. Else the icons are grayed out.
- Mass Update – this is available for both, primary and secondary line items, except you have separate configurations such as **Quantity** = *Read Only*.

Required Configuration

This functionality can be achieved by a simple configuration of creating a new flow with the Cart page name as *Apttus_Config2__Cart#cartgrid*. You must then specify the flow name in your **Configure Products** button to view the new UI for the shopping cart. For more information, see Configuring Flow Settings and Creating Custom Buttons for Different Flows.

> For more information on the service cart, see Working with the Service Cart.
> For more information on the cart views, see Managing Cart View

Managing Cart Views

Similar to Salesforce views, the cart grid supports Cart Views containing predefined filter values on the cart. These views help you filter and display cart line items as per your configurations. You can set the custom views based on many parameters such as product, color, and location, using the **Create New View** pop up.

To create Cart Views

1. On the Cart page, click view drop-down. **Create New View** pop-up is displayed.
2. Select a value in the **Group By** field. The default value of the **Group By** fields is *Product*.
3. Enter a name in the **View Name** field.
4. Select **Set as Default** checkbox to make the view default. If you select this checkbox then the administrator view is overridden.
5. Go to **Choose Cart Columns** tab.
6. Drag and drop the columns that you want to display for the view of the cart from the left-hand side list to the right-hand side list.

7. Click **Save**.

By default, you can create a new view, edit and delete views. When you delete a view, the record is deleted from your flow but does not get removed from the list of views. This means that your view is still visible to other flows.

You can create Cart Views with the Advanced Filters as well. To create a view with the Advanced Filters you must apply the filters to the line items in the Cart and select the **Save with Filters** checkbox when you create the cart view.

Required Configuration

For each view that you create, you must ensure that your administrator has listed your flow name as a picklist value in **Config Settings object > Flow picklist field**. If your administrator has not listed the flow name in this picklist field, you would be allowed to create a new view but would not be able to save the view. Also, after your administrator has listed the flow name in this picklist field, you can view and use all the views created by different users, provided the flow names are listed in the Flow picklist.

The columns displayed in your view are mapped with your flow. You can edit the columns using Display Column Settings. You can modify the group-by fields by listing the API names of the desired fields in the **Groupby** field available at **Setup > App Setup > Develop > Custom Settings > Config System Properties > Manage > System Properties**.

Applying Advanced Filters on the Cart

Advanced Filters are real-time filters that you can define on the cart. After you add products to the cart you can filter them to find particular products on the Cart page. If you have applied Cart Views on the Cart page, you can use Advanced Filters functionality to narrow the results further. You can save the filters you defined by selecting the **Save with Filters** checkbox when creating Cart Views.

To apply advanced filters

1. Go to Cart page.
2. Click the Advanced Filter icon() . A dialog box is displayed below the icon.
3. Enter the following details and click **Apply.**

Options	Description
Field	Select the column based on which you want to filter the line items. ⓘ You can only apply filters to the column defined in the default cart view.
Operator	Select a comparison operator.
Field Value	Enter the value of the column you selected in **Field**. ⓘ If you select Field that is of type lookup with Lookup Field Settings, CPQ only displays the list of values that are saved in at least one line item configuration in the Cart.
Consider Only Parent Lines	Select the checkbox to filter the line item based on standalone and bundle line items. The options line items are not considered in the filter.
Select All Qualified Lines	Select the checkbox to select the line items that fulfill the filter criteria. CPQ only selects line items on the current page in a paginated cart. You must navigate through other pages to allow CPQ to select qualified line items on those pages.

After you apply the filters, the line items are displayed based on the criteria. A counter is displayed next to the Advanced Filter icon that denotes the number of primary line items that fulfill the filter criteria. On the Advanced Filter dialog box, click **Clear All**, to remove the filter.

Advanced Filters on the cart feature has the following limitations.

- Filters are not applied on Line Item fields of the following types
  - Multi-select Picklist
  - Actions
- 'OR' operator is not supported. ‘AND’ operator is implicitly applied when you add more than one filter condition.

> The filters applied on the cart are saved only in the main memory, which means that after applying filters if you navigate to another page, the filters are lost when you come back to the cart. However, if filters are saved in some custom view which is set as the default view then the cart will be loaded with this default view each time with the filters applied on the cart.

Managing Mass Update of Product Fields

Mass update allows you to bulk update fields, such as quantity, adjustment type, and adjustment amount for multiple products at a time. You can mass update standalone, bundle, and option line items on the Cart page. You can manually select line items to mass update or select them using Advanced Filter on the Cart page. Refer to Applying Advanced Filters on the Cart. However, CPQ does not apply mass updates to rollup line items even if you select them.

To mass update product fields

1. From the cart page, select the desired product(s) and click **Mass Update.**

   > A counter is displayed on the Mass Update popup which indicates the total number of line items you selected for mass update.

2. In the Mass Update popup, specify the desired value for the fields and click **Apply**.

By default, you see the following fields in the Mass Update popup.

- Quantity
- Adjustment Type
- Adjustment Amount

You can either specify the value for a field or select the **Reset** checkbox. Mass Update will override any existing user-entered value. If you keep a specific field blank and click **Apply**, no action is taken on that field. When you select the **Reset** checkbox, the field is updated with the CPQ default value. For example, when you reset the **Quantity** field the value is updated to *1*. However, if you reset **Start Date** or **End Date** fields, the values are updated to the values from the Quote Details page.

> - To customize the fields displayed in the Mass Update popup, admin can navigate to **Config Settings > Display Column Settings**. Select *Mass Update* from the **Display Type**, and relevant flow from **Flow** dropdown.

- You need to ensure that **Allow Manual Adjustment** for the corresponding PLI is set to *True*.
- You should not use Mass Update along with Multiple Adjustments.

About Cart Locking for Concurrent Access

When you edit a cart, use cart locking to lock out all the Sales reps with concurrent access.. The other Sales reps can only view the cart in read-only mode. The cart is unlocked when you save, finalize, abandon, or close the cart. Sales reps in read-only mode can gain edit access only after these operations are performed. Cart locking helps prevent data loss while you are working on a configuration.

If you leave the cart idle for a predefined time limit, the cart is unlocked automatically. The cart is considered idle if you do not perform changes to the line items. Changes include updating a value, deleting line items, or adding line items. Other actions on the cart such as updating cart views, viewing the totals, or applying filters are not considered modifications to the line items, and time spent is counted as idle time. The time limit is defined in **Cart Edit Access Idle Timeout in Minutes** in Config System Properties. After the time limit is exceeded, the cart is unlocked allowing any Sales rep to open the cart in edit mode, locking the cart while that Sales rep is edits it.

Ensure that your administrator has configured the necessary custom setting before using Cart Locking. For more information, see Enabling Cart Locking for Concurrent Access.

While you are in edit mode, if another user requests control by clicking **Request Edit Access**, you are notified through an email with the message: "`<<user in read-only mode>> has requested to edit the cart. Would you like to grant access?`". The field **Current User** in the Product Configuration object contains the ID of the Sales rep currently accessing the cart in the edit mode. In edit mode, after you save, close, finalize, or abandon the current cart, an email notification is sent to the Sales reps who are waiting to get the edit access to the cart.

Email notifications are sent in the following format:

Action on the cart	Email notification content
Claim Edit Access	User <<name in read-only mode>> has requested edit access to <<proposal name>>. To grant this user with access, please either: save, close, finalize, or abandon your configuration.
Save or Close	<<User name - edit mode user>> is done configuring. You can claim edit access to the cart by clicking the **Claim Edit Access** button on the cart.
Finalize	<<User name - edit mode user>> has finalized the configuration. You can now view and edit this finalized version by clicking the **Claim Edit Access** button on the cart.
Abandon	<<User name - edit mode user>> is done configuring. You can claim edit access to the cart now by navigating to the quote header page and clicking **Configure Products**.

About Smart Cart

The Smart Cart feature allows you to manage the cart with a large number of line items. In the Smart Cart flow, line items are priced in groups that are divided based on the threshold and split criteria. The threshold defines the number of line items in a group. The split criteria list the fields based on which the line items are grouped. The threshold and split criteria are defined by the administrator in **Split Cart Threshold** and **Split Cart Criteria Fields** settings respectively, in Config System Properties.

Like the regular cart, in the Smart Cart flow, you can configure and add products to the cart. Once you add the products to the cart, you can use the pricing functionalities like adjustments, tiers promotions to adjust the pricing of the line items. Submit the cart for pricing after you finish applying all the adjustments. Click **Submit for Pricing (Async)**. Pricing of the line items in the Smart Cart flow is processed differently than the regular cart. For more information, see About Smart Cart Pricing.

Once the pricing is calculated, finalize the cart. After the finalization, a hierarchy of configurations is generated within your quote. The quote consists of the Main Configuration, a Config Configuration, and Pricing configurations. The Main Configuration consists of the Config Configuration and the Config Configuration lists all the Pricing Configuration. The Pricing Configurations represent the groups that are created based on the threshold and the split criteria. Each Pricing Configuration consists of the proposal line items the belong to that particular group.

You can reconfigure the finalized cart. In the Smart Cart flow, when you reconfigure a finalized cart, CPQ uses the saved product configuration to build the cart again.
Any changes you make to the cart are stored in the same configuration. The saved configuration is not superseded with a new copy of the configuration. You can sync the cart lines with the opportunity using the **Synchronize with Opportunity** button after generating the quote.

To enable the Smart Cart flow

To use the Smart Cart feature, on the Quote details page, set the field **QTC Profile** to *Split.* When you make this change, CPQ switches to Smart Cart flow.

Smart Cart feature does not support the following functionalities:

- Analyze Quote
- Asset-based operations such as Renew, Change, Swap, Terminate
- Constraint Rules are only supported in client-side execution mode
- Cost Model
- Creation of Assets
- Deal Guidance
- Favorite Catalog
- Line level Approval
- Order Generation
- Order Management Amend, Cancel operations
- Related Pricing
- Sales & Summary group Promotion
- Service CPQ
- Total level discounting
- Default Document Generation

Tasks Available in the Smart Cart Flow

You can perform the following tasks in a Smart Cart flow.

- Filter and search products on the Catalog page.
- Configure the products from the Catalog page.
- Add products to that cart. You can also, add multiple products to the cart simultaneously.
- View the mini cart.

- Search for products by name on the Cart Page. You can apply advanced filters to narrow the search further.
- Add more products. When you click **Add More Product** button and you are redirected to the Catalog page.
- Configure the line items on the Cart page.
- Copy and delete a line item.
- Apply a mass-update on the line items.
- Edit all the editable columns.
- Apply adjustment or multiple adjustments.
- Send the quote for collaboration.
- Execute pre-pricing logic. Click the **Pre-Price** button on the menu.
- Submit the cart for pricing by clicking the **Submit for Pricing(Async).**
- Execute post-pricing logic. Click the **Post-Price** button on the menu.
- Reprice the cart.
- Save the cart.
- Abandon the cart. By default, the configuration is deleted from the Quote details page. To retain a copy of the configuration your administrator must enable the **Keep Abandoned Carts** setting. For more information, see **Config System Properties** section in Configuring Custom Settings.
- Finalize the cart.
- Reconfigure the cart.
- Revalidate the cart.
- Sync proposal line item with associated opportunity. Click the **Sync with Opportunity** button on the Quote Details page. The administrator can also enable auto-sync as well.
- Generate and send the proposal. You can also use Doc Gen functionality to generate and send proposals.
- Accept the proposal.
- Clone a quote in Smart Cart. Click the **Clone(with Line Items)** button on the Quote Detail page. The new cloned quote retains all the information of the parent quote, including:
    - Proposal header details
    - Product Configurations

    > When you clone a finalized quote, Proposal Line Items are not cloned.
- Create agreements with line items in Smart Cart. Click **Create Agreement With Line Items** on the Quote Detail page. The agreement is created along with agreement line items. The time taken to create such an agreement depends on the number of line items in the quote. The agreement is automatically finalized and **Is Task Pending** flag is set to *false* after creation.

About Smart Cart Pricing

In the Smart Cart flow, CPQ allows you to run pre-pricing logic before you submit the configuration for pricing, provided that your administrator has defined pre-pricing logic. Pre-pricing feature executes any custom logic defined by the administrator to be executed before any pricing actions. Pre-pricing is used to define any fields that are needed to be predefined before pricing is calculated. Click the **Pre-Price** button on the menu.

After the pre-pricing logic is executed, click **Submit for Pricing(Async)** button on the Cart page to calculate the price. After you click **Submit for Pricing(Async)**, **Net Price** is calculated and the total is displayed next to the corresponding line item. If you change any fields that are used in pre-pricing logic or you add new line items to the cart that requires pre-pricing, you must execute pre-pricing again.

After the pricing is complete, you must run post-pricing logic, if your Administrator has defined a custom post-pricing logic. Post-pricing uses the pricing fields like **Net Price** and **Net Extended Price** which are calculated after you submit the configuration for pricing. Click the **Post-Price** button on the menu. If any of the pricing fields are changed after post-pricing, you must execute post-pricing again.

After you add products to the cart and if the pre-pricing and post-pricing logic are defined, all the pricing and finalization actions like **Submit for Pricing (Async)**, **Post-Price** and **Finalize** are disabled on the menu. The buttons are enabled one by one as you perform the corresponding actions. For example, before you can submit the configuration for pricing you must execute pre-pricing logic, then all the buttons except **Pre-Price** are disabled. Once you pre-price the cart, **Submit for Pricing (Async)** is enabled but the **Post-Price** and **Finalize** buttons are still disabled. After you click **Submit for Pricing (Async)** button and pricing are complete, **Post-Price** is enabled. And only after post-pricing, **Finalize** is enabled. In the scenario when you update a field after pricing that requires pre-pricing, all the buttons except **Pre-Price** are disabled. The only buttons that are always enabled are **Save**, **Abandon** and **Close**.

Pre-pricing and Post-pricing logic are automatically executed if your administrator has configured the settings in *Config System Properties*.

After the pricing is complete, you can finalize the cart and see that a hierarchy of configuration is generated on the Quote Detail page.

Pricing Products

In the Pricing stage, you can price the selected products and apply adjustments such as markup value, discounts, and more from the Cart page.

You can choose to view the cart in the normal UI or grid UI. The functionality and buttons remain the same, only the orientation and spacing of columns and buttons are different in both the UI.

You can perform the following actions:

Action	Description
Add More Products	Return to the Catalog page to add more products.
Installed Products	Select products from a list of products that have been purchased and associated with the account associated with the proposal. This enables you to select products directly that are previously purchased by the account.
Add Miscellaneous Item	Add any miscellaneous charges that are not SKU related such as shipping charges or sales tax to the quote.
Reprice	To recalculate the manual pricing adjustments.
Review & Finalize	Proceed to the final stage of configuration.
Submit for Pricing (Async)	Submits the cart for async pricing. When you click the button, you can perform one of the following actions: • **Do Not Submit**: Click this to cancel the cart pricing. • **Submit & Go To Proposal**: Click this to submit the cart for pricing and return to the Quote Details page. You will receive an email notification after the pricing is completed. • **Submit & Stay On Cart**: Click this to stay on the cart while CPQ completes the pricing is processed. The Cart becomes read-only while the pricing is processed.
Save	Saves and changes the status to Saved, if status did not equal to Saved, Pending Approval or Ready for Finalization. You may return to the specified return URL or proposal object. For example, if you came to arrive on the cart from the Opportunity, the system will return to the Opportunities page.

Action	Description
Quick Save	Saves and changes status to Saved if status did not equal to Saved, Pending Approval or Ready for Finalization. You stay on the same page
Save & Reload	Saves and changes status to Saved if status did not equal to Saved, Pending Approval or Ready for Finalization. Reloads the Cart by re-querying the Product Configuration and setting the cart to show the first page, by doing this it runs Pricing.
Edit Price Agreement	Enables you to add and set prices for a product that are associated with your contract. When you renew or regenerate a catalog from your contract the defined price agreement prices are applicable.

The following are the key fields in the Summary tab and their descriptions:

Fields	Description
Actions	You can see the one or more of the following icons: • Click **Configure** to configure a bundle option within a bundle. This offers increased control over product selection, as well as making it easier to select products as selecting one bundle full of products can replace having to select those products separately. • Click **Copy** to clone a bundle with attributes, options, and pricing and make modifications as needed and save it as another bundle. • Click **Remove** to remove a product from your current selection.
Totaling Group	The group used for a line item in the Totals view.
Product	The selected product.
Charge Type	The relevant charge type for a product. A product can have multiple charge types such as License Fees, Implementation Fees, and more.
Base Price	The base, or starting price of a product. This is the List Price with any line item automatic adjustments applied.

Fields	Description
Option Price	The sum of the Adjusted Price of the options that fulfil the following conditions: • Is optional = FALSE • Rollup Method = PER UNIT or BLANK • Price included in Bundle = FALSE • The charge type matches with the bundle line
Quantity	The quantity of product being quoted.
UOM	Unit of measure for the quantity field. Set up by the System Administrator per product/charge type.
Term	The selling term of the product, measured in Months, Days, Years etc.
Frequency	Whether the charge is One Time, Recurring, or Usage.
Extended Price	Base Extended Price + Flat Option Price + (Option Price * Quantity) ⓘ The Option Price is multiplied by the bundle quantity because *Rollup Method = PER UNIT* implies that the option quantity is per unit of the bundle.
Adjustment Type	Values for manual adjustments whether you want to apply discount or markup on the price.
Location	The location in which the said product will be available and delivered.
Adjustment Amount	Field to enter the amount or percentage of manual adjustment.
Adjusted Price	Extended Price + Line Level Manual Adjustments
Renewal Adjustment Type	Specifies the type of renewal you want to apply for an asset during renewal. Values are: *%Uplift* and *Uplift Amount*. Applying any value here shows the effect on asset renewal only. Ensure that you have an active Workflow executing asset pricing, else the price change will not appear in the Cart.

Fields	Description
Renewal Adjustment Amount	Specifies the actual amount of the value you have chosen in **Renewal Adjustment Type**. This amount is applicable when you renew an asset. Ensure that you have an active Workflow executing asset pricing, else the price change will not appear in the Cart.
Guidance	Specifies the percentage range within which you can adjust the amount. This amount can be the amount range by which a product can be marked up or discounted.
Net Price	Adjusted price with any category level adjustments applied.
Is Optional	Indicates if your line item is an optional one. This column works only when you have set **Enable Optional Items** to true in System Properties. Refer to System Properties topic for details.
Is Optional Rollup Line	Indicates if the price of the optional line item should roll up at the bundle level. This column works only when you have set **Enable Optional Items** to true in System Properties. Refer to System Properties topic for details.
Allocate Group Adjustment	Indicates if the line item is eligible for group adjustments. Set this to *true* when you want to allow group adjustments to be applied at the line item level also. Set this to *false* to lock a line item record and prevent manual and group adjustments. This functionality is helpful when your cart has many products and you want to prevent the sales representative from applying mass adjustments to certain products that are approved by your Sales Head. For such scenarios, you can lock these products on the cart.
Flat Option Price	The sum of the Adjusted Price of the options that fulfil the following conditions: • Is optional = FALSE • Rollup Method = FLAT • Price included in Bundle = FALSE • The charge type matches with the bundle line

The Show Duration link is for advanced functionality related to Pro-ration and co-termination of recurring charges. If this is setup for a price list item, then you can charge in a different frequency from what the price list item is defined in and set Start and End Dates and a Term. For example, a price list item may be a recurring yearly charge but on this quote, you may wish to charge only for 6 months.

In the **Shopping Cart** section, the price line items are further broken out to reflect pricing for any options associated with the bundle product.

The **Totals** section shows all Totals by product category and a Grand Total sum of one time plus recurring charges. The Totals tab also allows for a discount or markup to applied to a category as a whole and rolled down to the individual line items in that category. The system distributes the adjustments uniformly across each line item. If any automatic summary adjustments have been configured, they will also appear in this view.

Proposal Locations are defined to distinguish differentiate the products as per your customer locations.

On the cart page,

- The location drop-down list displays all the account and proposal locations.
- If your system admin has created any lookup filter criteria, the system follows the criteria and filters the locations accordingly.

To price a product on the shopping cart

You can edit any field for an added product on the cart only if the administrator has set requisite permissions for the Cart Detail View page.

1. In the **Quantity** column, you can change the quantity for a product. The values in the Quantity column for a product can be driven by the numeric expression builder configured for the selected product. You can click **Reprice** link individually to enable the system to recalculate the price considering the new quantity.
2. In the **Selling Term** column, you can enter a number to specify the term duration during which the product is sold.
3. In the **Frequency** column, you can select a frequency type. The frequency type is displayed for recurring charges.
4. From the **Adjustment Type**, you can select one of the following and enter a numeric value in the **Adjustment Amount** column:

Option	Description
% Discount	Select this and enter a discount percentage in the **Adjustment Amount** column. For example: If you enter 10 in the Adjustment Amount column, this implies that you want to give a discount of 10%.
Discount Amount	Select this and enter a flat discount amount in the **Adjustment Amount** column.

Option	Description
% Markup	Select this and enter a markup percentage in the **Adjustment Amount** column. For example: If you enter 10 in the Adjustment Amount column, this implies that you want to markup the price by 10%.
Markup Amount	Select this and enter a flat markup amount in the **Adjustment Amount** column.
Price Override	Select this and enter an amount in the **Adjustment Amount** column. The amount you enter overrides the net price of the product.
Base Price Override	Select this and enter an amount in the **Adjustment Amount** column. The amount you enter overrides the base price of the product and all the other calculations on the product would happen based on the new base price. **Allow Manual Adjustment** must be selected on the product line item, in order to make **Base Price Override** editable.
% Discount Off List	This is similar to % Discount; except it calculates the discount amount always on List Price of a Bundle and applies to Extended Price. ⓘ The option % *Discount off List* is not available by default. The administrator must add the value to the **Adjustment Type** field in the Line Item object. For more information, see Enabling Multiple Adjustments at the Line-Item Level.
% Uplift	This gets applied every time to products that are added from the Asset.
Price Factor	Select this and enter an amount in the **Adjustment Amount** column. The amount you enter gets multiplied by the **Extended Price** to calculate the new **Net Price**.
Base Price Discount	Indicates that the adjustment amount is reduced from the Base Price.
Base Price Markup	Indicates that the adjustment amount is added to the Base Price.

Option	Description
% Discount (Bundle Only)	Select this and enter a discount percentage in the **Adjustment Amount** column. The value you enter applies a discount on the bundle without applying that same adjustment on the options within the bundle.
Discount Amount (Bundle Only)	Select this and enter a flat discount amount in the **Adjustment Amount** column. The amount you enter applies a discount on the bundle without applying that same adjustment on the options within the bundle.
% Uplift (Bundle Only)	Select this and enter an uplift discount percentage in the **Adjustment Amount** column. The value you enter applies an uplift to the bundle every time a product is added from the Asset.
% Markup (Bundle Only)	Select this and enter a markup percentage in the **Adjustment Amount** column. The value you enter applies a markup on the bundle without applying that same adjustment on the options within the bundle.
Markup Amount (Bundle Only)	Select this and enter a flat markup amount in the **Adjustment Amount** column. The amount you enter applies a markup on the bundle without applying that same adjustment on the options within the bundle.
Price Override (Bundle Only)	Select this and enter an amount in the **Adjustment Amount** column. The amount you enter overrides the net price of the bundle without applying that same adjustment on the options within the bundle.
Price Factor (Bundle Only)	Select this and enter an amount in the **Adjustment Amount** column. The amount you enter is multiplied with the Extended Price to calculate the new Net Price of the bundle.
% Discount Effective	Select this and enter a discount effective percentage in the **Adjustment Amount** column. The value you enter applies a discount on the bundle or total so that the Net Adjustment % on the Bundle or Total is set to the user requested value. This adjustment type is applicable when the following requirements are fulfilled: Group Adjustment Spread is set to True. Max Adjustment Line must be blank or 1. Bundle price must be zero.

Option	Description
% Markup Effective	Select this and enter a discount effective percentage in the **Adjustment Amount** column. The value you enter applies a markup on the bundle or total so that the Net Adjustment % on the Bundle or Total is set to the user requested value. This adjustment type is applicable when the following requirements are fulfilled: Group Adjustment Spread is set to True. Max Adjustment Line must be blank or 1. Bundle price must be zero.
Price Factor Effective	Select this and enter an amount in the **Adjustment Amount** column. The amount you enter is multiplied with the Extended Price to calculate the new Net Price of the bundle or total so that the Net Adjustment % on the Bundle or Total is set to the user requested value. This adjustment type is applicable when the following requirements are fulfilled: Group Adjustment Spread is set to True. Max Adjustment Line must be blank or 1. Bundle price must be zero.

5. Click the **Details** tab. In this view, the price line items are further expanded to reflect pricing for any options associated to bundle products. You can make changes as per your requirement.
6. Click the **Totals** tab. This view shows all Totals by product category and a Grand Total sum of one time plus recurring charges. The Totals tab also allows for a discount or markup to applied to a category as a whole and rolled down to the individual line items in that category. The system distributes the adjustments uniformly across each line item. If any automatic summary adjustments have been configured, they will also appear in this view.
   If the admin setting **APTS_RollUpOptionNetAdjustment** is set to True, when you apply adjustments on options, CPQ displays the adjustments on options at the bundle and summary totals level. CPQ calculates the Net Adjustment % of the bundle and total as the difference between the price before and after all manual adjustments including option adjustments. For example:

Product	Base Extended Price	Flat Option Price	Extended Price	Adjustment Type or Amount	Adjusted Price	Group Adjustment %	Net Price	Net Adjustment %
Bundle	1000	90	1090		1090		1090	-0.91%
Option	100		100	10% Discount	90		90	-10.00%

7. Do one of the following:
   - To reprice all the parameters such as a change in the quantity, term, frequency, and any other custom field, click **Reprice All**.
   - To apply all the adjustments you made on the shopping cart, click **Apply All Adjustments**.

     When you update a line item using actions mentioned in the above steps, the pricing pending icon () appears at the end of that line. This icon indicates that you must reprice the cart to get the latest prices after you update a line item.

All your selected products are priced with the desired parameters.

- On the configuration page, you can observe the **See Price in Cart** link for the option product. When you click on that link, you notice that for the free item, Included with bundle text is displayed as a price value.
- For single or multiple price list items of the option products, the **See Price in Cart** link pops out well-aligned table for the prices of the corresponding products.
- You can see the carrot icon when the option has multiple charges of multiple charge types.
- You can see "Included in Price" when the option has a single charge, and it is included in the price of the bundle.
- You can see the pop-up for the charge types when you click **See Price in Cart**.
- On the product options screen during product configuration, you can see prices against the options and when you expand the arrow against the options a list of prices displays.

- In the multi-adjustment pop-up, when you apply **% Discount**, CPQ applies the **Discount Amount** to the **Base Price** after multiplying it with **Quantity** and **Selling Term** of the bundle product.

You can review the products and prices configured for your quote one last time before returning to the Quote/Proposal page.

- Creating Price Ramps for a Product
  You can create a price ramp for a bundle product or for an option inside the bundle. You can create ramp lines to spread across time periods and add new ramp lines, create user ramps where from one period to the next the quantity (number of users) increases (user count/quantity ramp), and also create revenue ramps where from one period to the next the level of revenue commitment increases (total price/override ramp).
- Defining Tiered Pricing
  Tiered pricing is used to establish rates for a future usage. There is no pricing until you know the future usage.
- Editing Price Agreements
  After product selection and configuration, you are redirected to the Shopping Cart page. The **Edit Price Agreement** button enables you to add and set prices for a product that is to be associated with your contract. When you renew your contract or regenerate a contract, the defined price agreement prices are applicable.
- Managing Contract Pricing
  Contract Pricing in CPQ enables you to store contracted prices in the system. When a quote is created in CPQ, the quote price is determined by looking up the price defined in the Contract.

Applying Multiple Adjustments on Line Items in the Cart

You can apply multiple adjustments at the line level either by applying the adjustments in the corresponding columns or clicking $ icon next to the line item from the cart page. This enhances the functionality when you want to apply multiple adjustments to get to the final price for a particular line item.

⚠ Before you apply multiple adjustments to your line item in the pop-up dialog box, ensure that you have not given any manual adjustments against the line item on the cart.

You can apply the adjustments in the following manner:

- Adjustments on the previous price
- Adjustments on the starting price
- Bucket Adjustments (For more information, see Applying Bucket Adjustments on Line Items in the Cart.)

Here is an example of the difference between the previous and starting price types:

Extended Price = $100

SR No	Adjustment Applies to	Adjustment Type	Adjustment Amount	Net Price on Cart	Description
1	Starting Price	% Discount	10	90	Create an adjustment line with adjustment applies to as either starting price or previous price (it doesn't matter because it is the first adjustment). The new net price will be $90
2	Starting Price	% Discount	10	80	Now create another adjustment line with adjustment applies to as starting price that applies a 10% discount. The new total adjustment amount will be 100 * (10% + 10%) = $20. The new net price will be $80.

SR No	Adjustment Applies to	Adjustment Type	Adjustment Amount	Net Price on Cart	Description
1	Starting Price	% Discount	10	90	Create an adjustment line with adjustment applies to as either starting price or previous price (it doesn't matter because it is the first adjustment). The new net price will be $90
2	Previous Price	% Discount	10	81	Create another adjustment line with adjustment applies to as previous price that applies a 10% discount. The new total adjustment amount will be 100*10%*10% = $19. The new net price will be $81. This is because the previous price would be 100 plus the 10% discount.

- When you save the adjustments, CPQ triggers a Reprice action.
- The number of adjustments applicable for a line item is defined in the **Max Adjustment Lines** custom setting. For more information, see Configuring Custom Settings.
- You can restrict adjustment types using Product Attribute Rule (PAR) at the line item level on the Cart Grid UI. This capability is supported for Adjustment Type picklist.

Applying Bucket Adjustments on Line Items in the Cart

A **bucket** (field on **Adjustment Line Item** object) serves as a container for the adjustments that a sales representative applies on a line item in the cart. The final value after applying the adjustments within the same bucket is stored in a field called **Running Total**.

You can group the adjustments in buckets. Adjustments in the same buckets are applied to the same starting price and the results form the running total price points. With bucket adjustments, you support cascading pricing discounts by controlling if you want to apply the discounts on the previous or starting price of the Line Item. The adjustments that fall under the first bucket are applied to the starting price. Subsequent bucket adjustments are applied on the adjusted price in the previous bucket, which is Running Total (Adjusted Price).

> ⚠ You can apply bucket adjustments when the Administrator has enabled the **Enable Adjustment Buckets** checkbox in Config System Properties.

If Bucket field value is null, a sales representative must choose between the starting price, from bucket 2 to the running total and the previous price. The pricing engine performs the following sequential tasks:

1. Applies the manual adjustments from bucket 1 to the starting price.
2. Applies the price from bucket 2 to the running total that is the result of calculating bucket 1 adjustments.
3. Applies bucket 3 to the running total of bucket 2 adjustments.

The same calculation logic applies to subsequent buckets. Numbered buckets are processed before null buckets.

Use Case: Applying Bucket Adjustments

Description: This use case describes how you can apply adjustment buckets on a line item in the cart page. You might use this functionality differently, depending on your business case.

Bucket adjustments help you control at which point you want to apply the adjustments in a cascaded adjustment cycle.

Suppose you are a sales representative and you want to provide cascading price discounts to a product. You have three categories of discounts you can provide. The first category is called “Valued Customer Discounts”, the second category is called “Marketing Incentives”, and the third category is called “Seasonal Promotions”. For each of the category, you want

to provide multiple discounts, being either % based or an amount based. This is also to incentivize the user to make an immediate purchase.

In this case, you can apply three buckets, each representing a category. In each bucket, you can define one or multiple % based discounts, one or multiple amount-based discounts.

Bucket 1: Valued Customer Discounts (VCD)

1. 5% off

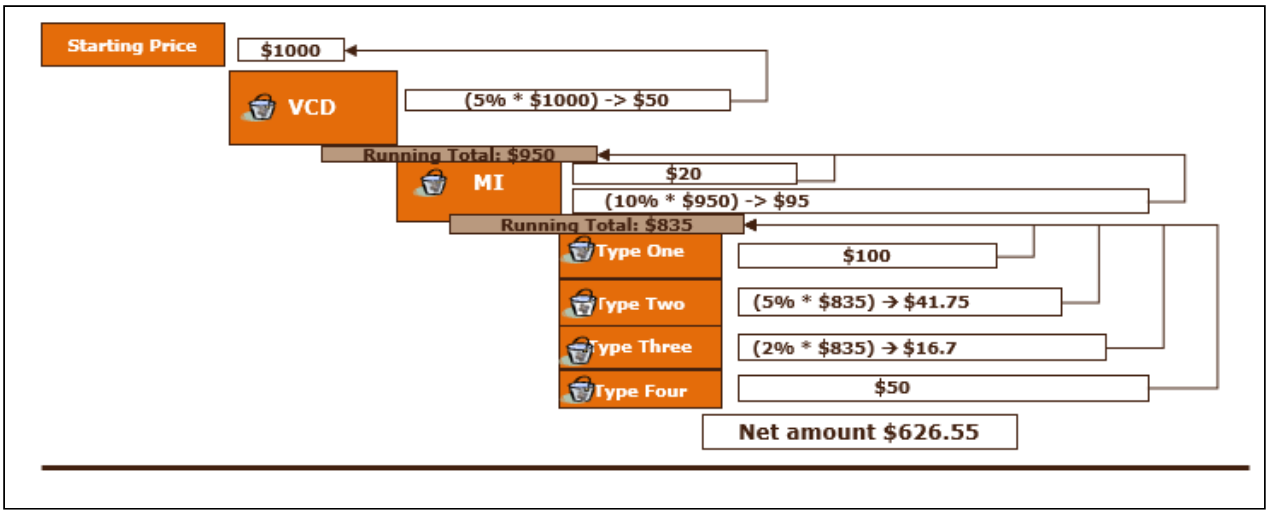
Bucket 2: Marketing Incentives (MI)

1. $20 off
2. 10% off

Bucket 3: Seasonal Promotions (SP)

1. Type One: $100 off
2. Type Two: 5% off
3. Type Three: 2% off
4. Type Four: $50 off

If the starting price is $1000, the calculation works as depicted in the graph below:



To apply bucket adjustments on line items in the cart

Prerequisite

You must have selected **Enable Adjustment Buckets** custom setting and defined a numeric value in **Max Adjustment Lines** custom setting.

1. Navigate to the cart.

2. For any line item in the cart, click the dollar icon (). The Adjust Price for <Product Name> <Charge Type> dialog box is displayed.
3. Enter the following details.
    a. **Type**: Select the type of adjustment.
    b. **Bucket**: Select the adjustment bucket.
    c. **Adjustment Applies To**: Select the price to which this adjustment applies.
        - **Starting Price**: Select this option to apply adjustments on the starting price.
        - **Previous Price**: Select this option to apply adjustments on the adjusted price of the previous bucket.
        - **Bucket <number>**: Select this option to apply adjustments on the adjusted price of a specific bucket. Buckets configured by your administrator are listed here; by default, you are allowed to apply 10 buckets.
    d. **Adjustment Type**: Select the type of adjustment.
    e. **Adjustment Amount**: Enter the adjustment amount.
4. Click **Add another adjustment** to add one more adjustment and enter the details as in Step 3. Depending on your requirement, you can select a bucket for multiple adjustments.
5. Click **Save**.

Result: Bucket Adjustments are applied for the selected line item in your cart.

> If you select a bucket that was not applied in any previous adjustments in the **Adjustment Applies To** column, the adjustment is automatically applied on the preceding bucket. If there is no preceding bucket then the adjustment is applied on the Starting Price.

Use Case: Applying Unit Level Discounts

You can apply the unit level discount on the line item level, adjustment line item level, or adjustment line items with different UoM (Unit of Measure) level. For instance, case: 10 Each (Frequency/UoM).For more information, see Configuring Unit of Measure (UOM) and Frequency Conversion Rate.

On the cart in the multi-adjustment pop-up, when you apply **% Discount**, CPQ applies the **Discount Amount** to the **Base Price** after multiplying it with **Quantity** and **Selling Term** of the bundle product.

> **You must select the custom setting `Enable Base Price Adjustment` in the System Properties > Price Settings in order to reprice the line item.**

Scenario 1: On Line Item

List Price	Base Price	Quantity	UoM	Extended Price	Adjustment Type	Discount Amount (%)	Adjusted Price
$100	$100	5	Each	$500	Base Price Discount	10	$450

Scenario 2: On Adjustment Line Item

List Price	Base Price	Quantity	UoM	Extended Price	Adjusted Price	Bucket	Adjustment Applies To	Adjustment Type	Amount	UoM	Net Price of Bundle	Unit Level Price
$100	$100	5	Each	$500	$350	Bucket 1	Starting Price	Discount %	10%	Each		$50
						Bucket 1	Starting Price	Discount	$50	Each	$400	$50
						Bucket 2	Bucket 1	Base Price Discount	$10	Each	$350	$80

Scenario 3: On Adjustment Line Item with Different UoM

List Price	Base Price	Quantity	UoM	Extended Price	Adjusted Price	Bucket	Adjustment Applies To	Adjustment Type	Amount	UoM	Net Price of Bundle	Unit Level Price
$100	$100	5	Each	$500	$395	Bucket 1	Starting Price	Discount %	10%	Each		$50
						Bucket 1	Starting Price	Discount	$50	Each	$400	$50
						Bucket 2	Bucket 1	Base Price Discount	$10	Case	$350	$80
									$1	Each	$395	$79

On the Bucket 2, CPQ calculates the price after discount based on the UoM for each unit. For instance, $1 discount is applied for 5 option products, which in turn adjusted the net price of bundle to $395 and unit level price $79.

> ⚠ If you have applied an adjustment on the promotion, an adjustment line is created and the discount is provided appropriately.

Creating Price Ramps for a Product

You can create a price ramp for a bundle product or for an option inside the bundle. You can create ramp lines to spread across time periods and add new ramp lines, create user ramps where from one period to the next the quantity (the number of users) increases (user count/quantity ramp), and also create revenue ramps where from one period to the next the level of revenue commitment increases (total price/override ramp).

> CPQ currently allows you to create and finalize a quote with 100 line items with a maximum of five ramps each. These line items can be standalone products or bundles with options. These line items can be 100 options in a single bundle.

You can also create price ramps for products with Related Pricing. When a standalone or a bundle product with options has Price Method = Related Pricing, you can create price ramps.

> CPQ considers only the following line items as sources while calculating the base price for an option:
> - One time charges
> - Recurring charges that have a period overlap with the option

While creating ramps the start date, end date and quantity are required fields. Ensure that the dates must not overlap between ramps.

> You cannot create ramps and tiers both for an option.

Creating Price Ramps on the Cart Page

On the Cart page, you can create price ramps for the products by following the steps mentioned below.

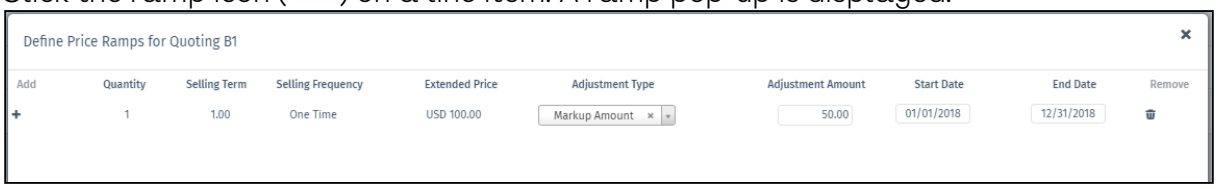
Before you begin

The administrator must enable **Enable Price Ramp** setting in the price list item.

To create a price ramp on the pricing cart page

On the cart page.

1. Click the ramp icon ( ) on a line item. A ramp pop-up is displayed.



2. Specify the price ramps for a product as described below. The options in the list depend upon the columns defined in Display Columns Settings. For more information, see Configuring Display Columns Settings.

Option	Description
Add	Click the add icon to add new ramps.
Quantity	Displays and allows you to specify the quantity of the selected product for a level in the price ramp.
Selling Term	Displays and allows you to specify the selling term.
Selling Frequency	Displays or allows you to define the usage period for a level in the price ramp for the selected product by specifying its term and the type of period.
Extended Price	Displays the selected products extended price.
Adjustment Type	Displays or allows you to specify the adjustment type for a level in the price ramp for the selected product.
Adjustment Amount	Displays or allows you to specify the adjustment amount for a level in the price ramp for the selected product.
Start Date	Displays and allows you to specify the starting date of the usage period for a level in the price ramp for the selected product. Note: Start Date must be before Ending Date.
End Date	Displays and allows you to specify the ending date of the usage period for a level in the price ramp for the selected product.
Status	Displays the status. A red **X** icon is displayed to indicate an issue.
Remove	Click the remove icon to the remove the selected ramp.

3. Click **Save**.

ⓘ The CPQ automatically enables price ramp on the second charge type, when price ramp is enabled for the first charge type on the cart page. You will observe such behavior in case if your administrator has selected **Enable Auto Ramp Creation** while configuring the price list item for the product.

If the ramp is on a bundle or standalone product, each ramp line becomes a line in the cart.

In case you defined more than one ramp line for a bundle and its options, if you change the quantity of the bundle in the primary ramp line, the primary ramp line of the options is automatically updated. You need to update the non-primary ramp lines of the options manually. This is true when Enable Auto Ramp and Auto Cascading are enabled.

Limitations

Please note that if you have selected a custom frequency, you see the following limitation:

- Option Price of Option ramp line is not added to the respective bundle.
- Selling Term is not calculated as per the Start Date and End Date.
- Canceling a ramp line during change operation creates multiple lines.
- A maximum of 50 ramp lines can be created in the current release.

Defining Tiered Pricing

Tiered pricing is used to establish rates for future usage. There is no pricing until you know the future usage.

Here is an example: In a cell phone plan, you are quoted possibly a tiered plan of $/minute for various blocks of time that you agree to pay. At the end of the billing cycle, the carrier will use the tier to determine the actual rate to apply to the minutes consumed and present you the bill for payment. At the time of quoting, only the rate is established.

If you have defined price matrices, the tiered pricing for a product is inherited from there. If you make any adjustments on the line item that has a tiered rate defined, the adjustments apply to the tiered rate and also affect the price matrices.

> You cannot create ramps and tiers both for an option.

To define tiers

1. Select a product on the options/configuration page. The tiers table appears below the selected product.
2. Type a mandatory **Tier Start Value** and **Tier End Value**. Ensure that the start and end values do not overlap. For example, the range must be 0-100, 101-200, and so on.
3. Type a mandatory **Usage Rate**.
4. To add more tier lines, click the or icon, and select **Add**.

A new tier line is added in the tiers table.

Editing Price Agreements

After product selection and configuration, you are redirected to the Shopping Cart page. The **Edit Price Agreement** button enables you to add and set prices for a product that are to be associated with your contract. When you renew your contract or regenerate a contract the defined price agreement prices are applicable.

To edit a price agreement

1. On the Shopping Cart page, click **Edit Price Agreement**.
2. Click **Add More Products** to add products the contract.
3. Select the products from the list and click **OK**. You can also search for the product using Product Name, Product Code, or Product family.
4. Specify the quantity range for the product in the **From** and **To** fields. To define product quantity ranges, such as 1-20, 21-50, click +.
5. Specify the **Start** and **End Date** ranges during which the product is valid.
6. Specify the **Unit Price**, **Flat Price**, and adjustments if any for a specific quantity range.
7. Click **Save**.

After you finalize the agreement and are redirected to your contract, the price agreement that you save will be associated with your contract as the new Price List.

Managing Contract Pricing

Contract Pricing in CPQ enables you to store contracted prices in the system. When a quote is created in CPQ, the quote price is determined by looking up the price defined in the Contract. Contract price list is a subset of the standard price list. This implies that when the same product is defined in multiple contract price lists, the former contract takes precedence. Contract Pricing is supported for standalone and bundles. Contract Pricing for Options is supported provided options are defined in the same contract price list as the bundle. Options cannot be in a different price list. Options inherit the contract price list from the bundle.

CPQ enables you (a sales representative) to know what adjustments were applied as part of a contract price list so that you can negotiate better with customers and convince them about the discounts they are getting as part of contracts. When a contract price list is generated, it holds the list price from the line item along with its negotiated contract price. When the contract price list is used in pricing a line item, you know the new contract price

(used for price calculation, that is, Base Price onward) and the original list price when the contract was negotiated.

CPQ enables you (a sales representative) to quote the correct price to the customer even if none of the contract price lists are applicable. When there is no contract price available, CPQ falls back to the quote price list to verify if it has an applicable price list item. If the quote price list does not have the applicable price list item, CPQ falls back to the *Based on Price List* to derive the price.

To select a contract pricing list

You must have an existing contract pricing list.

1. Navigate to the **Proposals** tab and select a proposal.
2. Click **Configure Products (Contract Pricing)** and add products to the cart.

> Your admin must have configured the custom **Configure Products** button. For more information, see Defining Contracts to be Used in a Quote.

You have selected to configure products having contract pricing.

To apply contract pricing on line items

The **Contract Number** field is available on Line Item record. This will apply contract pricing on specific line item only. This parameter useful when we want to apply contract pricing on specific set of products or some other criteria based on condition and not on all products. You can also enter the contract number on the line items in the cart

Applying Promotions on Line Items in the Cart

Of the two options available to apply promotions, you can apply the promo code on the cart page (PULL) or CPQ applies the promotion once your criteria are met (PUSH).

- **PULL promotion**: You can manually view the available promo codes and apply one or more promo codes to your line item (product level) and to the entire cart.
- **PUSH promotion**: This type of promotion is applied to your line items (products) or to your entire cart when the entry criteria for the promotion are met for the selected products and the entire cart. CPQ automatically updates your pricing for respective products or cart, depending on the promotion applied.

To manually apply a promo code to the shopping cart

You must have a cart that contains products to which you can apply promo codes. CPQ enables you to apply or remove promotions or coupons on line items individually or on multiple line items together (mass application). If there are line items with $0 price but eligible for promotions, CPQ populates incentive codes for those line items. Promotions and coupon applications must be mutually exclusive.

After coupons are generated from a promotion, the Apply Promotions dialog box does not list the promotion. This forces the user to enter the coupon code to apply coupons instead. Only coupons that have not reached the usage limits are displayed. When the user enters a coupon, it is stored on the Product Configuration object under the Coupon Code field.

> The **Apply Promotions** and **Remove Promotions** buttons are visible only when you select line items.

Applying Promotions or Coupons

1. Navigate to the cart.
2. Select the required line items and click **Apply Promotions**. The Apply Promotions/ Coupons window is displayed.

   > If you select one line item on the cart, CPQ displays the promotions eligible for that line item only.
   > If you select multiple line items, CPQ displays promotions based on the union of promotions that are eligible for those selected line items. When you select multiple line items and apply promotions, CPQ applies promotions only to the line items that are eligible for promotions.
   > You can verify the applied promotions by checking the incentive code on the line items. You can see only the eligible promotions' incentive code on those line items.

   - Select a single line item to apply one or more promotions or coupons to it.
   - Select multiple line items to apply promotions or coupons together (mass application).
   - Select all items to apply promotions or coupons to at the cart level.
3. On the Promotions tab:
   a. Select the required promotions. You can also search for promotions.

b. To remove any existing promotion, deselect the checkbox next to the promotion. This is possible when only one promotion is applied on the line item.

When you are applying a promotion to multiple lines:

- if the same promotion is already applied to a line, CPQ skips that line and applies the promotion to the remaining items.
- if a different promotion is already applied to a line, CPQ applies the new promotion to all items.
- if you deselect the same or different promotion already applied to a line on the Promotions tab, CPQ does not remove the promotion applied to the line.

In a *Buy X Get Y* scenario, if there are multiple line items of a benefit product (Y) with different attributes or configurations in the cart, you can apply the promotions as follows:

- If the administrator has enabled the *APTS_EnableMultiBenefitItems* setting, CPQ applies the promotion to all line items of the benefit product (Y).
- If the administrator has disabled the *APTS_EnableMultiBenefitItems* setting, CPQ applies the promotion only to the first line item of the benefit product (Y).

In a promotion of type *For Every X Get X* where X, if the administrator has defined multiple products in the criteria of the promotion, CPQ applies promotion on any of those products you add to the Cart. The X here can be multiple products or any product from a group. For example, Mobile Phones is a group of products consisting ABC 13, ABC 13 Pro, and ABC 13 Max phones. Your administrator has created a *For Every X Get X* promotion for Mobile Phones as "for every 5 mobile phones get 20% discount on the 6th mobile phone". When you add any product from Mobile Phones (that are defined in the criteria of the promotion) to the Cart; CPQ applies the defined *For Every X Get X* promotions to those products. For example, if you add 6 ABC 13 and 6 ABC 13 Max phones to the Cart, CPQ provides 20% on the 6th 6 ABC 13 and 6th ABC 13 Max phones. However, you must fulfil the quantity criteria for each product to get the benefit for that product. For example, the you must add 5 ABC 13 mobile phones to get 20% discount on the 6th ABC 13. You will not get any benefit if you add 4 ABC 13 and 2 ABC Pro phones to the Cart.
In a *For Every X Get X* or *For Every X Get Y* scenario, when a line item gets the benefit, CPQ displays the quantity of the product that received the promotion benefit, on the **Benefit Quantity** column on the Multiple Adjustments pop-up.

When a line item gets the benefit in these scenarios, the entire quantity of that line item may not always receive the benefit and hence the **Benefit Quantity** column displays the quantity that received the benefit. For example:

- You add 6 laptops to the Cart and apply a *For Every X Get X* promotion that is defined as "for every 2 laptops you buy get 10% discount on the third laptop". The **Benefit Quantity** column on the Multiple Adjustments pop-up displays 2.
- You add 5 laptops to the Cart and apply a *For Every X Get X* promotion that is defined as "for every 5 laptops you buy get 10% discount on each laptop". The **Benefit Quantity** column on the Multiple Adjustments pop-up displays 5.

When you are applying multiple promotions to a line item, refer to the field **Combine With Other Promotions**. This field indicates whether or not you can combine the promotion with other promotions.

4. On the Coupons tab:
   a. Select the required coupons. You can also search coupons.
   b. To remove any existing coupon, deselect the checkbox next to the coupon. This is possible when only one coupon is applied on the line item.
5. Click **Apply**.

⚠ You can click **Apply Promotions on Totals** to apply promotions to the total amount on the cart.

Removing Promotions or Coupons

1. Navigate to the cart.
2. Select the required line items and click **Remove Promotions**. The Remove Promotions/ Coupons window is displayed.
   - Select a single line item to remove one or all promotions or coupons applied on it.
   - Select multiple line items to remove promotions or coupons together (mass removal). When you select multiple line items, only those promotions or coupons that are common can be removed.
   - Select all items to remove promotions or coupons at the cart level. When you all line items, only those promotions or coupons that are applied at the cart level can be removed.
3. On the Promotions tab, select the required promotions. You can also search promotions.
4. On the Coupons tab, select the required coupons. You can also search promotions.

5. Click **Remove**.

The pricing on the cart is updated automatically to include the benefits of the promotions you apply. The **Incentive Adjustment Amount** denotes the adjustment amount that is applied using the promo code.

After applying promotions on your cart, you can now proceed to add more products or save and finalize the cart.

Adding Miscellaneous Items

You can use this functionality to add additional charges that are not SKU-related such as shipping charges or sales tax to the quote. These charges are not associated with any product. When you add a miscellaneous item, a new line item is created in the cart and is cascaded to proposal line item and order line item.

To add a miscellaneous item

Follow the steps below.

1. Go to the Cart page.
2. Click **Add Miscellaneous Item**. The Add Miscellaneous Item pop-up is displayed.
3. Enter the following details on the pop-up and click **OK**.

Fields	Description
Charge Type	Select the type of charge you want to add to the cart. By default, the charge types **Shipping and Handling** and **Sales Tax** are displayed, but your administrator can add more types.
Amount	Enter the amount you want to charge.
Description	Enter the description.

> You cannot modify the miscellaneous line items once you add them to Cart. You must delete and add the miscellaneous line item again if you want to modify the record.

After you click **OK**, a new line item with **Charge Type** as the name is created on the Cart page. Click the name to view miscellaneous line item details. When you finalize the cart,

the miscellaneous item is added to the quote as a proposal line item. After you accept the quote, an order is generated and an order line item is added for the miscellaneous item on the generated order.

Using Deal Guidance

You can use deal guidance to offer a deal to the customers that are considered good based on your company's criteria. For more information about how to use deal guidance, see About Deal Guidance.

Subtotaling by Groups in the Cart

A product group is a high-level grouping of the products. Product groups enable you to categorize the products which are displayed on the cart page. When you select the group level checkbox, CPQ auto-selects all the products in the group and allows you to select line items and perform actions such as **Copy**, **Remove From Cart**, and **Mass Update**. You must specify the API names in the **Config System Properties** > **Manage** > **System Properties** > **GroupBy Fields**. For more information, see Configuring Custom Settings. These line item fields can be of the Lookup, Text, and Formula types.

Viewing subtotal at the group by cart level: You can view the subtotals of the cart by a group so that you can make decisions while applying adjustments at the group line item. When you view a cart by the group, you see the relevant subtotals as separate line items. Group by subtotal has similar columns as Cart Line Item display columns.

About Quantity and Selling Term Calculation Using Default Pricing Settings

This section describes how CPQ calculates quantity and selling term using the **Default Quantity**, **Auto Cascade Quantity**, **Default Selling Term**, and **Auto Cascade Selling Term** settings.

For Auto Cascade Quantity, the price methods can be different for secondary charge types or options. The price methods need not be the same for all charge types or options. For example, if the bundle or primary charge type has *Recurring* as the price method, the options or secondary charge types can have *Per Unit*, *Recurring*, or any other price methods.

For Auto Cascade Selling Term, the price type must be the same for all. For example, if the bundle or primary charge type has *Recurring* as the price method, the options or secondary charge types must have the price method as *Recurring* only. For any secondary charge type

or option group, if the administrator has defined the default quantity and default selling term, and also selected the **Auto Cascade Quantity** and **Auto Cascade Selling Term** checkboxes, CPQ does not apply the defined default quantity and selling term to products; instead the products inherit the default quantity and selling term of the main charge type or the bundle. If the administrator has defined the default quantity and default selling term, but not selected the **Auto Cascade Quantity** and **Auto Cascade Selling Term** checkboxes, CPQ applies the defined default quantity and selling term to products and calculates pricing on the cart.

If the administrator has not defined any value for the default quantity and the default selling term of the primary charge type and bundle, but has selected the **Auto Cascade Quantity** and **Auto Cascade Selling Term** checkboxes for their secondary charge type or the option products, CPQ applies the system-generated default quantity and the selling term to those products by calculating the expected start date and end date with their primary charge type or bundle product.

The Auto Cascade Quantity and the Auto Cascade Selling Term overwrite the cart-level inputs. For example, if the administrator has selected the **Auto Cascade Quantity** and **Auto Cascade Selling Term** checkboxes for an option or secondary charge type, CPQ applies the quantity and selling term of its bundle or primary charge type, and displays the same in the cart. If you modify the cascaded quantity and selling term and click **Reprice**, CPQ restores the auto-cascaded quantity and auto-cascaded selling term from its primary charge type or bundle. However, if you modify the default quantity of the main bundle or primary charge type and click **Reprice**, CPQ overwrites the default quantity of the main bundle or primary charge type with its option product or secondary charge type.

For the Bundle and Option scenario, if any charge type of the bundle has **Auto Cascade Selling Term** set to True and the same charge type is there among the options, then at least one option must have **Auto Cascade Selling Term** set to True for the charge type.

In the following Bundle and Option scenario, a bundle has two charges (that is, primary charge type and secondary charge type) and two options. The administrator has defined the default quantity and default selling term for both primary and secondary charge types. The administrator has also selected the **Auto Cascade Quantity** and **Auto Cascade Selling Term** checkboxes for any of the option.

- It inherits the selling term and quantity from the primary charge type of the bundle only.
- It does not inherit the selling term and quantity from the secondary charge type of the bundle.
- Moreover, if administrator has not defined the default quantity and default selling term for the primary charge type, but defined the same for the secondary charge type, and then selected the **Auto Cascade Selling Term** and **Auto Cascade Quantity**

checkboxes for any option, it does not inherit the quantity and selling term of the secondary charge type.
- However, if the administrator has selected the **Auto Cascade Selling Term** and **Auto Cascade Quantity** checkboxes for the bundle product, and defined the default quantity and default selling term for the primary charge type, it inherits from the primary charge type.

In the following Bundle and Option scenario:

- If all options in a bundle have the Auto Cascade Selling Term set to True, the selling term of options is not editable on the cart. All options must always cascade selling term from the bundle only. In this case, only the bundle selling term is editable.
- If all options in a bundle have the Auto Cascade Selling Term set to False, the selling term of the bundle is calculated based on the highest selling term of options. You must not edit the selling term on the bundle.
- If a bundle has a mix of options with the Auto Cascade Selling Term set to both True and False, the selling term of the bundle is not editable on the cart. CPQ considers the highest selling term of the non-cascaded option and rolls it up to the bundle and the same term cascades to other auto-cascaded options.
- If a bundle has multiple charge types with Auto Cascade Selling Term set to True, only the term on the primary charge line is editable on the cart and the same term cascades to all other secondary charge lines. You cannot update the term of any secondary charge line if it is auto-cascaded.
- If proration is allowed and Auto Cascade Selling Term is set to True, upon a change in the selling frequency of the bundle, the term is auto-calculated based on the end date and selling frequency. The same term cascades to the option as well. Supported frequencies are: Monthly, Yearly, Quarterly, and Half-yearly.

In Use Case: Auto Cascade Quantity and Auto Cascade Selling Term for Bundle Products, scenarios 17 and 18 are important with respect to the selling term. If the administrator has defined the default selling term for any secondary charge type or options and also selected the **Auto Cascade Selling Term** checkbox, CPQ displays the default selling term while configuring that charge type or option in the cart, not the cascaded selling term.

Use Case: Auto Cascade Quantity and Auto Cascade Selling Term for Standalone Products

This section describes how CPQ calculates quantity and selling term of standalone products when different combinations of default quantity, Auto Cascade Quantity, default setting term, and Auto Cascade Selling Term are used.

Scenario	Standalone Product	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
1	Auto_NS_P_001	Standard Price	2	4	FALSE	2	FALSE	4
		Installation Fees			TRUE	2	TRUE	4
		Implementation Fees			FALSE	1	FALSE	4
2	Auto_NS_P_001	Standard Price	2	4	FALSE	2	FALSE	4
		Installation Fees			FALSE	1	FALSE	4
		Implementation Fees			TRUE	2	TRUE	4
3	Auto_NS_P_001	Standard Price	2	4	FALSE	2	FALSE	4
		Installation Fees			FALSE	1	FALSE	4

Scenario	Standalone Product	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
		Implementation Fees			FALSE	1	FALSE	4
4	Auto_NS_P_001	Standard Price	2	4	FALSE	2	FALSE	4
		Installation Fees			TRUE	2	TRUE	4
		Implementation Fees			TRUE	2	TRUE	4
5	Auto_NS_P_001	Standard Price	2	4	TRUE	1	TRUE	4
		Installation Fees			FALSE	1	FALSE	4
		Implementation Fees			FALSE	1	FALSE	4
6	Auto_NS_P_001	Standard Price	2	4	FALSE	2	FALSE	4

Scenario	Standalone Product	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
		Installation Fees			FALSE	1	FALSE	4
		Implementation Fees			FALSE	1	FALSE	4
7	Auto_NS_P_001	Standard Price			TRUE	1	TRUE	6
		Installation Fees			FALSE	1	FALSE	6
		Implementation Fees			FALSE	1	FALSE	6
8	Auto_NS_P_001	Standard Price			FALSE	1	FALSE	6
		Installation Fees			FALSE	1	FALSE	6
		Implementation Fees			FALSE	1	FALSE	6

Scenario	Standalone Product	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
9	Auto_NS_P_001	Standard Price			TRUE	1	TRUE	6
		Installation Fees			TRUE	1	TRUE	6
		Implementation Fees			FALSE	1	FALSE	6
10	Auto_NS_P_001	Standard Price			TRUE	1	TRUE	6
		Installation Fees			FALSE	1	FALSE	6
		Implementation Fees			TRUE	1	TRUE	6
11	Auto_NS_P_001	Standard Price			FALSE	1	FALSE	6
		Installation Fees			TRUE	1	TRUE	6

Scenario	Standalone Product	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
		Implementation Fees			FALSE	1	FALSE	6
12	Auto_NS_P_001	Standard Price			FALSE	1	FALSE	6
		Installation Fees			FALSE	1	FALSE	6
		Implementation Fees			TRUE	1	TRUE	6

Use Case: Auto Cascade Quantity and Auto Cascade Selling Term for Bundle Products

This section describes how CPQ calculates quantity and selling term of bundle and option products when different combinations of default quantity, Auto Cascade Quantity, default setting term, and Auto Cascade Selling Term are used.

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
1	ND_Bundle1	Standard Price	2	4	FALSE	2	FALSE	4

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
	ND_Option1	Implementation Fees			TRUE	2	TRUE	4
	ND_Option2	Installation Fees			FALSE	1	FALSE	4
2	ND_Bundle1	Standard Price	2	4	FALSE	2	FALSE	4
	ND_Option1	Implementation Fees			FALSE	1	FALSE	4
	ND_Option2	Installation Fees			TRUE	2	TRUE	4
3	ND_Bundle1	Standard Price	2	4	FALSE	2	FALSE	4
	ND_Option1	Implementation Fees			FALSE	1	FALSE	4
	ND_Option2	Installation Fees			FALSE	1	FALSE	4

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
4	ND_Bundle1	Standard Price	2	4	FALSE	2	FALSE	4
	ND_Option1	Implementation Fees			TRUE	2	TRUE	4
	ND_Option2	Installation Fees			TRUE	2	TRUE	4
5	ND_Bundle1	Standard Price	2	4	TRUE	1	TRUE	4
	ND_Option1	Implementation Fees			FALSE	1	FALSE	4
	ND_Option2	Installation Fees			FALSE	1	FALSE	4
6	ND_Bundle1	Standard Price	2	4	FALSE	2	FALSE	4
	ND_Option1	Implementation Fees			FALSE	1	FALSE	4

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
	ND_Option2	Installation Fees			FALSE	1	FALSE	4
7	ND_Bundle1	Standard Price			TRUE	1	TRUE	6
	ND_Option1	Implementation Fees			FALSE	1	FALSE	6
	ND_Option2	Installation Fees			FALSE	1	FALSE	6
8	ND_Bundle1	Standard Price			FALSE	1	FALSE	6
	ND_Option1	Implementation Fees			FALSE	1	FALSE	6
	ND_Option2	Installation Fees			FALSE	1	FALSE	6
9	ND_Bundle1	Standard Price			TRUE	1	TRUE	6

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
	ND_Option1	Implementation Fees			TRUE	1	TRUE	6
	ND_Option2	Installation Fees			FALSE	1	FALSE	6
10	ND_Bundle1	Standard Price			TRUE	1	TRUE	6
	ND_Option1	Implementation Fees			FALSE	1	FALSE	6
	ND_Option2	Installation Fees			TRUE	1	TRUE	6
11	ND_Bundle1	Standard Price			FALSE	1	FALSE	6
	ND_Option1	Implementation Fees			TRUE	1	TRUE	6
	ND_Option2	Installation Fees			FALSE	1	FALSE	6

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
12	ND_Bundle1	Standard Price			FALSE	1	FALSE	6
	ND_Option1	Implementation Fees			FALSE	1	FALSE	6
	ND_Option2	Installation Fees			TRUE	1	TRUE	6
13	ND_Bundle1	Standard Price (Primary)	2	4	FALSE	2	FALSE	4
	ND_Bundle1	Maintenance Fees (Secondary)	3	5	FALSE	3	FALSE	5
	ND_Option1	Implementation Fees			TRUE	2	TRUE	4
	ND_Option2	Installation Fees			FALSE	1	FALSE	4

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
14	ND_Bundle1	Standard Price (Primary)	2	4	FALSE	2	FALSE	4
	ND_Bundle1	Maintenance Fees (Secondary)	3	5	FALSE	3	FALSE	5
	ND_Option1	Implementation Fees			FALSE	1	FALSE	4
	ND_Option2	Installation Fees			TRUE	2	TRUE	4
15	ND_Bundle1	Standard Price (Primary)	2	4	FALSE	2	FALSE	4
	ND_Bundle1	Maintenance Fees (Secondary)	3	5	TRUE	2	TRUE	4
	ND_Option1	Implementation Fees			TRUE	2	TRUE	4

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
	ND_Option2	Installation Fees			TRUE	2	TRUE	4
16	ND_Bundle1	Standard Price (Primary)	2	4	TRUE	1	TRUE	4
	ND_Bundle1	Maintenance Fees (Secondary)	3	5	TRUE	1	TRUE	4
	ND_Option1	Implementation Fees			TRUE	1	TRUE	4
	ND_Option2	Installation Fees			FALSE	1	FALSE	4
17	ND_Bundle1	Standard Price (Primary)	2	4	FALSE	2	FALSE	4

Scenario	Bundle and its Options	Charge Type	Default Quantity	Default Selling Term	Auto Cascade Quantity?	Output Quantity in Proposal	Auto Cascade Selling term?	Output Selling Term in Proposal
	ND_Bundle1	Maintenance Fees (Secondary)	3	5	FALSE	3	FALSE	5
	ND_Option1	Implementation Fees	6	7	TRUE	2	TRUE	7
	ND_Option2	Installation Fees	8	9	FALSE	8	FALSE	9
18	ND_Bundle1	Standard Price (Primary)	2	4	FALSE	2	FALSE	4
	ND_Bundle1	Maintenance Fees (Secondary)	3	5	FALSE	3	FALSE	5
	ND_Option1	Implementation Fees	6	7	FALSE	6	FALSE	7
	ND_Option2	Installation Fees	8	9	TRUE	2	TRUE	9

Using Price Waterfall

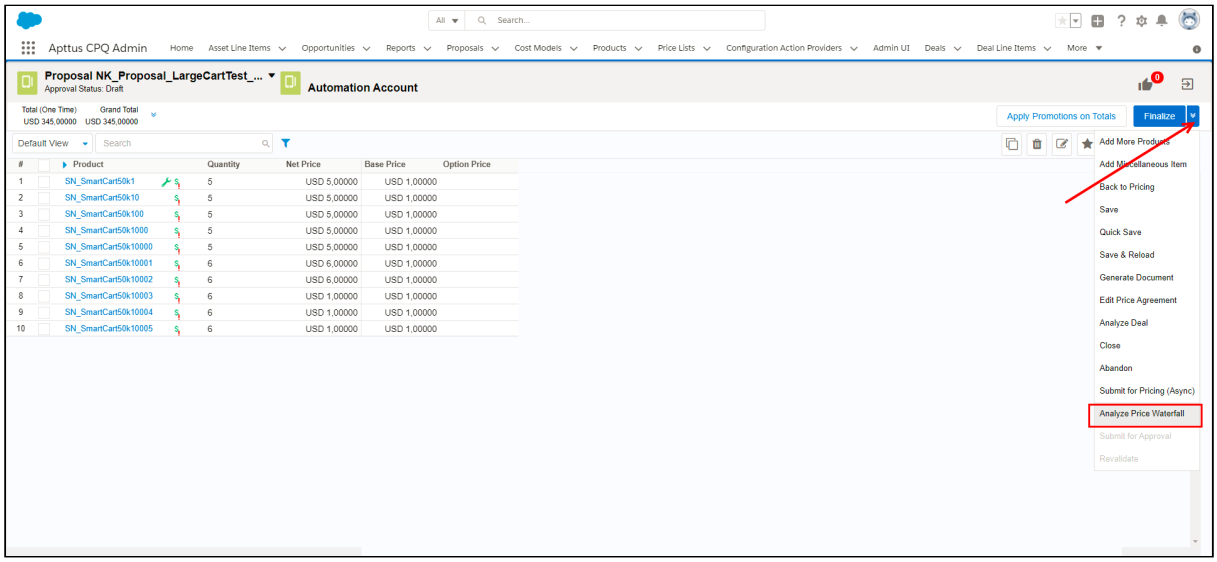
The Price Waterfall page enables you to view and analyze the price for each line item. In addition to that, you can also make manual adjustments to the price points derived in the price pipeline to analyze the key performance indicators.

Launching Price Waterfall:

You can launch a price waterfall chart using one of the following:

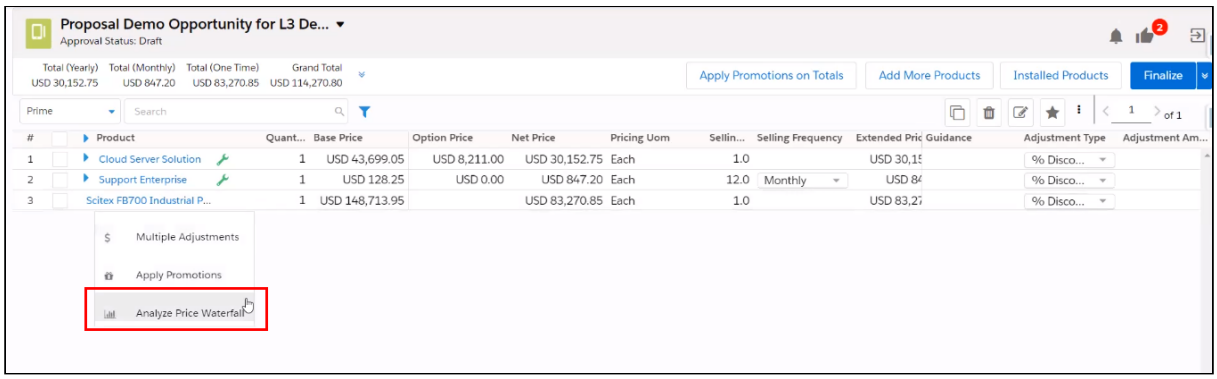
Launching Price Waterfall from the Actions menu

1. Navigate to the Catalog.
2. Click **Add products**.
3. Click **Go To Pricing**. The Cart page is displayed.
4. Click **Analyze Price Waterfall**. The Price Waterfall UI is displayed for the first line item.



To Launch Price Waterfall for a specific Line Item

1. Navigate to the Catalog.
2. Click **Add products**.
3. Click **Go To Pricing**. The Cart page is displayed.
4. Place the cursor on the line item and click the kabab menu of desired line item.



Analyzing Price Waterfall

1. Navigate to the Catalog.
2. Click **Add products**.
3. Click **Go To Pricing**. The Cart page is displayed.
4. Click **Analyze Price Waterfall**. The Price Waterfall UI is displayed for the specific line item.

1.

	Field/ Icon/ Feature	Description
1	List of Line Items	Displays a list of Line items added to the cart and Enter a keyword to search a required product/option, and the search will show the product/parent bundle and its options. Click the required option to view the waterfall chart.
2	Price Summary	The price summary of the cart line item.
3	Waterfall Chart	The waterfall chart is a bar graph where Line Item Fields are plotted vertically, and Cost is plotted horizontally. Click the editable or modifiable bar to update the values.
4	Chart Options and Icons	**Per Unit and Gross tabs:** Per Unit displays price waterfall in terms of unit prices. Gross shows price waterfall in terms of the gross price.
		Menu icons to display the Price waterfall information: You can select whether to view price waterfall in the form of a bar chart ( ) or table ( ).
5	Action buttons	To Reprice or Finalize the cart.

	Field/ Icon/ Feature	Description
6	Line Item fields	The cart line item fields are displayed based on the waterfall setup to a price pipeline. You must enable the 'Is Editable' to a cart line item field to edit the value while analyzing the price waterfall chart. Working with cart line item fields: a. Enter the desired values in the editable cart line fields. b. Click **Reprice** or **Finalize**.

Working with Manual Adjustments in the Price Waterfall UI

Suppose the price point is marked as modifiable in the price pipeline definition. In that case, you can make relevant manual adjustments to the particular price point in the Create Manual Adjustment window. Any manual adjustments you perform on the cart appear against the price point marked as a net price.

To add manual adjustments:

1. Click the price point bar in the price waterfall chart.

   ⚠ You can click only if the price point is set as modifiable by the Admin in the price pipeline definition.

2. Enter the following fields and click **Apply** to view the updated price waterfall chart.

Field	Description
Name	Enter a name for the adjustment.
Adjustment Type	Select adjustment type from the drop-down list. ⓘ You can only apply a *% Discount* as an Adjustment Type for the Net Price.

Field	Description
Amount	Enter the required adjustment amount.
Apply To	Select the source of price point from the following: • Previous: For the running price. • Prior Price Point: for the last point price. • List Price: For the List Price, which is the first price point in the waterfall chart.

Finalizing Products

This section explains the following topics:

- Finalizing the Shopping Cart
- Opening Cart in Read-only mode
- About Product Configuration
- Disabling Cart Versioning
- Revalidating the Product Configuration
- Using Express Proposal

Finalizing the Shopping Cart

This is the final step before taking the finalized set of products and prices back to the Quote/Proposal.

You can review the cart and its pricing and if you are satisfied with the configuration and pricing, you can finalize the cart.

To finalize the cart

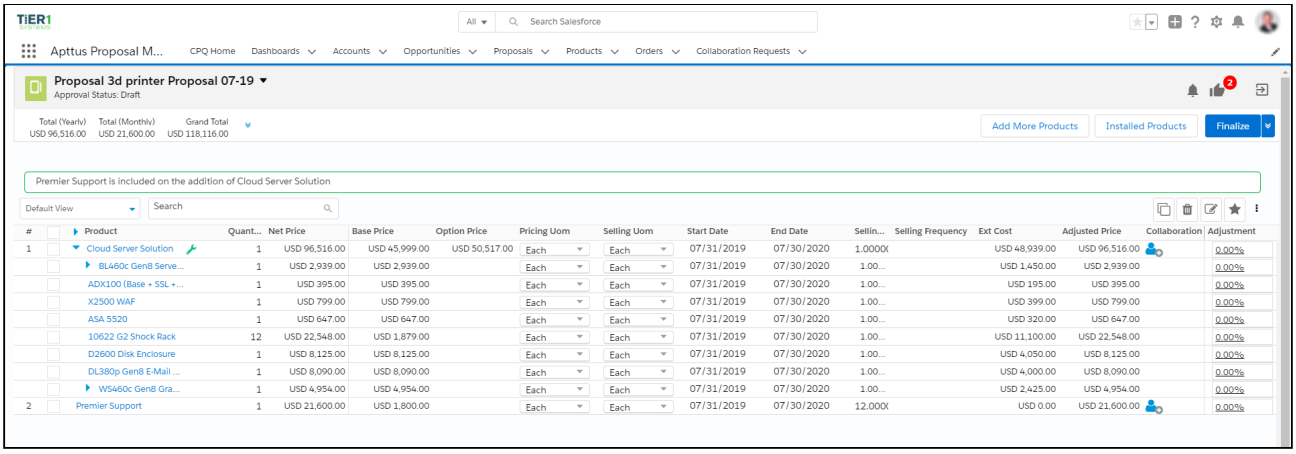
You must click the **Reprice** button to accept any pending changes in pricing. You must also resolve any validation errors that may appear due to validation rules.

1. On the Cart page, click **Finalize**.

> ⓘ The **Review** and **Finalize** button on the Cart page keeps you on the same page without finalizing the configuration. This feature has been deprecated.

2. To confirm the product configuration and pricing, and return to the Quote/Proposal detail page, click **Finalize**.

> CPQ updates the **Finalized Date** field on the quote and cart only if you update the status of the cart as *Saved* before re-finalizing the cart using API.



CPQ processes the finalization of the cart synchronously or asynchronously based on the value of the **QTC Profile** field you selected while creating the quote. The following table lists the fields **QTC Profile** and the mode in which the cart is finalized.

Value	Description
Regular	CPQ finalizes the synchronously unless the administrator has defined **asyncFinalize** as *true*.
Split	CPQ processes cart pricing and finalization asynchronously in a Smart Cart flow.
Enterprise	The cart finalization is processed asynchronously without dividing the product configuration.

You can see a progress bar on the Quote Details page that indicates CPQ is processing the finalization request. After the completion, all the product configuration and pricing information is displayed on the Quote Detail page. The progress bar only appears if the admin has enabled it, refer to Configuring Progress Tracker for Async Operation.

> If you want to change the **Account** associated with a quote with finalized or saved product configuration, you must abandon the cart and reconfigure it.

If your product contains Usage Tiers and Attributes, there might be a delay for the system to create these records while you land on the Quote/Proposal page. You must wait for the system to finalize the configurations before going for document generation or asset creation.

After the quote is finalized, the quote/proposal page appears. If any workflow processes have been setup to trigger approvals based on criteria, they will typically occur at this point, and the **Approval Stage** inside the Quote/Proposal record will be set to **Approval Required**. As soon as the Approval Stage changes, the **Submit for Approval** button becomes available and you can submit the Quote for approval to management if any approval processes have been configured. Once approved, the Quote/Proposal is generated and presented.

Opening Cart in Read-only mode

This feature allows you to open your Product Configuration in read-only mode. In read-only mode you are only able to see the line items in the cart, you cannot edit any fields. All the action buttons are disabled in a read-only cart. To open the read-only cart, you can use a separate **Configure Product** button with read-only access on the Quote Details page that your admin must configure. When you open a cart in read-only mode, no new product configuration is created on the Quote Details page.

You can perform the following tasks in a read-only cart:

- Search for products on the cart.
- Open or create new Cart Views.
- Use advanced filters.
- Save Cart Views with the filters.
- Close the cart

You can open a cart in read-only mode even when the pricing of the line item is pending. When you open such a cart, the priced line items are displayed and a progress bar is displayed with the status of the ongoing pricing process of the remaining line items. The action buttons remain disabled until you exit the read-only mode.

About Product Configuration

Product Configuration is created after you click the **Configure Products** button on the Quote Details page. Product Configuration stores the details about the product you configure on the cart, including the options, attributes, and pricing configuration. After the products are configured they are added to Product Configuration the

Configurations section on the Quote Details page. The status of the Product Configuration reflects the state of the configuration on the cart.

The following table describes the status of the Product Configuration based on the status of the cart.

Product Configuration Status	Description
Ready For Approvals	After you have finished the configuration and the configuration requires approval.
Ready For Finalization	After the configuration is approved and the cart can be finalized.
Finalized	When you finalize the cart.
Superseded	When you reconfigure a finalized cart then the status of previously finalized configuration changes to **Superseded**
Saved	When you save the cart.
Pending Approval	When you submit the configuration for approval.

CPQ clones the Product Configuration when you reconfigure a finalized cart and creates a new version of the Product Configuration. When you reconfigure a finalized cart, the status of previous Product Configuration is set to **Superseded**. The status of the cloned Product Configuration then reflects the current status of the cart. The Product Configuration is deleted when you abandon the cart.

> ⓘ If you want to change the **Account** associated with an existing quote with finalized or saved product configuration, you must abandon the cart and reconfigure it.

Disabling Cart Versioning

When you disable cart versioning, only one product configuration is created on the Quote Details page. Disabling Cart versioning disables the creation of **Superseded** configuration, if you reconfigure a cart after finalizing it, the existing product configuration is updated. If cart versioning is disabled, then on abandoning the cart the configuration is deleted and all changes done so far on the cart are lost. The prompt message on the click of **Abandon** informs you about the same if Cart Versioning is disabled.

Conga recommends you to enable **Disable Cart Versioning** if you do not need to maintain multiple configuration records and it also improves the cart launch time for the saved cart.

To disable cart versioning, select the **Disable Cart Versioning** checkbox on the Quote Details page.

> ⚠ It is not recommended to change the setting on an inflight quote.

Revalidating the Product Configuration

You must revalidate your saved quote for the product structure changes by clicking **Revalidate**. Saved quotes created in the past might have a product for which structure, rule, or price has been changed. The changes like addition or removal of options in bundle structure. This can affect the validity of such a quote when you reopen the configuration. When you open such a saved cart, CPQ shows a message informing the you that there are changes to the products in cart which must be revalidated. Upon clicking **Revalidate**, CPQ displays a popup listing the products that are changed.

The products on the revalidation popup are grouped into the following sections:

- Product Structure changes: Products in this section upon revalidation are marked as *Pending Configuration* and you must reconfigure such product before proceeding.
- Pricing Changes: Production in this section upon revalidation are marked as price pending and must be repriced.
- Changes provided by the Revalidation Callback: Products in this section are sent to the Revalidation Callback to be revalidated using custom logic defined in the callback. For more information on the callback, see Revalidation Callback Class

Revalidation is of the following types depending upon the changes made to the product:

- Hard: CPQ forces you to revalidate the cart while disabling all actions except **Save**, **Quick Save**, **Abandon**, and **Close**.
- Soft: CPQ displays a warning on finalize.

The following table describes the changes and type revalidation action you must perform.

Change	Revalidation Type
Major product version change: For example, changing Product Version from 1.00 to 2.00	Hard

Change	Revalidation Type
Minor product version change: For example, changing Product Version from 1.00 to 1.01	Soft
Inactive price list Item	Soft
Expired price list Item	Soft
Deleted price list Item	Soft
Inactive product: For products that are not active post quote saving, CPQ automatically deactivates the price list item and displays a message on the cart. When you revalidate, CPQ removes the product from cart.	Soft

When you create a new or change the list of a price list item, revalidation is not required. You must reprice the cart manually to see the changes as CPQ does not automatically detect such changes in price list item for save quotes.

To revalidate the cart

1. When you click **Revalidate** on the Cart page, a popup is displayed in which you see the products that are changed.
2. Do one of the following:
    - Click **Apply** to accept the changes.
    - Click **Cancel** to proceed with the previous structure of the product.

You see the **Revalidate** button on the previously saved Cart page when the Admin has made changes to the Product Configuration and has updated the Version field of the product.

> When you click Revalidate, CPQ shows *No records to display* a message in case if there is no change in the product structure.

Displaying revalidation messages

When a price list item for the bundle or options is revalidated, CPQ warns you with the message that some items may get changed upon revalidation.

CPQ displays the following revalidation messages based on the business scenarios.

For example,

- When an optional product is removed from a bundle, the system shows the message: `"Product Structure for the following product(s) has changed. Do you want to revalidate the product(s)?"`
- When a line item is expired, the system shows the message: `"The following line items have invalid price list items. During revalidation, some items in the cart may change or be removed."`
- When a version is changed in the quote, the system shows the message: `"Some items in the cart may have changed. Please revalidate the cart."`

For more information, see Configuring Field Set Settings.

Using Express Proposal

The Express Proposal feature allows you to generate and send proposal in fewer clicks. After you add and price the products in the cart, click **Express Quote** to finalize the cart, select a template, generate the quote and email the document to the customers, and mark the quote as presented. You can use the Express Proposal from Catalog, Configuration, and Cart pages by clicking the **Express Quote** button. Typically, you must finalize the products on the cart page, go to the Quote details page and complete the steps to generate the document first then present the quote to the customer. This feature allows you to complete all the tasks on one page rather than navigating several.

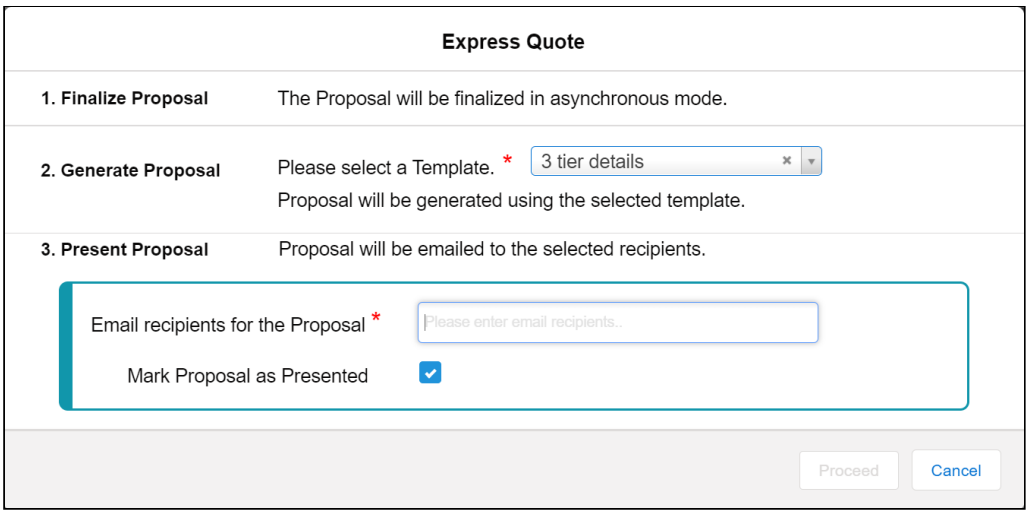
> The administrator must add this button to the Catalog, Configuration, and Cart pages. Also, the Checkout popup is displayed only if the administrator has configured it. For more information, see Configuring Express Proposal.

Click the **Express Quote** button after you configure products and complete the pricing. If there are no warnings or errors, CPQ either automatically executes the tasks in the order shown in the following table, or displays the Express Quote popup listing the tasks mentioned in the below table depending on the feature's configuration.

Order of Execution	Task	Description
1	Finalize the proposal	The proposal is finalized in asynchronous mode. You can track the progress of task completion using the Async Progress Tracker feature on the Quote Details page.

Order of Execution	Task	Description
2	Generate the proposal	The proposal is generated in PDF format only, using one of the following templates: • The default template defined in **Default Template Name** in Proposal System Properties, • A single template filtered using query template filters for the quote specifically • The template you select from the Checkout popup.
3	Email the proposal	By default, the email is sent to the quote owner. You can search and add more recipients from the contacts existing in the org. You cannot send the email to IDs that are not defined as a contact in the org. In this case, create a contact for that external ID first.
4	Mark the proposal as presented	Select the checkbox to change the status of the quote to Presented.

If the popup is displayed, click the **Proceed** button to start tasks.



Do not use this feature:

- To generate the document in any format other than PDF.
- To merge the generated document with any other document.

- If you require any approvals before finalizing the cart.

Finalizing Quotes

This section explains the following topics:

- Repricing Quotes
- Analyzing Quotes
- Auto-Synching Cart Line Items to a Quote
- Generating and Presenting Quotes
- Accepting Quotes
- Synchronizing Quotes with Opportunities
- Activating Orders

Repricing Quotes

Use the Reprice Quote feature to reprice all the parameters with a single click from the Quote/Proposal page.

Pricing is determined by Quote header level parameters. Normally after the first time Quote configuration is completed, and you were to adjust the Quote level parameters, you had to go back to the shopping cart to Reprice. Doing this one by one and then repricing is time-consuming and ineffective. With the Reprice Quote button, there is no need to do this, you can change product pricing and/or rules, and click the Reprice Quote button to reprice all the parameters without going to the cart page.

To reprice a quote from the Quote/Proposal page

You must complete a quote configuration at least once.

1. Navigate to the **Proposals** tab and select a proposal.
2. Click **Reprice Quote**.

All the quote level parameters are repriced on the Quote/Proposal page.

Analyzing Quotes

Analyze Quote page helps you to view and manipulate the cost and profitability of each **Line Item** and **Summary Group Item**, one at a time.

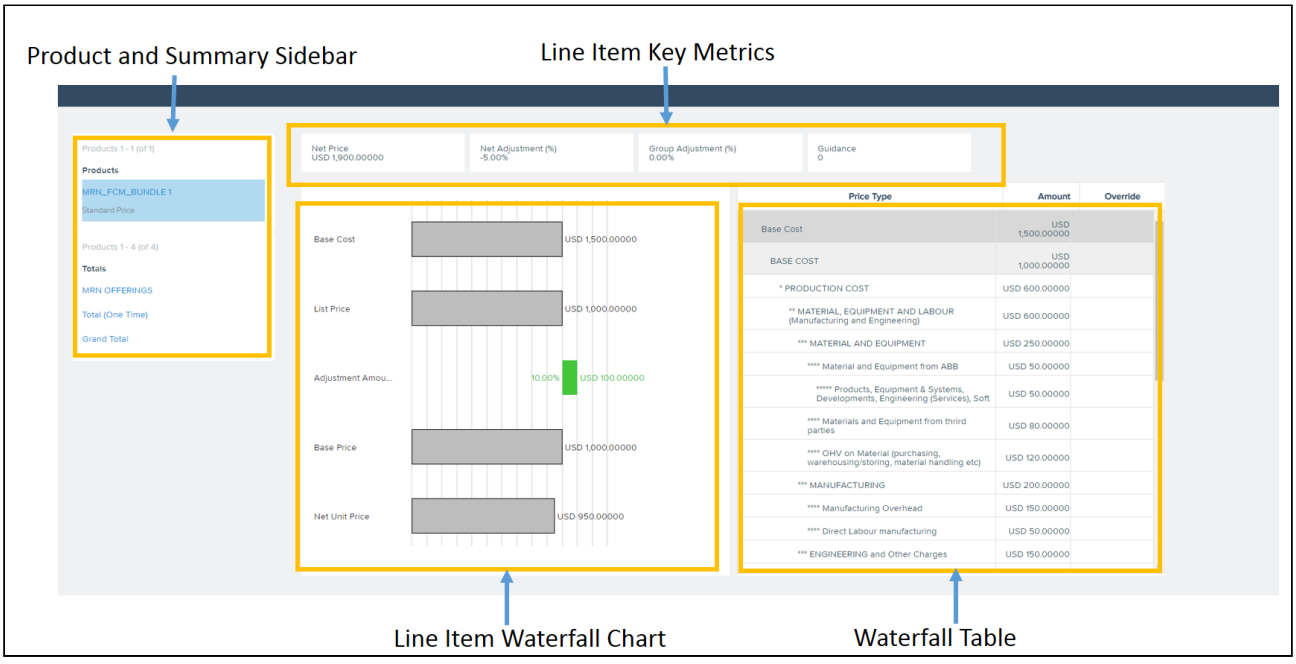
Enhanced Analyze Quote Page
CPQ shows the following UI improvements in the Analyse Quote page:
The column names or headers are available for the optional columns. If you configure a column from **Config Setting** > **Display Column Setting** > **Display Type : Option Cost & Profitability Cost**, then on clicking **Analyze Quote** from the cart shows **Optional Columns** on the page.

Add Analyse Quote button on the Cart

Go to **Config Settings** from (All tabs) > **Display Action Settings** and setup an *action* button for **Analyze Quote**.

Screen Elements

The following illustration shows the Analyse Quote page, calling out the key elements appearing on this page.



Product and Summary Sidebar: This section displays a list of Line items and Summary Group Items the way you would see it on the cart. You can select any of the line items or the summary groups to see its corresponding waterfall chart and cost breakdown.

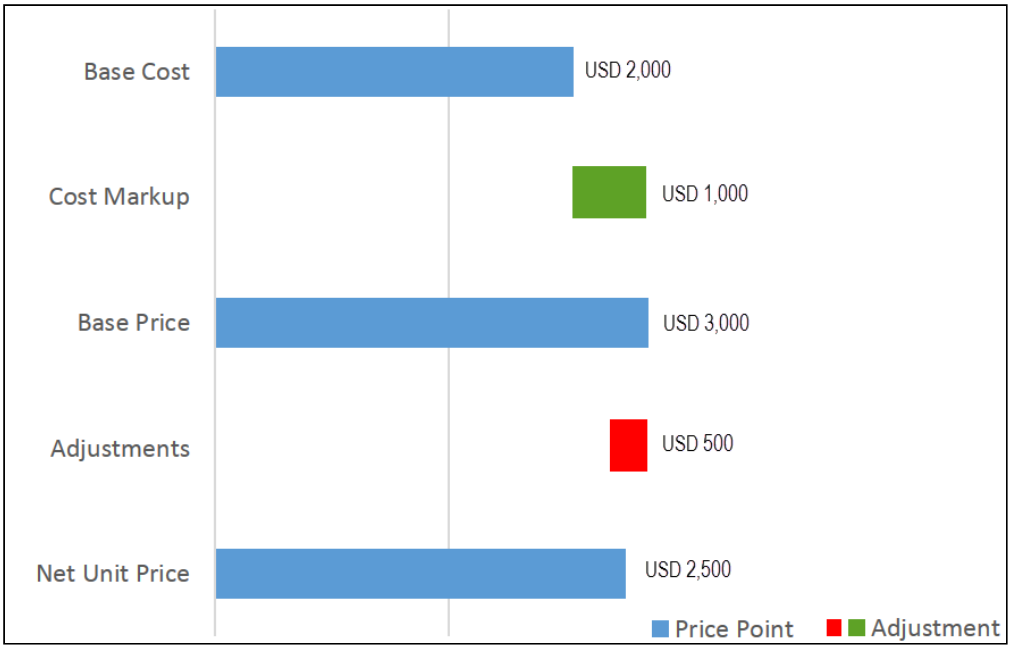
All information visible on this page is either for a single Line Item or a single Summary Group Item.

Line Item Key Metrics: Fields that are useful in deriving the costing profitability such as Margins, Discount etc. can be added to this Key Metrics.

Line Item Waterfall Chart: Various Line item Currency fields can be used to plot the Waterfall chart. It is basically a bar chart where Line Item Fields are plotted vertically and their values (Price) are plotted horizontally.

You can see two kinds of bars:

- **Price Point:** These bars begin from the base of the y-axis. Usually the Line Item fields such as List Price, Invoice Price etc. are the Price Points in the Waterfall chart.
- **Adjustment:** These bars are adjustments to the Price Point or Costs. These bars start from the endpoint of the previous bar and based on the type of adjustment, whether negative or positive, their endpoint is derived. As shown in the following illustration, Cost Markup is an addition to the Base Cost and hence the Adjustment bar starts from the point Base Cost ends. Similarly, Adjustment is a deduction from the Base Price which starts from where the Base Price ends but goes inward indicating a negative adjustment.



Waterfall Table: This table translates the Waterfall Chart with all cost breakups. You can override certain costs and click **Reprice** to see the Waterfall with a revised cost. In the "Amount" field, the currency value is left aligned on the Analyze Quote UI page.

After analyzing the deal, you can also collaborate with multiple users to finalize the pricing. For more information, see About Quote Lifecycle Collaboration.

CPQ uses the Base Price field of Line Item to determine what the Currency Code to be displayed in the cart.
As you do not want Base Cost to be seen by specific users for various reasons but Base Price to be made available to all users. Therefore, CPQ uses Base Price rather than Base Cost to determine the currency code to be used. The currency code as set in the user locale is displayed for all the currency fields.

Auto-Synching Cart Line Items to a Quote

When Auto-sync is turned on, the system does validate for errors and auto-syncs the cart lines with the Proposal Lines. You can view the synced line items in the Quote/Proposal detail page under the Line Items related list. Auto-sync is done when the following conditions are satisfied:

- No error produced by the Validation callback
- No Constraint Rules error
- No line items in price pending state
- No Must Configure product is in configuration pending state

Generating and Presenting Quotes

You can generate that document from the quote and present that to the customer.

The section explains the following topics:

- Generating Quotes
- Using Proposal Document Generation
- Presenting Quotes
- Sending Documents for eSignature Using Conga Sign

Generating Quotes

This scenario guides you to generate a Quote/Proposal document based on the information you provided.

After the quote is finalized, and if any approval processes have been configured in your organization, the **Approval Stage** changes to **Approval Required**. As soon as the Approval Stage changes, the **Submit for Approval** button becomes available and you can submit the

Quote for approval to management. For an overview of the various Approval stages, see Approval Stages.

After the quote is approved by an approver, the Approval Stage changes to *Approved*. You can now generate a proposal document to present to a trading partner. When you initiate the **Generate** action, a screen appears allowing you to select additional information about the document to be generated. Additionally, if document protection has been enabled, a document can be generated with the appropriate protection options.

To generate a proposal document

You must ensure that the Approval Stage is *Approved*.

1. Click **Generate** to generate a Proposal document.
2. From Options > 1. Select Output Format, select one of the following formats:
   - *DOC*: To generate an MS Word document.
   - *RTF*: To generate a proposal document in Rich Text Format.
   - *PDF*: To generate a PDF protected proposal document.
3. Select a template from the list.
4. Click **Generate**

   or

   Click **Submit**, if you have a large quote. Such document generation takes more time and you must refresh the page to see the generated document.

   > The **Submit** button is only visible if the Large Document Generation is configured in your org by the admin. For more information, see Configuring Document Generation for Quote.

The proposal document is generated and can be found in the **Notes and Attachments** related list. The proposal document is automatically attached to the Quote and can be viewed via the View action link. The Approval Stage changes to *Generated*.

You can now present the quote you generated.

Using Proposal Document Generation

Proposal Doc Gen provides the ability to select a template, select the output format, merge the Quote/Proposal document with additional documents and sequence them, email the collated file with attachments and preview the different versions of documents. As a Sales Rep, you can invoke the new wizard from the cart as well as from the Quote/Proposal header page.

This reduces the number of clicks and provides an integrated experience from the shopping cart as well as the Quote/Proposal header. All the functionality resides in a single-page application that is smooth and easy to use. Consider the following scenarios where this feature can be very useful:

Use case 1: After confirming the product selection and pricing, as a Sales Rep, you would like to instantly generate a Quote/Proposal from the Cart, merge some marketing material and product data sheets with my generated Quote/Proposal and send the document in an email to the customer.

User case 2: After receiving the email that my quote (that I had submitted earlier for approval) has been approved, you want to quickly generate the Quote/Proposal document, merge some marketing material and product data sheets with my generated proposal and send the document in an email to the customer.

> For information on how to configure Document Generation, refer to Configuring Document Generation for Quote.

The process flow of the Doc Gen is described below:

Accessing the Doc Gen Wizard

You can access the Doc Gen Wizard from the Quote/Proposal header by clicking **Send Proposal** button, which is available with the proposal management package. The major two differences in accessing the Doc Gen Wizard from the cart are that you must choose a proposal template to generate a proposal document and you get quick access to action buttons, such as **Go to Pricing**, **Add More Products**, **Installed Products**, and Go to Proposal below the Quote/Proposal name

Generating a Quote/Proposal document

To generate a quote document, click **New Proposal Document**. A popup appears where you can choose or search a template for your Quote/Proposal document (filtered using Query Templates). You can choose the Output Format for your document. Only *doc*, *docx* and *pdf* formats are supported, depending on the format your admin has configured, using Proposal Document Output Format. Using Proposal Document Output Format, if you've chosen *docx* and *pdf*, the drop-down list shows both. If you do not define any format, the system takes *pdf* as the default output format.

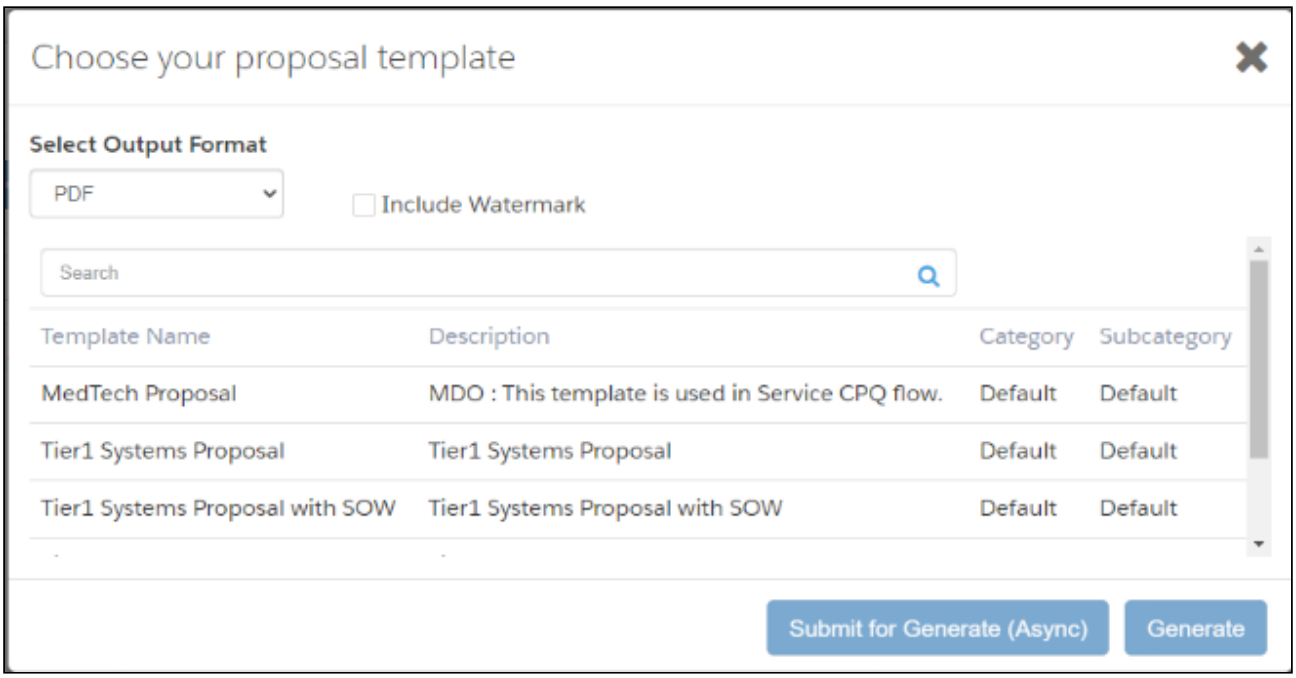
You can click **Generate** button to generate a new Quote/Proposal document. Upon clicking **Generate** button, CPQ generates documents in sync mode and you remain on the popup till the document is generated.

When you need to generate a document for a quote with a large number of line items, you can use **Submit for Generate(Async)**. CPQ generates the documents in async mode when you click **Submit for Generate(Async)** button. When CPQ processes documents in async mode, the popup is closed and you are directed back to the Doc Gen page. Document generation may take few minutes, you must refresh the page to see the generated document. The preview of the generated document is displayed on the right pane.

> ⓘ The **Submit for Generate(Async)** button is displayed only if the administrator enabled **Enable Submit Merge Call** setting in Proposal System Properties.

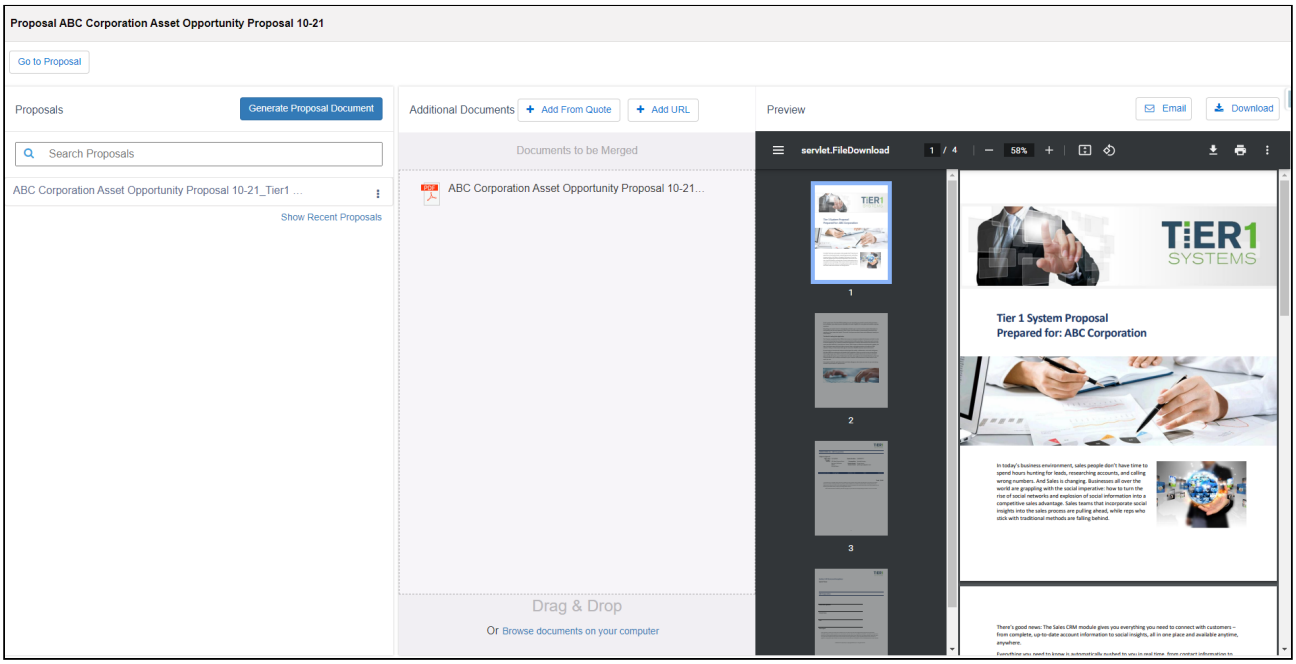
You can also select a default document that is automatically generated upon clicking the **Send Proposal** button. The default document is generated when Fast Doc Gen is enabled. The default document is generated using a default template that is defined in the org by the administrator.

> ⓘ The document is generated automatically only if the administrator has enabled **Enable Fast Doc Gen** Proposal System Property. For more information, see Configuring Document Generation for Quote. You cannot use Fast Doc Gen feature in a quote in Smart Cart flow.



The Quote/Proposal document generated using the Doc Gen Wizard follows the naming

convention as: *Quote/Proposal Name + Template Name + Date stamp*. Such documents appear in the left pane of Proposal Doc Gen wizard and as attachments in the **Notes & Attachments** related list.



Renaming and deleting a proposal

You can use the buttons **Rename** and **Delete** to rename and delete the proposal respectively. Click the menu icon(⋮) to display the actions.

Previewing generated documents

Once generated, the system automatically previews the generated document on the right pane of the window. In the case of multiple documents, you can preview documents by clicking the name of the generated document. Note that Preview is available only for documents created in *PDF* format. While using Files, the documents are not previewed automatically. You must click the thumbnails of the documents to preview.

If the documents are generated asynchronously or encounter a timeout limit, a record in the preview related list is generated on the Quote details page.

> ⓘ Timeout limits may exceed when you generate a large and complex document. You may define the timeout limit in **Merge Call Timeout Millis** in Proposal System Properties.

Merging with additional documents

You can add additional documents and merge them with the generated document.

Adding documents from Quote

You can attach additional documents to the Quote/Proposal by clicking **Add from Quote**. This button fetches all the documents (that are not generated using Proposal Doc Gen) from the **Notes & Attachments** related list of a Quote/Proposal. Select a document and click **Add to Merge Docs** to allow the system to consider this document for merge with the generated Quote/Proposal document. The merging happens only after you click **Merge Documents** on the right pane.

Adding URL to the documentation

You can add document hyperlinks from external sources and merge them with the Proposal. Click **Add URL** to launch a pop-up that gives you the ability to enter valid document hyperlinks and its corresponding Title. You can add multiple URLs by clicking **Add Another** link at the bottom of the pop-up.

> Ensure that you have added the endpoint for the document URL that you want to list on the Proposal Doc Gen from **Setup** > **Security Controls** > **Remote Site Settings**.

As soon as you attach more than one document to your Quote/Proposal (apart from the generated Quote/Proposal), the right pane of the window grays out and enables a single button called **Merge Documents**. You can choose to merge all the generated and offline documents in the **Documents to be Merged** section and turn them into a single document. After your documents are merged, the system displays the collated document in the Preview pane on the right. The merged document appears as a record under the **Document Collate Infos** related list on the Quote/Proposal.

> During the proposal workflow, the system detects the acceptable file type (For instance, .docx, .doc, .pdf) and displays a distinct icon for the file format.
> Also, when you try to merge a file with an unsupported format or the file size is more than 4 MB, the system alerts you with an error message.

Adding a local file from computer

You can add attachments from your local storage to the Quote/Proposal by clicking **Browse to documents on your computer**. You can simply drag and drop an offline document

to the Doc Gen wizard and the system considers it for merging with the Quote/Proposal document.

> The size limit of the file you can upload from local storage is 3.5 MB.

Sending the generated document to the customer

After the sales rep has reviewed, merged and finalized the content in the final Quote/Proposal document, he can send the collated document to the customer for further actions. Buttons like **Email** and **Download** are available for the following purposes:

Action button	Usage
Email	Use this button to send the Quote/Proposal (or merged) document to your customer for review. The dialog box allows you to provide the following information: • **To** • **CC** • **BCC** • **Subject** • **Email Folder** • **Email Template** • Any additional documents. You can search and select Contacts that you want to add in the **To**, **CC**, and **BCC** fields. To search the Contact name, enter the first 3 letters of the name and you can see the matching results in the drop-down. CPQ, by default, populates the email information of the **Primary Contact** field in the **To** field. > You can only edit the email body of Email Templates of type *Text*. CPQ displays a warning when you select any other template. If you have defined an Email Signature in **My Email Settings**, CPQ automatically adds that at the end of the email body. You can select the **Mark Proposal as "Presented"** checkbox to ensure that the Approval Status of the Quote/Proposal is changed to *Presented* and the **Presented Date** field is updated after you send the Email.

Action button	Usage
Download	Use this button if you want to download a local copy of the generated Quote/Proposal (or merged) document.

Presenting Quotes

After generating a proposal document, it is presented to the trading partner. The system sends an email with the generated Quote attached. If the email is sent, the Approval Stage will be automatically be set to *Presented*. If you prefer to print or present the Quote a different way, you can manually set the Approval Stage to *Presented*.

To present a quote/proposal

The Approval Stage must be in *Generated* stage.

1. Click **Present**. The Present Proposal page appears.
2. Select the proposal document to be attached and click **Next**. The **Send an Email** page appears.
3. Enter details as you would for an email and click **Send**. A confirmation popup appears. For more information on email templates and defining custom email templates, see Managing Email Templates. You can select an email template, attach additional files, and perform a spell check before sending an email. The system will prompt whether the email was sent.
4. Click **Yes**.

> ⓘ You must add contact details in Account to present a quote.

The email is sent and the **Approval Stage** status changes to *Presented*. The **Accept** button is enabled.

After the quote has been accepted by the trading partner, click the Accept button.

Sending Documents for eSignature Using Conga Sign

After you complete generating and presenting the documents, you can send documents to gather customer signatures using Conga Sign. On the Quote Detail page, click the **Send with Conga Sign** button. You are redirected to the Conga Sign application. For more information, see Sending Documents for eSignature.

You can perform the following task while using Conga Sign:

- Send a document from the attachments in the quote or by uploading them from the local machine.
- Use documents in Word or PDF format.
- View the list of recipients in the **Conga Sign Recipient** related list.
- Receive email notification of every action taken by the recipient on the sent document.
- View status of the transaction in the **Conga Sign Transaction** related list. The status changes according to the action of the recipient such as opening, signing, or rejecting documents.

Accepting Quotes

After a trading partner has accepted a proposal, you can finalize a quote/proposal by accepting the proposal, synchronizing products and prices with the related Opportunity and then create an Agreement.

After the quote/proposal has been presented to the trading partner either manually or via an email, the **Accept** button is enabled. Selecting Accept signifies that the trading partner has accepted the Quote. If the current Quote is not primary, clicking the **Accept** button will make all other quotes non-primary and the current one primary.

Simultaneously, the line items in the **Summary** related list is synchronized back to Opportunity Products for forecasting purposes.

> ⓘ The related opportunity must have a Price List associated with it or else the summary items will not synchronize. The products do not need an entry in the Price List, but the Opportunity must be associated with a Price List.

Alternatively, if the synchronization is not required to happen automatically, you can choose when to synchronize with the Opportunity by using the **Make Primary** and **Synchronize with Opportunity** buttons.

To accept a Quote/Proposal

The Approval Stage must be in *Presented* stage.

1. From Actions, click **Accept**. The Approval Stage changes to **Accepted**.
2. Do one of the following:
    - To create an agreement, click **Create Agreement**.
    - To create an agreement with line items, click **Create Agreement with Line Items**.

- To synchronize the quote with an opportunity, click **Synchronize with Opportunity**.
- Synchronizing Quote/Proposal With Opportunities
  Synchronization ensures that both the Quote/Proposal and Opportunity accurately reflect the current state of the Quote. A best practice for sales pipeline reports and forecasts in Salesforce is to display the amount of revenue and the products or services that are forecasted to be sold.

The contracting process begins with the creation of an agreement. For detailed information, see Creating an Agreement.

Synchronizing Quotes with Opportunities

Synchronization ensures that both the Quote/Proposal and Opportunity accurately reflect the current state of the Quote. A best practice for sales pipeline reports and forecasts in Salesforce is to display the amount of revenue and the products or services that are forecasted to be sold.

Salesforce provides forecast reports that drive off of the amount of the Opportunity and the associated Opportunity product records. Opportunity products are stored in a standard Salesforce object and represent the line items for an Opportunity. When using Salesforce without Conga, products are added to the opportunity manually or by synching with a standard Salesforce Quote record.

CPQ can synchronize Proposal Line Items from a Quote Proposal to the related Opportunity. This action updates the Opportunity Amount and maintains accuracy between the Quote/ Proposal and the Opportunity, both on the record and in reports. This is especially important if you have configured sales reports using Opportunities with Products. With Conga, they can continue to use these reports and the deep functionality that CPQ provides.

When a Quote/Proposal is synced with an Opportunity, the Proposal Line Item records are copied to the Opportunity Products related list. The Amount of the Opportunity will reflect the summed total of the opportunity products. The Sales Price field on the Opportunity Product record is populated by the Proposal Line Item.Net Price field.

While a quote/proposal and an opportunity are synced, any addition or change to the list of products in the quote/proposal syncs with the list of products in the Opportunity.

If a quote/proposal is re-configured and the shopping cart is finalized, the new proposal line items will be copied to the opportunity's product related list.

- Assigning the Default Pricebook
  A Pricebook is the standard Salesforce version of a price list or container of products

and pricing. In CPQ, the Price List object is used as a pricing container. Standard Salesforce synchronization requires that a Salesforce Pricebook is assigned to the Opportunity to enable synchronization between Quote and Opportunity.

- Marking Quotes as Primary
  An opportunity can have multiple quotes, but it can only sync with one quote at a time. The sync process looks for Quote/Proposal records where the Primary field is marked as true.
- About Auto Sync and Manual Sync
  The Synchronize with Opportunity action button is visible on a Quote/Proposal that is marked as Primary.

Assigning the Default Pricebook

A Pricebook is the standard Salesforce version of a price list, or container of products and pricing. In CPQ, we use the Price List object as the pricing container. Standard Salesforce synchronization requires that a Salesforce Pricebook is assigned to the Opportunity to enable synchronization between Quote and Opportunity. The Pricebook may be assigned to the Opportunity manually or automatically using a trigger. In either case, the Standard Price Book should be used.

To assign a default pricebook

You must have an existing Quote/Proposal created from an Opportunity.

1. Go to the **Opportunities** tab and select an existing opportunity.
2. Scroll down to the Products related list, and click **Choose Price Book**.
3. From the Price Book picklist, select **Standard Price Book**.
4. Click **Save**.

A standard price book is assigned to the opportunity. Go to the Quote/Proposal and make it Primary.

Marking Quotes as Primary

An opportunity can have multiple quotes, but it can only sync with one quote at a time. The sync process looks for Quote/Proposal records where the Primary field is marked as true. To ensure that only one Quote/Proposal is marked as Primary at any time, CPQ provides a **Make Primary** action button. The button is conditionally displayed on a Quote/Proposal page if:

- There is an Opportunity related to the Quote/Proposal

- The Quote Proposal is not already marked as Primary

The first Quote/Proposal created on an Opportunity is automatically marked as Primary. When the Make Primary action is invoked with the button on a Quote/Proposal, the Primary checkbox on any other Quote/Proposal related to the same Opportunity is unchecked.

> When a second Quote/Proposal is related to an Opportunity, it must be made Primary before configuring and finalizing products.

To make a quote as primary

You must have an existing non-primary quote/proposal created from an opportunity.

1. Go to the required quote.
2. Click **Make Primary**. The Make Primary page is displayed.
3. Click **Continue** to make the quote as primary.

About Auto Sync and Manual Sync

To manually sync the Quote/Proposal click the **Synchronize with Opportunity** button on the Quote Details page, which is visible on a Quote/Proposal that is marked as Primary.

To automatically synchronize Primary Quotes, enable the **Auto Sync with Opportunity** setting in **Proposal System Properties** custom setting. This setting syncs Quote/Proposals that are marked as Primary each time the shopping cart is Finalized. CPQ syncs the Quote/Proposal with opportunity again each time you click **Accept Quote** when **Auto Sync with Opportunity** is enabled.

Activating Orders

Once you have accepted the Quote/Proposal, an Order is created with the Status = *New*. All the Proposal Line Items are copied to the Asset Line Items and Order Line Items. To proceed further with creating Assets for the products/services listed in your Quote/Proposal, you must activate this Order.

You can activate an Order by following any of the below-mentioned procedures:

- By selecting **Auto-Activate Order** check box on the Quote/Proposal page:
  If you choose this option, once you accept a Quote/Proposal, the Order is created and activated simultaneously.
- By manually entering an **Order-Activation Date**:
  If you choose this option, you have to manually enter a Date and Time in the **Order Activation Date** field on the Quote/Proposal page.

Once activated, you can see the status of the Asset Line Items and Order Line Items is changed to *Activated*. See Managing Assets for more details.

Managing Assets

Asset Management is based on the order information that you have already defined in your sales quote and the resulting contracts.

When your customers agree to purchase a product, the product changes from an agreement line item to an order line item in CPQ. The status of the agreement in this stage is *Accepted* and the order is now ready to be activated and realized as an asset for that customer. After you receive payments for an order is when the order line item changes to an Asset line item.

With Asset-based Ordering, you can

- create quotes for new products and services based on existing assets
- create quotes to modify existing products and services
- modify in-process orders that have been submitted for fulfillment
- renew, change, swap, and terminate an existing service
- have visibility into the asset transaction history during customer interactions

After being processed and fulfilled, new quotes and orders result in new assets that are listed on a customer's account and a line item from an asset becomes an Installed Product. You can access the Accounts page to see a customer's information and their purchase history.

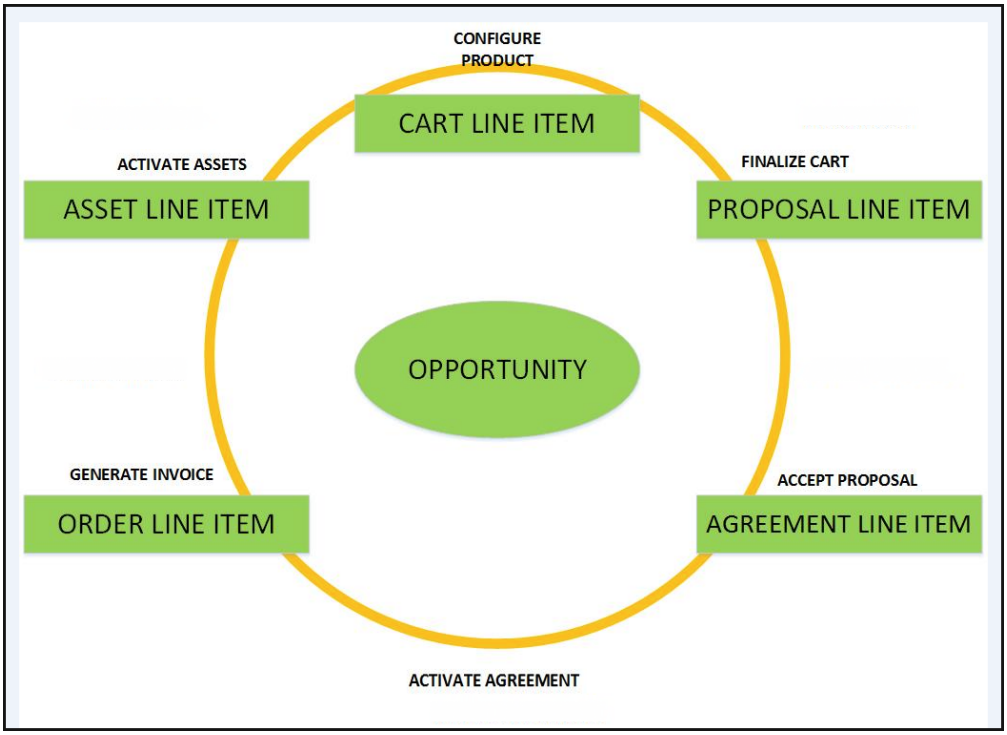
The assets are created upon finalization of a Quote/Proposal and/or activation of an Agreement. If your product contains Usage Tiers and Attributes, there might be a delay for the system to create these records on the Proposal Line Items. You must refresh the Quote/Proposal or Agreement page and wait for the system to finalize the configurations before accepting a Quote or activating an Agreement.

In this Section:

About Assets and Asset Line Items

Any product or service that your customers buy, become that customer's assets. While your organization may have a unique and customized Order Fulfillment process, all the processes and procedures in Asset Management start with the Line Item in an order.

The following image describes the life-cycle of a Line Item in the Quote-to-Cash flow.



After your customers agree to purchase a product, the product changes from a Proposal Line Item to an Order Line Item. The status of the Quote/Proposal in this stage is **Accepted** and the order is now ready to be invoiced.

After you receive payments for an order is when the order line item changes to an **Asset** line item.

CPQ allows Customer Service and Sales Representatives in your organization to perform the following actions to manage assets.

1. Generate asset records from orders
2. Renew the term for an asset
3. Change the **Term**, **Quantity**, **Options**, and **Attributes** of a product
4. Cancel products and subscriptions
5. Swap one purchased product with another
6. Create quotes for new products and services based on assets
7. Create quotes to modify existing products and services
8. Get visibility into the asset life cycle during customer interactions

Before you manage a customer's assets, you must understand how they use Asset-Based Ordering.

About Asset-Based Ordering Flows

Before you start using Asset-Based Ordering, you must understand how you, as a Sales and Customer Service Representatives in your organization, can use the ABO functionality.

There are total 3 flows that you can consider for using ABO. Ensure that you understand and choose one of the flows to implement error-free ABO.

1. **Quote/Proposal Flow:** This is one of the signature events in Asset-based Ordering involving Quote/Proposal. This is the normal flow of going through the Quote/Proposal to an Order and then to the Assets. Once you accept a Quote/Proposal, the Order is generated. Upon activation of the order, assets are created.
2. **CSR Flow:** This is the flow designed to enable the Sales Rep to skip the creation of a Quote/Proposal. No signature event, such as creation of an Agreement or a Quote/Proposal is involved. A new button called **Asset Manager** is configured on the Accounts page to help the Sales Rep navigate directly to the Installed Products page. However, if you perform asset renewal through the Asset Manager flow, the **Use the Proposal End Date** option is not displayed on the Confirm Renewal intermediate page.
3. **Contract Flow:** This is one of the signature events in Asset-based Ordering involving Contracts/Agreements. The assets are created through Contracts/Agreements using the normal Contract lifecycle flow. Once you finalize a Contract (containing Agreement Line Items), the corresponding Asset Line Items are created.

Quote/Proposal Flow

In this flow, you configure the Quote/Proposal and finalize a configuration. Once the configuration is finalized, you present the configuration to the customer. Upon acceptance from the customer, you click the **Accept** button on the Quote/Proposal. An order is created for this Quote/Proposal, which upon activation, creates the Asset Line Items with the *New* status. You can configure the Quote/Proposal again and navigate to the Installed Products page to view all the active assets. No custom triggers are required here.

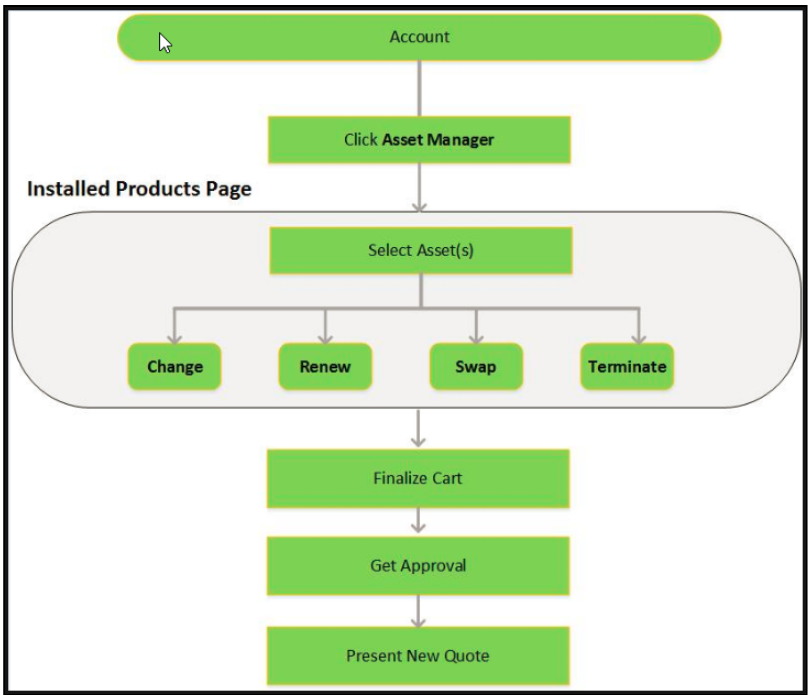
Flow: Create **Quote/Proposal** > **Configure Products** > **Finalize** > **Accept** > Activate **Order** > **Installed Products** page.

CSR Flow

There can be scenarios when you do not want to configure the Quote/Proposal again to land on the Installed Products page. For these type of scenarios, you can configure the **Asset Manager** button on your Account page. Clicking **Asset Manager** creates an Order

and navigates you directly to the Installed Products page. In this flow, you skip the navigation of going to the Installed Products page through the **Quote/Proposal > Configure Products**.

The following image illustrates the CSR flow from the Sales and Customer Service perspective.



An order remains in the Draft status in spite of the fact that CPQ displays ready for activation date, hence ABO operation does not get applied on the asset line items. If you keep the status of an order as Draft, you need to Accept the order before starting the CSR flow.

Contract Flow

A contract contains Agreement Line Items, Asset Line Items or Order Line Items. In this flow, you follow the normal Contract Lifecyle flow of creating a Contract, presenting it to your customer and finalizing the Contract. Here, the Asset Line Items are created when you finalize and activate a Contract. CPQ provides the ability to synchronize the changes in Assets with the changes in a Contract (see details here). The changes you make inside a Contract (such as change in **Contract End Date**) will be reflected in the Assets contained in

that Contract. Similarly, when you perform actions on your purchased Assets (such as changing the **Quantity** of an Asset), these changes will be reflected in the Contract.

Flow: Create **Agreement** > **Configure Products** > **Finalize** Quote/Proposal > **Accept** Quote/Proposal > **Activate** Agreement > **Installed Products** page.

About the Installed Products Page

Installed products can include both products and services. A line item from an order becomes an installed asset if it is marked as an asset in the product record. With Asset-based ordering, you can create quotes and orders based on a customer's existing assets. Asset-based ordering is particularly useful when your business offers complex service products, such as phone services and equipment.

The Installed Products page lists all assets in decreasing order (▼) of the values in the **Created Date** column by default. You can sort assets dynamically, a single column at a time, by clicking the column header. You can switch the sorting to another column by clicking a different column header. Assets are sorted in the descending order first and a subsequent click sorts the assets in the ascending order (▲).

On the Installed Products page, you can Renew, Terminate, Swap and Change an existing standalone, fixed, or a configurable bundle asset. For each transaction involving ABO actions, the system creates a new order while you are navigating on the Cart. The status of the Asset Line Item is not changed until you finalize the Cart and activate the Order containing that Asset Line Item.

For more information on permitted configurations for the Installed Products page, see Configuring Installed Products Settings.

> **Note**
>
> Please ensure you add Allowable Actions for assets in order to perform Swap, Change, Renewal or Termination. For more information, see Defining Allowed Actions for Assets.

For a sales-driven industry with products such as equipment (new purchase) and service (purchased for an equipment), you might want to achieve the following:

- As a manufacturing sales representative, for a service flow, you want to view the **Renew** and the **Terminate** button. For an equipment flow, you want to view the **Change** and the **Swap** button.
- As a subscription sales representative, you want to see the **Renew** button only when I make a purchase for a quote of type *renew*. For an add-on quote or an upgrade

quote, you do not want to see the **Renew** button. For example, managing the renewal of a magazine subscription.

The **Swap** and **Renew** buttons are disabled for the superseded asset line items. CPQ checks for the **Status** field of related asset line item and disables these buttons for an asset line item on the Installed Products page. This is to prevent a sales rep from using the superseded asset line items in the renewed opportunities and Quote/Proposals.

For example, you have an Asset A and you perform **Change>Config** action to create an asset B. The relation between these two assets in the related asset line item object is displayed as follows:

From	Relation Type (From)	To	Relation Type (To)
Asset A	Is Superseded By	Asset B	
Asset B		Asset A	Is Superseded By

After this transaction, **Swap** and **Renew** buttons are disabled for Asset A.

Viewing Assets from an Account

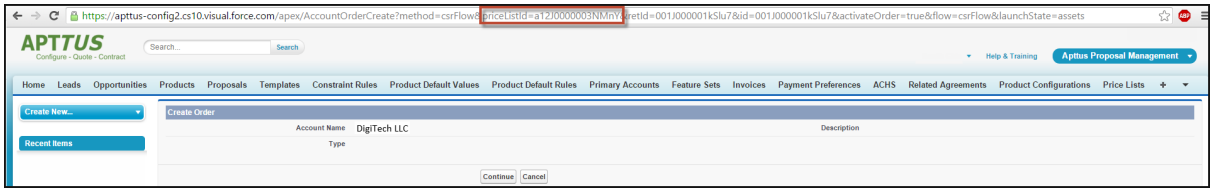
An account with assets has the **Asset Manager** button. Refer the configuration part and ensure that the **Asset Manager** button is configured correctly in your org.

> If you perform asset renewal through the Asset Manager flow, the **Use the Proposal End Date** option is not displayed on the Confirm Renewal intermediate page.

To view the installed products or assets for an account

1. Click **Asset Manager**.

2. Verify the **Price List ID** in the URL. Click **Continue**.



3. On the **Installed Products** page, select the assets that you want to manage.

> **Note**
>
> If your administrators have disabled an asset action, the button is grayed out for that asset. For example, from the image above, if your administrator has not defined *Renew* as an allowed action for Advanced Analytics, the **Renew** button is grayed out and you cannot renew that asset for this customer. Also, if you select two or more assets from the list of installed products, the **Swap** button is automatically disabled.

You can Renew, Terminate, Swap, and Change an existing standalone, fixed, or a configurable bundle asset.

Viewing Installed Products in Different UIs

Installed Products page is redesigned for quick search of assets and better control of column visibility. The Grid UI for Installed Products gives better performance in terms of page load time and intuitive navigation.

> Only the Asset Grid UI is supported on the Installed Products Page.

> **Styling for Installed Products Page**
> You can apply column settings to the Installed Products page. You can change coloring, formatting, and styling of the asset grid page. This helps you customize the look and feel of the Installed Products page for better readability.
> You can see the **Search** bar adjacent to the installed products header area on the right corner on both Asset and Assets Grid pages.
> You can see a floating menu for the Asset Grid page similar to the Cart Grid page.

> You can see pagination at the top of the Asset Grid page, which helps proper page alignment. Also, notice that the left side search filter section gets collapsed by default.

On the Installed Products page, Primary Service columns are available for Equipment type assets only. From the new user interface, you can

- Expand or Collapse the product attributes
- Use Pagination to control the display of assets on a single page and to extreme left or right of the view as chosen from the options
- Search using a product or option (with a full or a partial string
- Perform a basic search or an advanced search
- Create asset views
- View assets across account hierarchy

To enable the new interface, append *grid* at the end of the Installed Products URL.

For example,

```
https://apttus-config2.cs13.visual.force.com/apex/Cart?
cartStatus=New&oldConfigId=a1IW0000000gybHMAQ&configRequestId=a1cW000000044MrIAI&id=a
1IW0000000gyt7MAA&flow=LAngFlow&launchState=cart#/assets
```

should be changed to

```
https://apttus-config2.cs13.visual.force.com/apex/Cart?
cartStatus=New&oldConfigId=a1IW0000000gybHMAQ&configRequestId
=a1cW000000044MrIAI&id=a1IW0000000gyt7MAA&flow=LAngFlow&launchState=cart#/assetsgrid
```

Similarly, you can switch back from the grid UI to the classic interface by removing *grid* from the URL.

You can use the following features to navigate the Installed Products homepage if you are using the Assetsgrid (New UI):

Product Search: Enables you to search existing assets. Enter the name of the product and click the magnifier icon. All products that are relevant to the search string (full name of the product or partial name) are displayed. Click the **(+)** icon to use the **Basic** and **Advanced Search** options. You can also search an option product, all bundles with their options are displayed.

Account Hierarchy icon  : Click the **Accounts Hierarchy** icon and select one or more a displays accounts hierarchy as per show assets settings. By default, the app will display all assets of the current account of the proposal.

Items per Page: You can select the number of assets that must be displayed on the home page of the Installed products page. You can select a maximum number of 200 items that must display on a page.

Managing Views for Assets Grid

This section provides information on managing user-level private views for the assets grid.

> If your administrator has created any view in the Admin UI and shared it with you, you can select the view. However, you cannot edit or delete the view created by your administrator. For more information, see Configuring Installed Products Settings.

To create a view

1. Click **Installed Products**. The Installed Products Page is displayed.
2. Click the **Views** drop-down list and select **Create New View**. The Create New View dialog box is displayed.
3. Enter a name for the view in the **View Name** field.
4. Drag and drop the required columns from the **Available** list to **Viewing** list.
5. Select the **Set as Default** check box to make this view as the default view on the Installed Products Page.
6. Click **Save**.

To edit an existing view

1. Click **Installed Products**. The Installed Products Page is displayed.
2. Click the **Views** drop-down list, select the required view, and click **Edit**. The Create New View dialog box is displayed.
3. Make the required changes and click **Save**.

To delete a view

1. Click **Installed Products**. The Installed Products Page is displayed.

2. Click the **Views** drop-down list, select the required view, and click **Delete**. The view is deleted.

Searching Assets

CPQ provides the **Search Products** feature on the Installed Products Asset Grid page only and lets you search by option as well. You can search the install base with a full or partial search on the option name and then you get all the bundles with that particular option including nested bundle options. You can apply other filters on the install base in addition to the search based on the option name. If the product catalog does not have bundles with that option, it presents a message that bundles with that option could not be found in the install base. Option-based search works in accordance with existing features such as Asset Source, Account Hierarchy, and Pagination.

In the **Search Products** field, you can enter a keyword, option, option name, or partial name in order to search assets. Based on the following values, CPQ determines what asset lines associated with a product need to be searched:

- Product Name
- Product Family
- Product Code
- Product Description
- Configuration Type

After fetching results using **Search Products**, you can further filter your search using **Account Hierarchy** and **Filter By** functions. The **Filter By** function provides two options: **Basic Search** and **Advanced Search**. The **Advanced Search** allows you to define your own query to select asset line items. It uses only the Asset Line Item fields for the definition of the query. CPQ does not currently support searching records based on related objects such as attributes. The **Advanced Search** filters search results (**Search Products**) if there are any criteria applied while searching assets using **Search Products**. If there are no criteria applied, the **Advanced Search** directly works on asset line items and retrieves the first 20,000 records.

When you clear the search criteria from a secondary or tertiary level search, it does not impact the search criteria of higher level search. For example, clearing the search criteria of **Filter By** fields does not reset the search criteria of **Account Hierarchy**.

The current pagination on the Installed Products page supports only 20,000 records. If the install base has more than 20,000 records, you must apply a filter on the install base to see the relevant records. The filter can be **Search Products**, **Accounts Hierarchy**, or **Filter By**.

If you do not apply any filter, CPQ displays only the first 20,000 records on the Installed Products page.

For example, the install base has 100,000 bundles (as counted by the bundle primary line). After you apply the **Search Products** filter, CPQ retrieves 45,000 bundles out of which CPQ displays only the first 20,000 bundles (as counted by the bundle primary line) on the Installed Products page. You can apply other filters (**Accounts Hierarchy** and **Filter By**) on this subset of 45,000 bundles to bring the search results further down to 20,000 bundles or less.

> Alternately, you can directly apply filters (**Accounts Hierarchy** and **Filter By**) on the original set of 100,000 bundles. If that search retrieves under 20,000 records, all results are displayed on the Installed Products page. If that search retrieves 45,000 records, only the first 20,000 records are displayed and you must modify the search further to see all intended records.

Sorting Product Items Per Page

By default, you can sort 10, 15, 25, 50, 100, 200 assets per page. In addition, you can set the custom value in the **Config User Preferences** custom setting. This feature works in association with the search and filter functions to display search records per page. You can use the **Previous** and **Next** buttons to move to the next or the previous pages. You can also jump to a certain page using the page search component.

Displaying Assets from the Account Hierarchy

You can view all the assets of either the parent or the child on the **Installed Products** asset grid page and search them using various criteria.

To set the account hierarchy

1. Under **Custom Settings** > **Installed Product Settings** > **Show Assets**, specify **Parents, Children**.

   > - If **Show Assets** is blank, on the asset grid page you cannot see an icon for **Account Hierarchy** and CPQ shows only assets from the current account.
   > - If **Show Assets** set to **Parents, Children**, on the asset grid page you can see an icon for **Account Hierarchy** and CPQ shows assets from the current account as well as from Parent and Children.

2. Add or update the Accounts that you want to be displayed in the **Account Hierarchy** screen on the asset grid page.
3. Execute the batch job in CPQ Console > Maintenance Console > Account Hierarchy, whenever you add or update the account's parent or create a new account with parent.
4. Click **Account Hierarchy** icon on the asset grid page. The **Account Hierarchy** pop-up opens.
5. Select the account for which you want to view the list of assets. The default account is always marked in bold in the **Account Hierarchy** pop-up.
6. Click **Apply** in the pop-up to see the list of asset for a particular account in the right pane. You can see the hierarchy of assets under the **Account Hierarchy** icon in the left pane.

Pricing Assets

Unless you override the **List Price**, the price of the asset depends on the values you enter for defined the Price List Item fields under the **Tax and Billing** tab. Before you make changes to an asset, consider the pricing settings done on the product PLI. For more information, see Defining Tax and Billing for Products.

If you select the **Is Asset Pricing** checkbox on the Proposal Line Item detail page, when you Renew, Swap, Change, or Terminate the asset, CPQ accounts for any list price overrides on the original order to compute the price of the asset.

Renewing an Asset

Renewals are the most common and effective ways to retain existing customers and drive growth and sales for your business. Asset renewals eliminate the scope of pricing errors by extending the agreement. Therefore when you renew an asset, you basically regenerate the asset life-cycle for a new duration. CPQ supports the following types of renewals:

- Manual renewal of assets
- Auto renewal of assets

Renewing Assets Manually

CPQ allows you to renew an existing asset anytime you want, based on your business requirements.

To renew an asset

New Sale

1. From an account, create an opportunity.
2. Create a quote from the opportunity and configure products.
3. Add products to the cart.
   If the administrator has configured the **Renewal Adjustment Type** and **Renewal Adjustment Amount** columns for the cart of the current flow, you will see those columns.
   a. In the **Renewal Adjustment Type** drop-down for a line item, select *% Uplift* by how much you want to uplift the asset price during renewal. In this case, enter the required percentage in the **Renewal Adjustment Amount** field of the line item.
   b. In the **Renewal Adjustment Type** drop-down for a line item, select *Uplift Amount* by how much you want to uplift the asset price during renewal. In this case, enter the required uplift amount in the **Renewal Adjustment Amount** field of the line item.

   > CPQ considers these renewal adjustments only during asset renewal. These values do not impact the current transaction.
4. Finalize the cart.
5. After your customer accepts the quote, an order is generated.
6. After you activate the order, assets are generated. The assets are active and visible on the account.

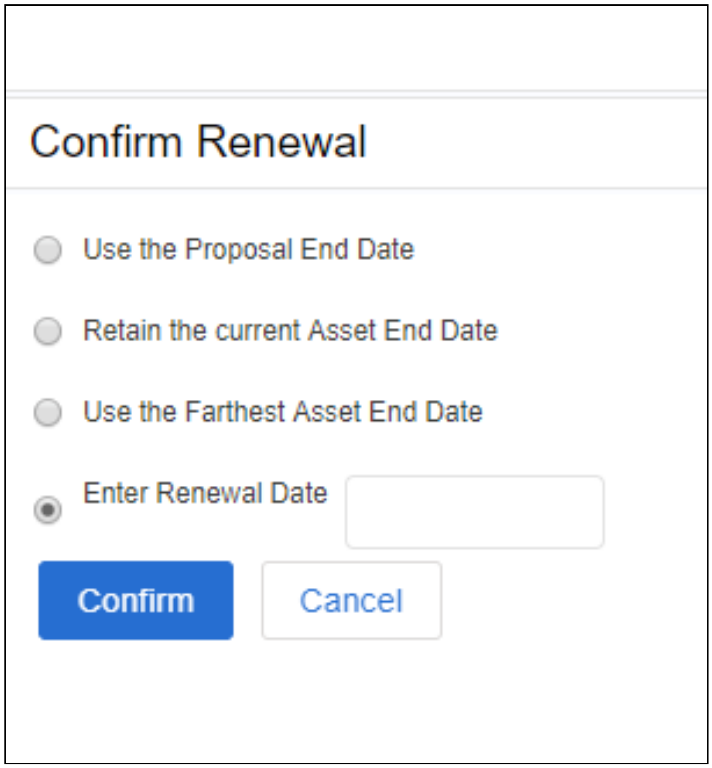
Renewal

1. Create a quote to renew assets.
2. Navigate to the Installed Products page and select the assets that you want to renew.
3. Click **Renew**.

   > If an asset is associated with a system-generated renewal quote and the **Alert Asset Related To Renewal Cart** setting is enabled, a warning is displayed about the existence of a renewal quote for this asset. The **Renewal Line Item** field on the asset indicates which renewal configuration is this asset currently associated with. Click **Proceed without delinking** to continue with manual renewal or **Cancel** to go to the Installed Products page for further action.

If the asset is not associated with any system-generated renewal quote or if the **Alert Asset Related To Renewal Cart** setting is disabled, CPQ either reloads the Installed Products page or displays the Confirm Renewal pop-up or the Confirm Renewal intermediate page depending on how your administrator has configured the **Cotermination Preferences During Renewal** and **Default Renewal Cotermination Option** settings. For more information, see Configuring Custom Settings.

4. Perform one of the following actions:
   - If the Installed Products page is reloaded, click **Go to Pricing**. The cart page is displayed.
   - If the Confirm Renewal intermediate page is displayed, select one from the following options to define the renewal date and click **Confirm**. The cart page is displayed.



   - **Use the Proposal End Date:** To renew the asset until the date Quote/ Proposal ends. The Proposal End Date is displayed next to the option.

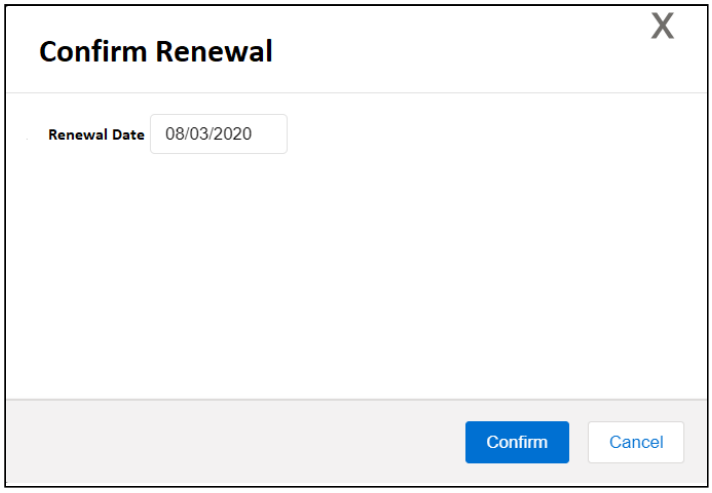
     ⓘ If you perform asset renewal through the Asset Manager flow or CSR flow, the **Use the Proposal End Date** option is not displayed on the Confirm Renewal intermediate page.

- **Retain the current Asset End Date:** To renew the current asset on the date it expires.
- **Use the Farthest Asset End Date:** To choose the farthest end date of all assets for renewal.

  > This option is visible only if the asset has more than one renewable charge types or if the selected asset has multiple renewable lines with different end dates.

- **Enter Renewal Date:** To select a custom renewal date.

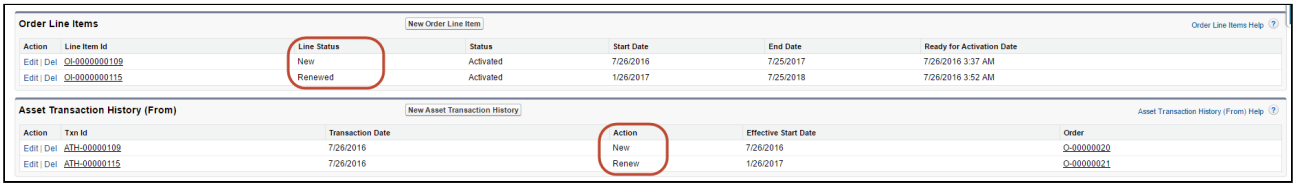
- If the Confirm Renewal pop-up is displayed, select a date in the **Renewal Date** field and click **Confirm**. The cart page is displayed.



5. On the cart, make changes to the asset if required.
   While finalizing the original Quote/Proposal, if you have specified the renewal adjustment (%Uplift or Uplift Amount), you will see the change in the Net Price. Click **Confirm**.

   > If the renewal is loaded with asset pricing and if the renewal adjustment was specified during the original sale, the base price on the renewal transaction reflects the adjustment. If there is not asset pricing, the base price is the new list price of the asset.

6. Click **Reprice** to apply and load the asset with the changes on the cart page.
7. Click **Finalize**.
8. After your customer accepts the quote, an order is generated.
9. After you activate the order, renewal assets are generated. On the Accounts page, you can verify that the status of the renewed asset changes from *New* to *Renewed*. You can trace the renewal activity and modifications from the **Order Line Items** and **Asset Transaction History** Related Lists.



> **Line Status** for one-time products will remain *Existing*.

Check out how renewal works with Bundle and Standalone Assets.

Use Case: Different Renewal Dates

This section provides use cases for different renewal dates. When you are renewing an asset, CPQ provides the following renewal date options and the asset will be renewed until the date you select:

- Using Proposal End Date
- Retaining the Asset End Date
- Using the Farthest End Date
- Entering a Renewal Date

> **Note**
>
> For renewal transactions, **Asset Start Date** will always be the current **Asset End Date + 1**. The **Asset End Date** is derived from the selected renewal options.

Let us take an example to understand how an asset is renewed based on your selection of the Renewal options.

For an Account 'TierOne', you have created a proposal 'W3Courses' which has a Start Date '01/01/2016' and End Date '12/31/2017'. The proposal contains the following Assets:

Asset Name	Asset Status	Start Date	End Date	Selling Term
Programming with Python course	Activated	01/01/2016	06/30/2016	6 months
JAVA Learning	Activated	01/01/2016	12/31/2016	12 months
CSS Learning	Activated	01/01/2016	12/31/2017	24 months

Your subscriber wants to renew the subscription for the 'Programming with Python course' asset. Let us see how renewal works with each of the options:

Using Proposal End Date

This option fetches the Proposal End Date from an active proposal which the selected asset is a part of. The Proposal End Date in this example is '12/31/2017' and so the asset will be renewed till this date. Since Asset Start Date will be the current Asset End Date + 1, in this case the Asset Start Date becomes '07/01/2016'.

Asset Name	Asset Status	Start Date	End Date	Selling Term
Programming with Python course	Activated	07/01/2016	12/31/2017	17 months

The asset is renewed for 17 months from a revised date of start. The 'Action' field on Asset Transaction History is set to Renew.

Retaining the Asset End Date

The most common practice is to renew an asset immediately after it expires. This option renews an asset with the same term from its current End Date. For example, if the selling term for an asset is 24 months (2 years), the asset will be renewed for 2 further years after the asset expires.
In this scenario, the Asset is renewed for another 6 months (derived from the Selling Term) from its Start Date.

Asset Name	Asset Status	Start Date	End Date	Selling Term
Programming with Python course	Activated	07/01/2016	12/31/2016	6 months

Farthest End Date

An account will have multiple proposals and assets with varying Start and End Dates and Term. This option picks the farthest end date of an Asset out of all Assets in the Account and sets this date as the End Date of the renewed Asset.

ⓘ The **Farthest End Date** option is visible only:
- If the asset has more than one renewable charge type

- If the selected asset has multiple renewable lines with different end dates

Let us assume that the farthest End Date is '12/31/2018' for an asset in this Account. Now your asset will be renewed as:

Asset Name	Asset Status	Start Date	End Date	Selling Term
Programming with Python course	Activated	07/01/2016	12/31/2018	29 months

Renewal Date

Instead of pulling a renewal date from various sources, if you want to explicitly specify a Renewal End Date you can do so by selecting this option. If you enter '01/01/2018' as the renewal date, the Asset will be renewed as

Asset Name	Asset Status	Start Date	End Date	Selling Term
Programming with Python course	Activated	07/01/2016	01/01/2018	18 months

Please ensure the Renewal Date entered here is greater than the Asset End Date.

Renewing Assets Automatically

CPQ allows you to auto-generate renewal quotes based on the fulfillment of order or the expiration of subscription. Through renewal quotes, you can forecast and analyze the sales pipeline. With renewal quotes, you get the finer visibility into the pricing and configuration of assets within the renewal pipeline. The benefits of auto renewals are:

- Accurate forecasting on the asset state and sales pipeline
- Faster and simpler processing with automated renewal
- Automatic closure of opportunity on asset expiry

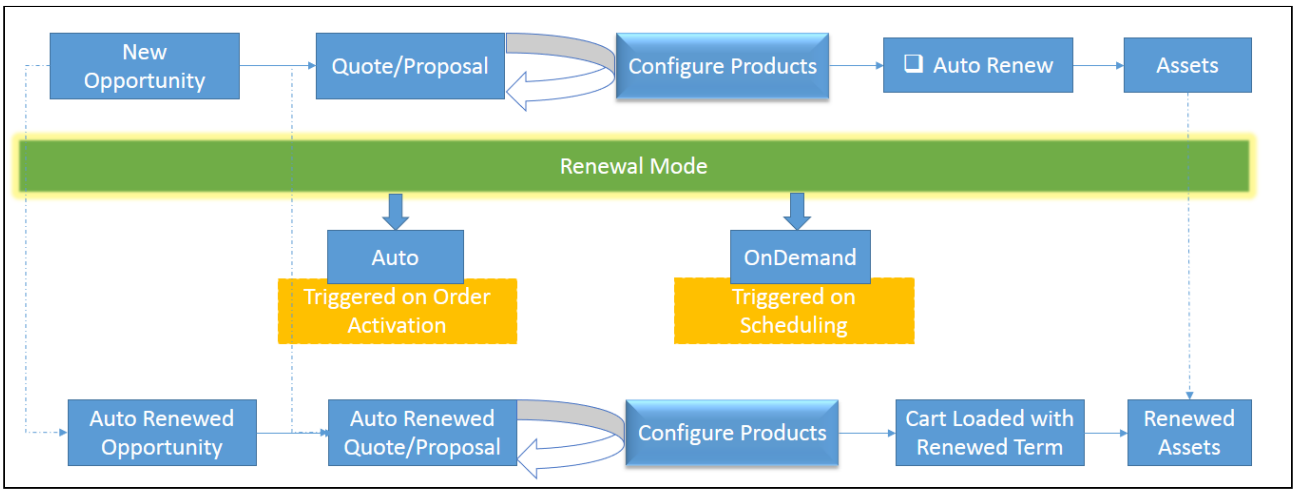
Types of Asset Renewal Modes

There are two types of renewal modes, defined in **Renewal Execution Mode**:

- Auto
- OnDemand

> ⓘ It is recommended not to use these renewal modes interchangeably.

The following diagram explains how auto renewals work:

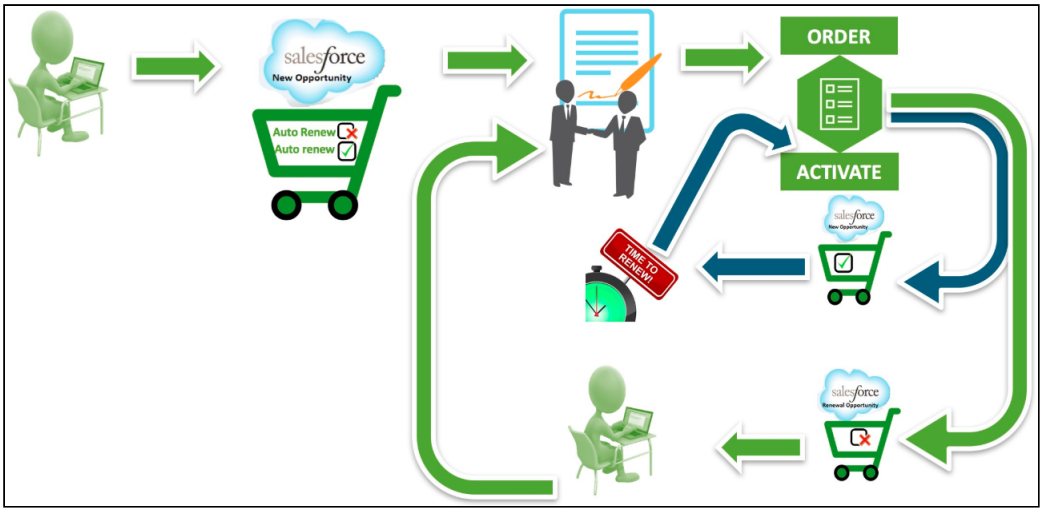


Renewing Assets in Auto Renewal Mode

When **Renewal Execution Mode** is set to *Auto*, CPQ automatically renews an existing quote for all assets it has, as soon as the order is activated. You may want to renew an existing opportunity in Auto renewal mode when:

- You require renewal quotes for forecasting
- You must work on renewal quotes in advance

The following diagram describes the flow of **Auto** renewals.



Prerequisites

- The **Renewal Business Object Type** is set to Proposal.

- The **Renewal Execution Mode** is set to *Auto*.
- The **Renewal Group Fields** are defined.
- The AssetRenewalJobScheduler batch job is scheduled.

How Auto Renewal Mode Works

1. From an account, create an opportunity.
2. Create a quote from the opportunity and configure products.
3. Add products to the cart and finalize it.
4. After your customer accepts the quote, an order is generated.
5. After you activate the order, assets are generated. The assets are active and visible on the account.
6. CPQ creates auto renewal quotes immediately after you activate the order.

   By default, CPQ groups asset line items into renewal quotes by the **Auto Renew** flag and other **Renewal Group Fields** defined. There will be two renewed quotes: one for the asset lines with Auto Renew = True and the other for asset lines with Auto Renew = False.

   > ⓘ Renewal quote that has assets with Auto Renew = True: On the expiry date of the asset, CPQ automatically processes the renewal quote and renews the asset for the next term. The quote will be automatically set to Accepted without any intervention (this is called Touchless Renewal).
   > Renewal quote that has assets with Auto Renew = False: You need to manually process the quote.

   On the Account page, the renewed asset groups are listed under Temp Renew Asset Groups in the New status. After the AssetRenewalJobScheduler job is executed, the status of asset groups changes to Completed and the renewal quote is created. The name of the renewal quote will be Renew:<original_quote_id>-<name_of_price_list>-<quote_end_date>.

   When the renewal quote is generated, the **Renewal Line Item** field on the asset line item is updated with the latest renewal configuration ID on the renewal quote. When the user tries to manually renew this asset, a warning is displayed about the existence of a renewal quote for this asset.

7. When you configure products in this renewal quote, CPQ takes you to the Cart page directly where assets with renewed Start Date, End Date, and Selling Term are

available. The Line Status of assets is Renewed and asset pricing calculated according to the renewed Start Date, End Date, and Selling Term.

> On the new renewal quote, the **Start Date** = **Parent Quote's End Date** + 1 and **End Date** = **New Start Date** + Selling Term.

The Approval Stage of the quote is marked back to Draft allowing you to perform any changes to the renewal configuration. Any change or update to the assets are reflected in the opportunity based renewal quote.

8. Add products to the cart if required and finalize it.
9. After your customer accepts the quote, an order is generated.

Use Case: Grouping of Assets without Renewal Group Fields During Renewal

In this example, CPQ renews a quote which has both auto-renew and non-auto renew lines. In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	One-time	10000	4/1/2015	3/31/2016	10	TRUE
Product A	Recurring	100	1/1/2015	12/31/2015		FALSE
Product B	One-time	5000	5/1/2015	4/30/2016	10	FALSE
Product B	Recurring	200	1/1/2015	12/31/2015		TRUE

Configuration

Field	
Renewal Execution Mode	Auto
Renewal Lead Time	-
Renewal Group Fields	-

Quotes Created

All non-auto renew

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	Recurring	100	1/1/2015	12/31/2015		FALSE
Product B	One-time	5000	5/1/2015	4/30/2016	10	FALSE

All auto renew

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	One-time	10000	4/1/2015	3/31/2016	10	TRUE
Product B	Recurring	200	1/1/2015	12/31/2015		TRUE

Use Case: Grouping of Assets with Renewal Group Fields During Renewal

You can use any field on the Asset Line Item in **Renewal Group Fields** when creating a renewal quote. This section explains how CPQ groups renewal assets with use cases.

Grouping a Renewal Quote by Product

In this example, the original quote has both auto-renew and non-auto renew lines. In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	One-time	10000	4/1/2015	3/31/2016	10	TRUE
Product A	Recurring	100	1/1/2015	12/31/2015		FALSE
Product B	One-time	5000	5/1/2015	4/30/2016	10	FALSE
Product B	Recurring	200	1/1/2015	12/31/2015		TRUE

Configuration

To group the renewal assets by the Product, enter the *API name* of the **Product Name** field in **Renewal Group Fields**.

Field	
Renewal Execution Mode	Auto
Renewal Lead Time	-
Renewal Group Fields	Apttus_Config2__ProductId__r.Name

Renewal Quotes Created

Quote 1

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	One-time	10000	4/1/2015	3/31/2016	10	TRUE
Product A	Recurring	100	1/1/2015	12/31/2015		FALSE

Quote 2

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product B	One-time	5000	5/1/2015	4/30/2016	10	FALSE
Product B	Recurring	200	1/1/2015	12/31/2015		TRUE

Grouping Renewal Quotes by Product and Auto Renew Flag

This is the quote that will undergo renewal. In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	One-time	10000	4/1/2015	3/31/2016	10	TRUE
Product A	Recurring	100	1/1/2015	12/31/2015		FALSE
Product B	One-time	5000	5/1/2015	4/30/2016	10	FALSE
Product B	Recurring	200	1/1/2015	12/31/2015		TRUE

Configuration

To group the Renewal Quotes by the Product and Auto Renew Status, enter the *API name* of these fields in **Renewal Group Fields**.

Field	
Renewal Execution Mode	Auto
Renewal Lead Time	-
Renewal Group Fields	Apttus_Config2__ProductId__r.Name, Apttus_Config2__AutoRenew

Quotes Created

Quote 1 Product A: auto renew

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	One-time	10000	4/1/2015	3/31/2016	10	TRUE

Quote 2 Product B: auto renew

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product B	Recurring	200	1/1/2015	12/31/2015		TRUE

Quote 3 Product A: non-auto renew

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	Recurring	100	1/1/2015	12/31/2015		FALSE

Quote 4 Product B: non-auto renew

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product B	One-time	5000	5/1/2015	4/30/2016	10	FALSE

Renewing Assets in OnDemand Renewal Mode

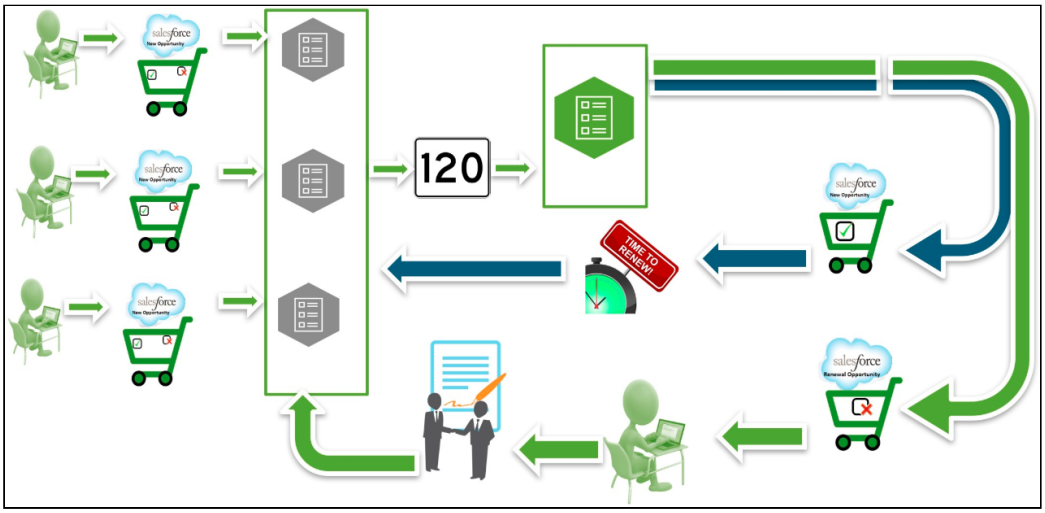
When **Renewal Execution Mode** is set to *OnDemand*, CPQ automatically renews assets based on the defined **Renewal Lead Time**. CPQ supports Account mode and Product mode for OnDemand renewal of assets. However, these modes are mutually exclusive, which means you must use only one mode in one org. You may want to renew an existing asset in OnDemand renewal mode:

- During one-time legacy asset data migrations
- During one-time renewal quote creation for existing active asset base
- When minimal changes are required on the renewal quotes because of any ABO changes

Account Mode

This mode works based on the global **Renewal Lead Time** setting in Installed Product Settings. CPQ automatically renews all assets on the account (possibly from different quotes) based on the defined global **Renewal Lead Time** after you click a **custom button** on the account.

The following diagram describes the flow of **OnDemand** renewals. In this diagram, **Renewal Lead Time** is 120 days.



Over a period of time, you sell some products/services through different quotes, which might have varying end dates. After the original quotes are converted into orders and the order are activated, assets are generated. All assets are visible on the account. When you want to renew the assets that will expire in 120 days from today, you click the **custom button** on the account. CPQ compares the asset line items' end dates with the **Renewal Lead Time** defined and triggers the creation of renewal quotes for all those assets (from different quotes) that will expire in 120 days.

Prerequisites

- The **Renewal Business Object Type** is set to Proposal.
- The **Renewal Execution Mode** is set to OnDemand.
- A custom button is created on the account or a custom link is created on the quote.

  > If the custom button is on the account, you can renew all assets in the account based on the defined Renewal Lead Time. If the custom link on the quote, you can renew assets from a quote based on the defined Renewal Lead Time.

- The **Renewal Lead Time** is defined.
- The **Renewal Group Fields** are defined.
- The *AssetRenewalJobScheduler* batch job is scheduled.

How OnDemand Renewal Mode Works

1. From an account, create an opportunity.
2. Create quotes from the opportunity and configure products.
3. Add products to the cart and finalize it.
4. After your customer accepts the quotes, orders are generated.
5. After you activate the orders, assets are generated. The assets are active and visible on the account.
6. Click the **custom button** on the account when you want to trigger the creation of renewal quotes.
7. CPQ creates renewal quotes based on the defined **Renewal Lead Time**, considering the expiry date of assets (from different quotes).

   By default, CPQ groups asset line items by Account and the Price List. However, if other parameters are configured in **Renewal Group Fields**, CPQ groups asset line items accordingly. If some assets had **Auto Renew** flag set to True or False, there will be two renewed quotes: one for the asset lines with Auto Renew = True and the other for asset lines with Auto Renew = False.

> Renewal quote that has assets with Auto Renew = True: On the expiry date of the asset, CPQ automatically processes the renewal quote and renews the asset for the next term. The quote will be automatically set to Accepted without any intervention (this is called Touchless Renewal).
> Renewal quote that has assets with Auto Renew = False: You need to manually process the quote.

After CPQ triggers renewal quote creation, on the Account page, the renewed asset groups are listed under Temp Renew Asset Groups in the New status. After the *AssetRenewalJobScheduler* job is executed, the status of asset groups changes to Completed and the renewal quote is created. The name of the renewal quote will be Renew:<name_of_price_list>-<quote_end_date>.

When the renewal quote is generated, the **Renewal Line Item** field on the asset line item is updated with the latest renewal configuration ID on the renewal quote. When the user tries to manually renew this asset, a warning is displayed about the existence of a renewal quote for this asset.

When you execute the *CreateRenewAssetsGroupJob*, CPQ creates the required TempRenewAssetGroups on active asset line items. After the *AssetRenewalJobScheduler* job is executed, CPQ processes those TempRenewAssetGroups, changes the status of asset groups to Completed, and creates the renewal quotes and configurations. After you perform some changes on the assets, if you execute *CreateRenewAssetsGroupJob* again, CPQ does not create any new TempRenewAssetGroups.

8. When you configure products in this renewal quote, CPQ takes you to the Cart page directly where assets with renewed Start Date, End Date, and Selling Term are available. The Line Status of assets is Renewed and asset pricing calculated according to the renewed Start Date, End Date, and Selling Term.

   > On the new renewal quote, the **Start Date** = **Parent Quote's End Date** + 1 and **End Date** = **New Start Date** + Selling Term.

   The Approval Stage of the quote is marked back to Draft allowing you to perform any changes to the renewal configuration. Any change or update to the assets are reflected in the opportunity based renewal quote.
9. Add products to the cart if required and finalize it.
10. After your customer accepts the quote, an order is generated.

Product Mode

This mode works based on the product-level **Renewal Lead Time**. CPQ automatically renews all assets (possibly from different quotes across all accounts in the org) based on the defined product-level **Renewal Lead Time** after you execute a code in Developer Console.

Prerequisites

- The **Renewal Business Object Type** is set to Proposal.
- The **Renewal Execution Mode** is set to OnDemand.
- The **Renewal Lead Time** is defined for the required products. See Creating Products. For example, there are two products with the Renewal Lead Time of 20 and 40 days.
- The **Renewal Group Fields** are defined.
- The *AssetRenewalJobScheduler* batch job is scheduled.

How OnDemand Renewal Mode Works

1. Create quotes and configure products with Renewal Lead Time.
2. Add products to the cart and finalize it.
3. After your customer accepts the quotes, orders are generated.
4. After you activate the orders, assets are generated.
5. Run the following code in Developer Console:

   ⚠ In the following code, (20,5,100,30} is used as an example Renewal Lead Time. CPQ allows you to use multiple lead times separated by a comma.

   ```
   List<Integer> productLeadTimes = new List<Integer>

   {20,5,100,30}
   ;

   Apttus_Config2.AssetRenewalSubmitController controller = new
    Apttus_Config2.AssetRenewalSubmitController(productLeadTimes);

   // do submit
   ID jobId = controller.doSubmitJob();
   ```

6. CPQ creates renewal quotes based on the defined **Renewal Lead Time**, considering the expiry date of assets (from different quotes across all accounts in the org). For example, if there are two products with the Renewal Lead Time of 20 and 40 days, only the product with Renewal Lead Time=20 is triggered and its OnDemand renewal quote is created.

By default, CPQ groups asset line items by Account and the Price List. However, if other parameters are configured in **Renewal Group Fields**, CPQ groups asset line items accordingly. If some assets had **Auto Renew** flag set to True or False, there will be two renewed quotes: one for the asset lines with Auto Renew = True and the other for asset lines with Auto Renew = False.

> ⓘ Renewal quote that has assets with Auto Renew = True: On the expiry date of the asset, CPQ automatically processes the renewal quote and renews the asset for the next term. The quote will be automatically set to Accepted without any intervention (this is called Touchless Renewal).
> Renewal quote that has assets with Auto Renew = False: You need to manually process the quote.

After CPQ triggers renewal quote creation, on the respective Account page, the renewed asset groups are listed under Temp Renew Asset Groups in the New status. After the *AssetRenewalJobScheduler* job is executed, the status of asset groups changes to Completed and the renewal quote is created. The name of the renewal quote will be Renew:<name_of_price_list>-<quote_end_date>.

When the renewal quote is generated, the **Renewal Line Item** field on the asset line item is updated with the latest renewal configuration ID on the renewal quote. When the user tries to manually renew this asset, a warning is displayed about the existence of a renewal quote for this asset.

When you execute the *CreateRenewAssetsGroupJob*, CPQ creates the required TempRenewAssetGroups on active asset line items. After the *AssetRenewalJobScheduler* job is executed, CPQ processes those TempRenewAssetGroups, changes the status of asset groups to Completed, and creates the renewal quotes and configurations. After you perform some changes on the assets, if you execute *CreateRenewAssetsGroupJob* again, CPQ does not create any new TempRenewAssetGroups.

7. When you configure products in this renewal quote, CPQ takes you to the Cart page directly where assets with renewed Start Date, End Date, and Selling Term are available. The Line Status of assets is Renewed and asset pricing calculated according to the renewed Start Date, End Date, and Selling Term.

> On the new renewal quote, the **Start Date** = **Parent Quote's End Date** + 1 and **End Date** = **New Start Date** + Selling Term.

The Approval Stage of the quote is marked back to Draft allowing you to perform any changes to the renewal configuration. Any change or update to the assets are reflected in the opportunity based renewal quote.

8. Add products to the cart if required and finalize it.
9. After your customer accepts the quote, an order is generated.

Use Case: Renewals without Grouping

In this example, CPQ renews a quote which has both auto-renew and non-auto renew lines. In this example, Quantity = 1 and Auto Renewal Term = 24.

Product	Charge Type	Base Price	Term	Start Date	End Date	Net Price	Renewal Adjmt Amount	Renewal Adjmt Type	Auto Renew
Product A	One-time	10000	12	4/1/2015	3/31/2016	10000	10	percent	TRUE
Product A	Recurring	100	12	1/1/2015	12/31/2015	10000		percent	FALSE
Product B	One-time	5000	12	5/1/2015	4/30/2016	10000	10	percent	FALSE
Product B	Recurring	200	12	1/1/2015	12/31/2015	10000		percent	TRUE
Product C	One-time			7/1/2015	6/30/2016	10000			TRUE
Product C	Recurring			7/1/2015	6/30/2016	4	10	percent	TRUE

Configuration

Field	
Renewal Execution Mode	OnDemand

Field	
Renewal Lead Time	120
Renewal Group Fields	-

Quotes Created

Non-auto renew (all assets expiring within 120 days from 1st Dec 2016)

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	Recurring	100	1/1/2015	12/31/2015		FALSE
Product B	One-time	5000	5/1/2015	4/30/2016	10	FALSE

Auto Renew (note that Product C is not in the list)

In this example, Term = 12, Quantity = 1, Net Price = 10000, Auto Renewal Term = 24, and Renewal Adjustment Type = percent.

Product	Charge Type	Base Price	Start Date	End Date	Renewal Adjustment Amount	Auto Renew
Product A	One-time	10000	4/1/2015	3/31/2016	10	TRUE
Product B	Recurring	200	1/1/2015	12/31/2015		TRUE

Use Case: Renewing Bundle Assets

This section describes the renewal process for a bundle asset. The Asset 'ABO Budle1_AS' with 3 different charge types has 2 Options- Option1 and Option2. On the cart, Option1 is selected with 6 months and 3 months of validity for corresponding charge types.

The cart is finalized and an order is generated. You can see the Asset Line Items on the Quote/Proposal or from the Account.

Let us see how this Asset is renewed based on the Renewal Option of **Retaining the Asset End Date**.

Deriving Start and End Dates

For renewal transactions, **Asset Start Date** will always be the **Current Asset End Date + 1**.

In this renewal option, **Asset End Date** will be the date derived from adding the **Selling Term** to the new **Asset Start Date**.

Before Renewal

Asset	Start Date	End Date
ABO Bundle1_AS	07/26/2016	07/25/2017
Option1	07/26/2016	01/25/2017
Option1 (Different Charge Type)	07/26/2016	10/25/2016

The **Asset Status** on the Installed Products page is **Activated**.

After Renewal

Asset	Start Date	End Date
ABO Bundle1_AS	07/26/2017	07/25/2018
Option1	01/26/2017	07/25/2017
Option1 (Different Charge Type)	10/26/2016	01/25/2017

The **Asset Status** on the Installed Products page is *Pending Renewal*. When you confirm and go to the Cart page, the **Status** changes to *Renewed*.

Use Case: Renewing Standalone Assets

This section describes the renewal process for a standalone asset. The Asset 'ABO Standalone1_AS' has 4 different charge types.

The cart is finalized and an order is generated. You can see the Asset Line Items on the Quote/Proposal or from the Account.

Let us see how this Asset is renewed based on the Renewal Option of **Retaining the Asset End Date**.

Deriving Start and End Dates

For renewal transactions, **Asset Start Date** will always be current **Asset End Date** + **1**. In this renewal option, **Asset End Date** will be the date derived from adding the Selling Term again.

Before Renewal

Asset	Start Date	End Date
ABO Standalone1_AS	07/26/2016	07/25/2017

After Renewal

Asset	Start Date	End Date
ABO Standalone1_AS	07/26/2017	07/25/2018

The **Asset Status** on the Installed Products page is **Pending Renewal**. When you confirm and go to the Cart page, the Status changes to **Renewed**.

Use Case: Renewing Merged Assets with the Same Purchase Identifier

When the user merges two assets with the same purchase identifier, the merged line is associated with the same renewal quote that contained the original lines. For more information on merging assets, see Merging Assets.

Sale 1

Asset	Purchase ID	Renewal Quote
Asset 1	A123	RQ1

Sale 2

Asset 2 is also associated with the same renewal quote because the purchase identifier is same.

Asset	Purchase ID	Renewal Quote
Asset 2	A123	RQ1

Merge Assets

When Asset 1 and Asset 2 are merged, CPQ creates merged Asset 3. Because the merged asset also has the same purchase identifier, it will be created in the same renewal quote that had original assets.

Asset	Purchase ID	Renewal Quote
Asset 3	A123	RQ1

The cancelled lines will be removed from the original quote.

Use Case: Renewed Ramped Assets Start from the End Date of the Last Ramp Line

CPQ begins the renewal start date of a ramped bundle asset (multi-year deal) based on the end date of the last ramp line (in the current term). CPQ calculates the renewal start date of the second ramp line based on the end date of the first ramp line; CPQ calculates the renewal start date of the subsequent ramp lines based on the end date of the previous ramp lines. While calculating the renewal start date of ramped asset line items, CPQ aligns the selling term of all renewal ramp lines with the selling term of ramp lines from the new sale.

Key Points to Note:

- CPQ supports both manual and auto renewal of assets.
- You must use only the **Retain with current Asset End Date** option on the Confirm Renewal pop-up. This use case does not work with other renewal end date options.
- The **Renew One Ramp** setting must be *False*.

Use Case 1

In this use case, you renew a bundle after changing the end date of the last ramp line.

New Sale: You sell a bundle to a customer during new sale.

Ramp	Start Date	End Date	Selling Term
1	1/1/2021	12/31/2021	1
2	1/1/2022	12/31/2022	1
3	1/1/2023	12/31/2023	1

Change Asset: You change the end date of the last ramp line from 12/31/2023 to 6/30/2023.

Ramp	Start Date	End Date	Selling Term
1	1/1/2021	12/31/2021	1
2	1/1/2022	12/31/2022	1
3	1/1/2023	6/30/2023	0.5

Renew Asset: After you renew the bundle asset, the renewal start date is based on the end date of the last ramp line. Though you change the end date of the last ramp line (resulting in different selling term from the previous two ramp lines), CPQ still honours the changed date and starts the renewal of the asset from the end date of the last ramp line.

Ramp	Start Date	End Date	Selling Term
1	7/1/2023	6/30/2024	1
2	7/1/2024	6/30/2025	1
3	7/1/2025	12/31/2025	0.5

Use Case 2

In this use case, you renew a bundle after changing the end date of multiple ramp lines.

New Sale: You sell a bundle to a customer during new sale.

Ramp	Start Date	End Date	Selling Term
1	1/1/2021	12/31/2021	1
2	1/1/2022	12/31/2022	1
3	1/1/2023	12/31/2023	1

Change Asset: You change the end date of multiple ramp lines.

Ramp	Start Date	End Date	Selling Term
1	1/1/2021	12/31/2022	2
2	1/1/2023	12/31/2023	1
3	1/1/2024	6/30/2024	0.5

Renew Asset: After you renew the bundle asset, the renewal start date is based on the end date of the last ramp line. While calculating the renewal start date of a ramped asset line item, CPQ aligns the selling term of all renewal ramp lines with the selling term of ramp lines from the new sale. CPQ calculates the renewal start date of the second and third ramp lines based on the end date of the first and second ramp lines respectively. Though you change the end date of multiple ramp lines (resulting in different selling terms), CPQ still honours the changed dates and starts the renewal of the asset from the end date of the last ramp line.

Ramp	Start Date	End Date	Selling Term
1	7/1/2024	6/30/2026	2
2	7/1/2026	6/30/2027	1
3	7/1/2027	12/31/2027	0.5

Renewing Assets Using Current Contract Value

> The Contract Term feature is supported only through Proposal.

As a sales representative for a company that sells subscriptions, you may have to uplift and discount the subscription multiple times during its life cycle. Because of such change operations, the net price of the subscription changes, but the contract price remains the renewed value so there is a difference in the asset net unit price and contract net unit price. Due to such operations, sometimes you get higher asset net unit price and lower contract price, or lower asset net unit price and higher contract price. You can renew the subscription for the total contract value or its annual contract value depending on the business case. CPQ loads the asset line with base price as defined by the **Base Price Defaulting Method For Renewal** setting. For more information, see Configuring Custom Settings.

You can provide discount on the subscription for a value greater than the current net price of the subscription (post a renewal and upgrade operation). CPQ can calculate the discount from the start date of the current contract term (not the original start date of the asset), if the **Apply Adj To Current Contract Term** setting is enabled. If **Apply Adj To Current Contract Term** is False, after deep discount, both asset and contract term will start from the original start date. If **Apply Adj To Current Contract Term** is False and if the customer has a deep discounting use case (discount an amount > net price but < asset TCV), the recommended best practice is to not expose the contract fields (Current Contract Start Date, Current Contract Term, Current Contract Value, Current Contract Unit Price) on the termination UI. These fields will reset to the original term after a deep discount and the values will be repetitive. **Apply Adj To Current Contract Term** is a global flag. The value of this flag must be either TRUE or FALSE throughout the cycle of asset. You should not switch this flag during asset operations.

> CPQ does not apply delta price adjustment on a subscription if the bundle price becomes the price from PLI after you change an attribute.
> Delta price adjustment must be applied only on bundle line.
> Delta price adjustment is not applied to a one-time bundle line (new or existing) even though the **Enable One Time Change** is set to True.

Use Case: Early Renewal of Ramped Bundle Assets with Options

You can perform an early renewal of a bundle asset that has options and multiple ramps, by changing the bundle start date to a date earlier than current asset end date.

Key Points to Note:

- CPQ supports both manual and auto renewal of assets.
- The **Auto Cascade Selling Term** setting must be *True* on the price list item of the options.
- The **Enable Auto Ramp Creation** setting must be *True* on the price list item of the bundle and options.

New Sale

In this use case, you purchase 1 bundle and 1 option with three ramp lines during the new sale.

Product Type	Start Date	End Date	Selling Term	Line Status
Bundle	01/01/2021	12/31/2021	1	New
Bundle	01/01/2022	12/31/2022	1	New
Bundle	01/01/2023	12/31/2023	1	New
Option	01/01/2021	12/31/2021	1	New
Option	01/01/2022	12/31/2022	1	New
Option	01/01/2023	12/31/2023	1	New

Assets are created for the ramped bundle and option.

During Renewal

During renewal, you see the assets with the following details.

Asset Type	Start Date	End Date	Selling Term	Line Status
Bundle	01/01/2024	12/31/2024	1	Renewed
Bundle	01/01/2025	12/31/2025	1	Renewed
Bundle	01/01/2026	12/31/2026	1	Renewed
Option	01/01/2024	12/31/2024	1	Renewed
Option	01/01/2025	12/31/2025	1	Renewed
Option	01/01/2026	12/31/2026	1	Renewed

Renewal with a Modified Start Date

After 18 months, you want to renew this ramped bundle with its option for next 3 years under a good deal. You want to renew the service earlier then current asset end date (12/31/2023). You renew the assets by changing the start date of the first renewal bundle asset to 6/30/2022.

Asset Type	Start Date	End Date	Selling Term	Line Status
Bundle	06/30/2022	12/31/2024	2.5	Renewed
Bundle	01/01/2025	12/31/2025	1	Renewed
Bundle	01/01/2026	12/31/2026	1	Renewed
Option	06/30/2022	12/31/2024	2.5	Renewed
Option	01/01/2025	12/31/2025	1	Renewed
Option	01/01/2026	12/31/2026	1	Renewed

Key Points to Note:

- When you change the start date of the primary ramp line on the bundle, CPQ changes the start date of the primary option line to the same date as the bundle.

- CPQ recalculates the selling term of the primary ramp line on the bundle and option based on the changed date.
- CPQ recalculates the pricing correctly, based on the changed date.
- CPQ retains the line status of bundle and option lines as *Renewed*.

Changing an Asset

With the Change function, you can increase or decrease the quantity of assets according to your customer's requests. Change operation also allows you to change the Start Date, End Date, Selling Term, Billing Rule and other fields on the cart for both, Standalone and Bundle assets. You can update the value of several fields on your cart under a single Change action, for a given Asset.

Of the various parameters you can change, the following are the most commonly updated fields:

- **Change Quantity:** If you update the quantity of an asset, this update will be reflected as an amendment in the current Asset Line Item. A new line item will not be created for this change.
  You should specify an absolute value for the **Quantity** field regardless of an increment or decrement in the actual quantity. For example, if you want to increase the asset quantity from 100 to 120, you should enter 120 and not 20. Therefore you must enter the total quantity you want as a result of the change action.

- **Change Start Date:** Updating Start Dates indicates the date from which the change is effective. If you update both the Quantity and the Start Date, the updated quantity will come into effect from the given start date.

  For example, an Asset with 100 Licenses has a **Start Date** of 01 January 2016 and **End Date** of 31 December 2016. As part of the **Change** action, Start Date is set to 01 March 2016 and Quantity is increased by 30. This means, from 01 January to 29 February customer has 100 Licences and starting from 01 March till the End Date, the number of Licenses is 130.
  Please note that prior to the new date, the asset will remain in its previous state and will be billed accordingly. The older information will *not* be deleted or overwritten.

- **Change End Date**: You can update the End Date to extend or shorten the asset validity. On updating the Asset End Date, the existing Line item is amended.

> **Note**
>
> For any **Change** action on the asset, **Net Price** indicates the price of the Asset as seen from the **Start Date** to the **End Date** on the cart.

Let us take a look at the procedure to change an asset.

To change an asset

To change the Quantity and Selling Term for a standalone asset.

1. On the **Installed Products** page, select the asset you want to update and click **Change**. On the cart page, **Line Status** changes to *Amended*.
2. You can change one or more of the following parameters:

Parameter	Description
Attribute Value	To change the Attribute value, click the wrench icon ( ) next to the asset. You will see the **Product Attributes** page. Select the new attribute value and click **Go to Pricing** to load the cart page.
Option	Expand the Show Options tree and select from the listing of the options to display all Product Options. From the resulting list, select the option with which you want to replace the original product option.
Quantity	Enter the number of units of the product your customer wants to purchase.
Adjustment Type	If you want to adjust the price of the asset, select an adjustment type.
Price	Enter a value that is the amount your customer is charged for the product without accounting for taxes and other charges.
Start Date	The date you want to start billing your customer for this purchase.

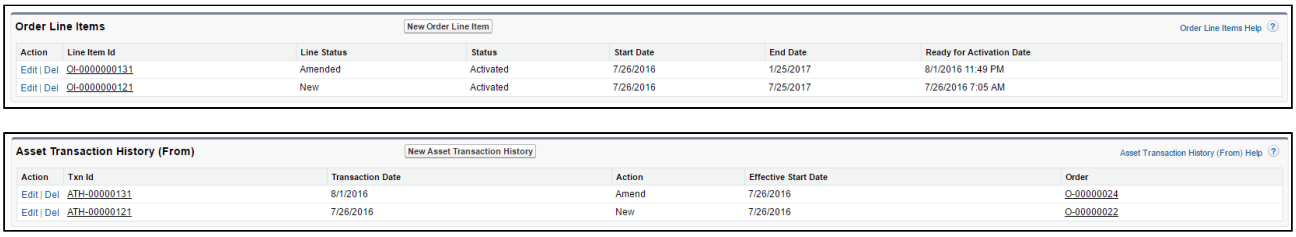
Parameter	Description
End Date	The date you want to stop billing your customer for this purchase.
Billing Frequency	Select one of the following options **Monthly** - To generate a bill once every month **Quarterly** - To generate a bill once every three months **Half-yearly**- To generate a bill once in the middle of a calendar or financial year **Yearly** - To generate a bill once every year **Usage** - To generate a bill based on usage For more information, see Change Billing Frequency.
Billing Rule	Select one of the following options **Bill in Advance** - To bill your customer before the product is delivered **Bill in Arrears** - To bill your customer after the product is delivered **Bill on Ready for Billing Date** - To bill your customer with a consolidate invoice, on a day of their choice.
Billing Preference	Select a predefined billing preference. For details, see Billing Preferences.
Payment Term	Lookup and select a predefined Payment Term so your customers know that they must pay outstanding charges within a stipulated time. For more information, see Payment Term.

3. Click **Reprice** to view the changed price and **Finalize** to go back to the Quote.
4. The status of the changed asset changes from *New* to *Amended*.

You can rename the **Change** button as per your business requirement. The functionality remains the same and only the label is renamed. For more information, see Configuring Display Actions Settings.

Impact of Changing the Asset on Order Line Items and Asset Transaction History

The status of the asset changes to *Amended* on the Order Line Item and a new entry for the **Change** action is added as to the Asset Transaction History as *Amend*.



Order Line Items — New Order Line Item — Order Line Items Help

Action	Line Item Id	Line Status	Status	Start Date	End Date	Ready for Activation Date
Edit | Del	OI-0000000131	Amended	Activated	7/26/2016	1/25/2017	8/1/2016 11:49 PM
Edit | Del	OI-0000000121	New	Activated	7/26/2016	7/25/2017	7/26/2016 7:05 AM

Asset Transaction History (From) — New Asset Transaction History — Asset Transaction History (From) Help

Action	Txn Id	Transaction Date	Action	Effective Start Date	Order
Edit | Del	ATH-00000131	8/1/2016	Amend	7/26/2016	O-00000024
Edit | Del	ATH-00000121	7/26/2016	New	7/26/2016	O-00000022

To check the status of your assets, the Status Category of the proposal must be **Accepted**. To view the assets, select the account of the proposal and navigate to Asset Line Items (Sold To) related list.

Every change you make to an asset impacts its Billing Schedule. If the asset is usage-based, then it impacts the associated Usage Schedule as well.

> **Note**
>
> Even though you can extend your asset's validity using **Change**, it is recommended that you perform **Renewal** action for explicit renewal purposes.

> **Renewing Ramp Lines after Change Operation**
> The Asset Grid retains renew ramp line upon Change or Renew operation after you enable Renew One Ramp and Renew of Option.
> When you click **Change** for an asset on the **Installed Products** page, CPQ shows one renewed option line. However, you must enable `Renew One Ramp` in the Installed Product settings along with Price Ramps for the bundle.

Changing Configurable Bundle Assets

You can change the parameters of a bundle asset, much like you do with a standalone. On the Installed Products page,

1. Select the Bundle asset that you want to change and click **Change**. The cart page is displayed.
2. To view the options, click the **Show Options** list below the product name.
3. To change the options, click the wrench icon ( ) next to the asset.

4. From the list of displayed options, select the checkbox for options you want to include and deselect the checkbox for options you want to remove.
5. Click **Go to Pricing** to load the cart.

> **Note**
>
> For *Change: Configuration*, note that terminating an asset from the original start date results in incorrect **Net Price** for the changed and upgraded asset line items.

6. Enter values manually for the fields described in the Changing an Asset page.
7. Click **Reprice** and then **Finalize**.

You have successfully created a new order with different product options.

Changing the Attributes of an Asset

Similar to a Line Item (For more information, see About Assets and Asset Line Items) in the Quote to Cash flow, that starts as a Cart Line Item, progresses to the Proposal, Agreement, Order, and finally the Asset Line Item, the Attribute values that you define for each Product, are copied to the Proposal Product Attribute values object, Order Product Attribute values object, and the Asset Product Attribute values object. You can view the Product Attribute values that the Asset Line Item inherits from the cart, on the Asset line item detail page.

You can customize the product configuration on your cart based on the Product Attribute values that you define. For more information, see Managing Attribute-Based Configuration.

To change the Attribute values for an asset

1. On the Installed Products page, select an asset and click **Change**. The cart page is displayed.
2. Click the wrench icon ( ) next to the asset.
3. On the Product Attributes section, select the attributes that you want to change and enter their values.
4. Click **Go to Pricing**.

> **Note**
>
> Only those Attributes for which you select the **Is Primary** check box on the Attribute Configuration page, are shown in the Product Attributes section of the Installed Products page. These attributes are called Critical Attributes.

Use Case: Cancelling Options in a Bundle Asset with Ramp Lines

This section describes how CPQ behaves when you cancel an option in a bundle asset that has ramp lines.

Case A

New Sale

1. From an account, create an opportunity.
2. Create a quote from the opportunity and configure products.
3. Add a bundle product with an option (Option-1) to the cart. Make sure that the bundle does not have any ramp lines.
4. Finalize the cart.
5. After your customer accepts the quote, an order is generated.
6. After you activate the order, a bundle asset is generated. The asset is active and visible on the account.

Change Asset

1. Create a quote to change the asset.
2. Navigate to the Installed Products page and select the bundle assets that you want to change.
3. Click **Change**. The cart page is displayed.
4. Cancel the existing Option-1 and add the new Option-2.
5. Add three ramps to the bundle asset.

   ⓘ CPQ does not cascade ramp lines from the bundle to the cancelled Option-1.

6. Click **Go to Pricing** to load the cart.
7. Click **Reprice** to apply and load the asset with the changes on the cart page.
8. Click **Finalize**.
9. After your customer accepts the quote, an order is generated.
10. After you activate the order, change assets are generated.

Case B

New Sale

1. From an account, create an opportunity.

2. Create a quote from the opportunity and configure products.
3. Add a bundle product with an option (Option-1) to the cart. Make sure that the bundle does not have any ramp lines.
4. Finalize the cart.
5. After your customer accepts the quote, an order is generated.
6. After you activate the order, a bundle asset is generated. The asset is active and visible on the account.

Change Asset

1. Create a quote to change the asset.
2. Navigate to the Installed Products page and select the bundle assets that you want to change.
3. Click **Change**. The cart page is displayed.
4. Add the new Option-2.
5. Add three ramps to the bundle asset. On the Cart both Option-1 and Option-2 have three ramp lines. For Option-1, there is one ramp in Existing status and two ramps in New status.
6. Click the wrench icon ( ) next to the bundle asset and cancel the existing Option-1.

   The first line is in Cancelled status and the other two lines are removed. CPQ adjusts the amount on the bundle properly.

7. Click **Go to Pricing** to load the cart.
8. Click **Reprice** to apply and load the asset with the changes on the cart page.
9. Click **Finalize**.
10. After your customer accepts the quote, an order is generated.
11. After you activate the order, change assets are generated.

Use Case: Cancelled Asset Options Fetch Original Asset Start and End Dates During Renewal

While renewing a bundle asset with options, if you cancel an option and go to the cart, the cancelled option line items fetch the original asset start date and end date. Further, these dates are cascaded to the proposal line items, order line items, and asset line items.

New Sale

In this use case, you purchase 1 bundle and 1 option with three ramp lines during the new sale.

Product Type	Start Date	End Date	Selling Term	Line Status
Bundle	01/01/2021	12/31/2021	1	New
Bundle	01/01/2022	12/31/2022	1	New
Bundle	01/01/2023	12/31/2023	1	New
Option	01/01/2021	12/31/2021	1	New
Option	01/01/2022	12/31/2022	1	New
Option	01/01/2023	12/31/2023	1	New

Assets are created for the ramped bundle and option.

During Renewal

During renewal, you see the assets with the following details.

Asset Type	Start Date	End Date	Selling Term	Line Status
Bundle	01/01/2024	12/31/2024	1	Renewed
Bundle	01/01/2025	12/31/2025	1	Renewed
Bundle	01/01/2026	12/31/2026	1	Renewed
Option	01/01/2024	12/31/2024	1	Renewed
Option	01/01/2025	12/31/2025	1	Renewed
Option	01/01/2026	12/31/2026	1	Renewed

Renewal with the Option Removed

You remove the option and go to the cart.

Asset Type	Start Date	End Date	Selling Term	Line Status
Bundle	01/01/2024	12/31/2024	1	Renewed
Bundle	01/01/2025	12/31/2025	1	Renewed
Bundle	01/01/2026	12/31/2026	1	Renewed
Option	01/01/2021	12/31/2021	1	Cancelled
Option	01/01/2022	12/31/2022	1	Cancelled
Option	01/01/2023	12/31/2023	1	Cancelled

Key Points to Note

- The line status for cancelled option is Cancelled.
- The cancelled option line item fetch the original asset start date and end date, not the renewed asset start date and date.
- The Contract_numbers (cancellation date) have the original asset end date for that ramp line when the asset is cancelled.

Managing Asset Increment with Coterminate Lines

As a Sales Representative, you can increase the quantity of an asset and manage these incremental licenses independently as a separate asset or you can chose to update the existing asset to reflect the new state after adding the incremental licenses. You can sell the incremental quantity at the same price point as the original sale or at a different price point. During the sale of these incremental licenses, you can coterminate them with the existing licenses.

> ⓘ In Winter 2018 release, CPQ on Salesforce does not support decreasing the quantity of an asset. If you decrease the quantity by clicking the minus icon (**–**), the **Go to Pricing** button is disabled.

1. Click Installed Products. The Installed Product page is displayed.
2. Select the check box for a product and click **Change** > **Quantity**. The Change Quantity page for the selected asset is displayed.
3. In the **Change quantity by** field, increase the quantity by clicking the + icon. The **New Quantity** field displays the updated quantity immediately.

4. From the **Cotermination** drop-down list, select a cotermination option for incremental licenses.

   > During the sale of incremental licenses, you can decide whether you want to coterminate them with the existing licenses. At the time of renewal, the incremental licenses can be renewed independently. You can selectively coterminate all or some of the incremental streams, or you can selectively merge all or some of the incremental streams into one stream. The **End Date** field is editable only if you select the **Enter Custom Date** option.
   > The options available depend on the settings defined by your Administrator. For more information, see Configuring Installed Products Settings.

   - **Retain the current Asset End Date**: Select this option to retain the current asset end date for cotermination.
   - **Enter Custom Date**: Select this option to enter a custom date for cotermination. The **End Date** field is editable if you select this option.
   - **Use the Proposal End Date**: Select this option for cotermination to use the proposal end date.

     > If you perform asset increment through the Asset Manager flow, the **Use the Proposal End Date** option is not displayed on the Change Quantity intermediate page.

5. Modify the **Start Date** and **End Date** fields if you selected Enter Custom Date from the **Cotermination** drop-down list.
6. Select the **Update Existing Assets** checkbox to enable these quantity changes to reflect in the existing asset.

   > If you selected **Enter Custom Date** in the **Cotermination** drop-down and modified the **End Date**, you cannot select the **Update Existing Assets** checkbox.

   - If you select this checkbox, on order activation, the incremental quantity updates into the existing asset. Line status is changed to Incremented and Merged. You can verify the quantity increment at Account > Asset Line Items list. The quantity increment is displayed under the Delta Quantity column.
     - When the price point, identifier and end date are the same, the existing quote line corresponding to the asset in the existing renewal quote is updated with the updated asset detail.

    - When the price point is a weighted average, and the identifier and end date are the same, the existing quote line corresponding to the asset in the existing renewal quote is updated with the updated asset detail.
- If you do not select this checkbox, on order activation, a new asset with the same name of the existing asset is created. Line status is changed to Incremented. You can verify the quantity increment at Account > Asset Line Items list. There are two assets with the same name, one with the original quantity and the other with the increment quantity. However, the Total Quantity column indicates the cumulative quantity, that is, original quantity plus increment quantity.

  > In CPQ on Salesforce Winter 2018 release, CPQ does not support the calculation of total quantity and asset quantity, and renewal automation on the lines with the Incremented status.

    - When the price point of the incremental asset is not the same as the existing asset (and you are not using the weighted average), but the identifier and the end date are the same, a new line is created in the existing renewal quote.
    - When the price point of the incremental asset is not the same as the existing asset (and you are not using the weighted average) and the end date is the same but the identifier is different, a new renewal quote is created.
    - When the end date of the incremental asset is not the same as the existing asset, a renewal quote is created.

7. To coterminate other asset streams with the current increment, in the Cotermination with Increment panel, select any or all assets.

   > The Cotermination with Increment panel is visible only if your administrator has not enabled the Hide Co-Term setting.
   > If your administrator has configured Purchase Identification Criteria, you can see other assets that match the purchase identifier of the current increment, in the Cotermination with Increment section.
   > You can coterminate a few other assets that are related to the current asset. The related assets are the assets that are available under Related Purchases on the Installed Product page. However, you can only coterminate related assets with the same end date, but cannot increment the quantity of the related assets.

If you selected **Enter Custom Date** in the **Cotermination** drop-down and modified the **End Date**, the new date is updated in the New End Date field of the selected related assets, under the Cotermination with Increment section. For more information, see Configuring Installed Products Settings.

8. Click **Go to Pricing**. The Cart page is displayed.
9. Click the  icon on the Asset Line Column to understand the projected asset details. An API is called and the Project Asset Value window is displayed. You can see the current asset unit price, current asset net price, updated asset unit price, and updated asset net price.

Splitting Assets

As a Sales rep, you can divide an existing asset into multiple split lines to negotiate an upsell quote with customers. You can split standalone assets and bundle assets that have option line items. You can split a bundled asset into multiple components each with its own quantity, start date, and end date. On each split line of the asset, you can perform various asset-based actions such as, split, split and swap, and split and renew.

Prerequisites

A Product Replacement Constraint Rule must be configured by your administrator for the Swap asset action.

To split an asset

1. Click Installed Products. The Installed Product page is displayed.
2. Select the check box for a product and click **Change** > **Split**. The Define Split <asset name> page is displayed. The original asset line item is in Existing status with the existing asset quantity, start date, and end date. One more split line is added with zero quantity, with the same asset start date and end date.
3. Select one of the following split asset actions.

   The split asset actions depend on what your administrator has configured. For more information, see Configuring Installed Products Settings.

   - **Split and Renew**: Click the more icon  and select **Renew** to split the asset into two split lines and renew the second split line.

- **Split and Swap**: Click the more icon  and select **Swap** to split the asset into two split lines and to exchange a split line of the existing asset with a predefined alternate asset. When you select **Swap**, the Swap Asset page is displayed. Click **Split** to return to the Define Split <asset name> page. You can now see the swapped asset for the Split and Swap split line. For more information, see Swapping an Asset.
- Click the add icon  to add more split lines.
- Click the delete icon  to remove an split line.

4. Enter the following details:

> For a bundle asset, you can edit the data either at the bundle line level or at the option item levels. When you edit the data at the option item level, the corresponding line status changes to Amend. The unedited options remain in the Existing status.
> You must edit the parameters on each line within the bundle construct.

Field	Description
Quantity	Enter the desired quantity for the split lines. You can also change the quantity of the original asset. When you change the quantity of the original asset, the status of the asset changes to Amend. Until you enter the quantity for the split lines, the **Go to Pricing** button is disabled. Option quantity is not cascaded to options for all split actions. You must populate the option quantity manually. Until then, the **Go to Pricing** button is disabled.
Start Date	All split lines fetch the original asset start date by default. Modify the split line start date if required. For Split-Renew and Split-Swap actions, the split line will have the modified start date on the cart page. If you do not enter a date on the option lines for a split component, the **Go to Pricing** button is not enabled.

Field	Description
End Date	All split lines fetch the original asset end date by default. Modify the split line end date if required. For Split-Renew and Split-Swap actions, the split line will have the modified end date on the cart page. ⓘ If you do not enter a date on the option lines for a split component, the **Go to Pricing** button is not enabled.

5. Click **Go to Pricing**. The cart page is displayed where you can also change the quantity, start date, and end date.
    - Existing asset: Unless you made any changes, the existing asset remains unchanged (with zero Delta Quantity). The original line item and split line (both standalone and bundle asset) are in Amended status. Unless you made any changes, the option product is in Existing status. If you made any changes on an option line item (for example, change in end date), the changes are rolled up to the bundle line item and status of both bundle and option is Amended.
    - Split: The split line is in Split status. Unless you made any changes, the split line retains the start date and end date of the original asset. The quantity of the split line is the total quantity that includes original quantity and new quantity.
    - Split and Renew: The split line is in Split and Renewed status. The start date and end date are renewed dates. The quantity of the renewed split line is the total quantity that includes original quantity and new quantity.
    - Split and Swap: The split line is in Upgraded status. The start date and end date are modified dates. Both split-swap bundle line item and option line item display the updated quantity.

      ⚠ The Asset Line Item column on the cart indicates that the new asset line items have references to the original asset line item.
      When the effective date of swap is before the end date of the original line, the swap line is coterminated with the original line. When the effective date of swap is after the end date of the original line, the swap line added for the whole term.
      For a bundle product, you can have different start date and end date combination.

To avoid constraint rules from being triggered on Split and Renewed or any other line status on the Cart, the administrator must have defined constraint rule criteria. For information on configuring constraint rules, see Creating Constraint Rules.

6. Click **Reprice**. You can now finalize your cart.
7. Navigate to the account. After the quote is accepted and the order is activated, the account displays all split lines as new asset line items.
8. Click an asset line item link. The Asset Line Item page is displayed.
9. Click **Related Asset Line Items (From)** or **Related Asset Line Items (To)** link. The New Related Asset Line Item window displays the relation between the original asset line item and the new split asset line item.
10. Click **Related Transaction History (From)** or **Related Transaction History (To)** link. The New Asset Transaction History window displays the complete history of a specified asset line item.

Merging Assets

As a Sales Representative, you can merge multiple assets created over a period of time as a result of actions such as new sale, quantity increment, split. You can consolidate individual streams of assets into a single asset, which can also help you avoid duplicates. You can track the origin of the newly-consolidated asset using the relationships created between the original assets and the merged asset.

ⓘ The Merge feature is supported only for standalone products with a single charge type.

Prerequisites

- There must be more than one asset line item.
- The asset line items that you want to merge must have the same product, charge type, selling frequency, price type, and any other customized field. For more information, see Configuring Installed Products Settings.

To merge assets

1. Click Installed Products. The Installed Product page is displayed.
2. Select the checkboxes for the required assets and click **Change** > **Merge**. If the assets lines have the common mandatory fields, the Merge popup is displayed.

> If the merge criteria are not fulfilled, an error message is displayed. For example, if the assets have different end dates, you cannot merge them.
> You must select at least two assets to see the **Merge** option under **Change**.

3. From the **Effective Date** calendar, select a start date for the new merged asset.

> The farthest start date is populated in the **Effective Date** field by default. The effective date must be greater than or equal to the farthest start date of the assets; otherwise, the **Merge** button is disabled. If you enter a date that is less than the farthest start date, an error message is displayed on the Merge popup.

4. Click **Merge**. The selected assets are merged into a new asset. The line status of the new asset is Merged with the start date you provided in the **Effective Date** field.

> The cart also loads the selected assets, but they are in Cancelled status. Their end date is one day before the start date of the merged asset. You cannot edit the end date of cancelled asset line items.
> The Merged asset does not retain the attributes from Cancelled assets.
> When the user merges two assets with the same purchase identifier, the merged line is associated with the same renewal quote that contained the original lines. For more information, see Use Case: Renewing Merged Assets with the Same Purchase Identifier.

The merge line with a different date creates a renewal quote.

- If both cancelled assets have the same purchase identifier as the one on the quote for merge, the renewal line for the new merged asset is created in the same renewal quote as the one that has the lines for the purchase identifier.
- If both cancelled assets have different purchase identifiers, a renewal quote is created for the merged asset.
  Merge lines are created on a new renewal quote and cancelled lines removed from the existing renewal quote.

5. On the Cart grid, hover the mouse on the Merged or Cancelled assets, click the more icon (⋮), and select **Related Line Items**. The related line items popup is displayed.

> The **Add/Remove**, **Save**, and **Split** buttons are visible for relationship types other than Merger.

You can see related line items of the cancelled assets and their weighted net price.

> The term on all related asset line items that are associated with the merged line is equal. The quantity on the merged line is the sum of all quantities on the related asset line items. The base price of the merged line is weighted average of the related lines. You can apply adjustments on the merged line, but the applied adjustment has no impact on the related asset line items.

Swapping an Asset

The Swap feature allows you to exchange an existing asset with a predefined alternate asset. You can also upgrade one asset with the next version of the asset. You can swap a standalone, bundle, and an option product. Swapping of a product depends on the Product Replacement Constraint Rule that is set by your administrator. As a sales rep, you can cancel an existing asset and replace it with an equivalent new sale either through a swap action or by adding the cancelled and new lines manually.

From the example discussed earlier, your subscriber wants to upgrade from the Beginner level of the Business Intelligence course to the Advanced level.

To swap an asset

> The fields displayed during swap depend on how your administrator has configured the **Asset Termination Fields** setting. For more information, see Configuring Installed Products Settings.

1. On the **Installed Products** page, select the product that you want to swap.
2. Select the product from **Available Products** section.
3. Click **Purchase**. The Confirm Installed Products Swap page or the Confirm Swap pop-up is displayed.
4. From the **Effective End Date** lookup, select the date from when you want the swap to be effective, and click **Calculate**.
   The current asset will end on the effective end date you selected and swapped asset will start from the next day. If the **Start Date** of the swapped asset is a future date, a validation message is displayed to enter a valid **Start Date**. For example, an asset A with **Start Date** = 1 Jan, 2017 and **End Date** = 31 Jan, 2017 is swapped with a product B on June 5, 2016. Instead of taking June 6, 2016, as **Start Date** for product B, the system shows a validation message that the **End Date** of product B should be after the **Start Date** of product A.
5. View the Description and the Difference in the Original and the New Product.

6. To proceed with the Swap, click **Confirm**. Click **Back to Selection** to go back to the last page.

> **Note**
>
> If you select multiple assets on the Installed Products page, the Swap button will be disabled.

7. Click **Finalize** to go back to the quote.

Field	Description
From	The product that you want to swap for is displayed here.
To	The new product that you want to swap to is displayed here.

The status of the renewed asset changes from *New* to *Upgraded*. To check the status of your assets, the Status Category of the proposal must be **Accepted**. To view the assets, select the account of the proposal and navigate to Asset Line Items (Sold To) related list.

You can swap an asset on the same day of its expiry. A new Line Item is created for the swapped product with the same term as the previous asset. CPQ calculates the selling term using the calendar by default. CPQ derives the total value of the asset on the cancelled line. CPQ calculates the net price (that is, the new TCV for the asset) by considering the term calculation logic (prorates each history line based on the term start date identified).

> CPQ can also calculate the selling term based on the billing preferences. When the Conga Billing package is installed, you must set the **Selling Term Calculation Method** setting to Billing Preference. In that case, CPQ calculates the selling term based on the billing preferences. For more information, Billing Preferences.

Mass Update for Assets

From the **Allow Mass Change** setting on the **Installed Products Settings** under **Custom Settings** page, you can enable or disable mass changes for *must configure* assets on the Installed Products page.

Action	Allow Mass Change	Type of Asset	Selection	Behavior
Swap	Unchecked	not Must Configure	Single Select	Enable swap and land on an intermediate page
Swap	Unchecked	Must Configure	Single Select	Enable swap and land on an intermediate page
Swap	Checked	not Must Configure	Single Select	Enable swap and land on an intermediate page
Swap	Checked	Must Configure	Single Select	Enable swap and land on an intermediate page
Swap	Checked	Must Configure	Multi Select	Disable Swap
Swap	Checked	not Must Configure	Multi Select	Disable Swap
Swap	Unchecked	Must Configure	Multi Select	Disable Swap
Swap	Unchecked	not Must Configure	Multi Select	Disable Swap

Terminating an Asset

Using the Terminate feature, you can cancel a standalone, fixed bundle, or a configurable bundle asset. You can terminate an asset midway through the selling term. When an asset is canceled, CPQ captures the Termination Date of an asset on the Line Item object in the **Contract Number** field in *CCYY-MM-DD* format, which is used by the order and asset to store the Cancellation Date. The appropriate order line items are created and assets are also updated immediately after termination.

The billing impact is shown during termination only if you have the Conga Billing package installed in your org. For more information, see Use Case: Terminating Assets When Billing Management is Installed. If the Conga Billing package is not installed, CPQ calculates the different values based on the calendar dates.

Prerequisites

- The fields displayed during termination depend on how your administrator has configured the **Asset Termination Fields** setting. For more information, see Configuring Installed Products Settings
- CPQ enables you to edit selected fields on a cancelled line, based on the configured **Editable Fields For Cancelled Lines**. You can edit the end date on the cancelled line and reprice the cart to calculate the net price. You can also edit a custom currency field to pass credit amounts to downstream applications through the proposal or order. Apart from the fields you selected, the remaining fields are read-only fields. You cannot edit sensitive managed package fields such as list price related currency fields. Refer to "Configuring Installed Product Settings" in CPQ on Salesforce Spring 2020 Administrator Guide.
    - The values you edit at the bundle level cascade to the associated option lines.
    - The end date of a cancelled bundled header line overrides any change to the end date at the option level.
    - If a particular field is read-only at the bundle level, the field is read-only at the option line level also in the Cart.
- When the Conga Billing package is not installed, it is recommended to display the Confirm Termination pop-up instead of the Confirm Termination page. In **Display Column Settings** for the **Display Type** *Asset Termination*, only the **Termination Date** field must be added. For more information, see Configuring Display Columns Settings.

To terminate an asset

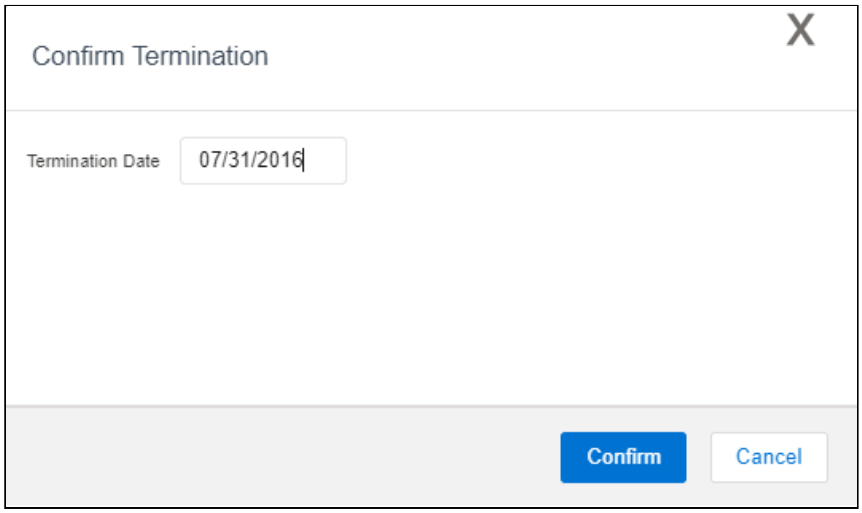
New Sale

1. From an account, create an opportunity.
2. Create a quote from the opportunity and configure products.
3. Add products to the cart and finalize it.
4. After your customer accepts the quote, an order is generated.
5. After you activate the order, assets are generated. The assets are active and visible on the account.

Termination

1. Create a quote to terminate assets.

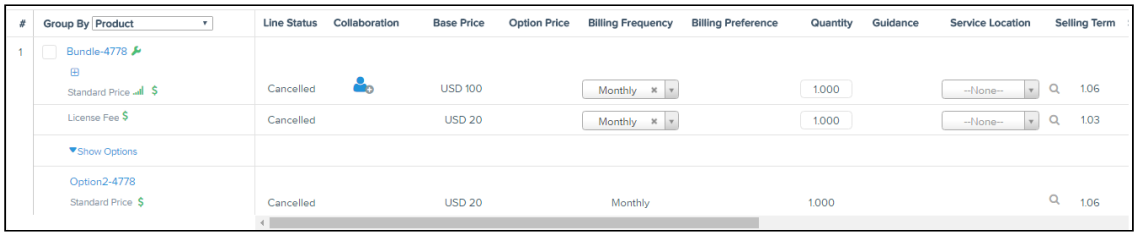
2. Navigate to the Installed Products page.
3. Select the assets that you want to cancel and click **Terminate**. The Confirm Termination pop-up is displayed.



4. Enter the **Termination Date** and click **Confirm**.
   The termination date can be in the past, the current date, or any date in the future. You can use custom logic to validate the date entered for the cancellation. If the validation fails, an error message is displayed.
   The status of the cancelled asset line items shows *Pending Cancellation* on the Installed Products page.
5. If your administrator has configured **Editable Fields for Cancelled Lines** and **Display Columns Settings**, CPQ displays the custom asset line item fields on the Confirm Termination page after you click **Terminate**.

> CPQ carries these custom asset line item field values from the Confirm Termination page to the Cart page where you can make further changes to those field values, if required.

6. Go to the Cart and review the following fields and their values.
   - Net Price
   - Line Status
   - Selling Term
   - End Date



> If you edited any of the custom asset line item fields on the Confirm Termination page, CPQ carries these edited field values from the Confirm Termination page to the Cart page where you can make further changes, if required.

7. Click **Finalize**.
   After you accept the quote and activate the order for the terminated assets, you can view the impact on the following fields on the Account page:

- For an Asset Line Item, **Asset Status** changes to *Cancelled*.
- For an Order Line Item, **Line Status** changes to *Cancelled*.

After the assets that are cancelled, CPQ selects the **Inactive** flag for the asset line items with *Cancelled* status so that these asset line items do not show on the Cart for the successive ABO actions.

Apart from the **Terminate** action, following are the scenarios when the status of the asset line item changes to *Cancelled*:

- While using **Change** action, if you deselect an Option product (inside a Bundle product) and finalize the Cart containing this Bundle Product. The status of the asset line item for Bundle product will show as *Amended* and that for the Option product will show as *Cancelled*.
- While using **Swap** action, if you replace a product with another product, the status of the asset line item for the former will show as *Cancelled*.

To check the status of your assets, the **Approval Stage** of the Quote/Proposal must be *Accepted*. To view the assets, select the account of the proposal and navigate to Asset Line Items (Sold To) related list.

A new entry is also made in the Asset Transaction History related list. For more information, see Asset Transaction History.

Use Case: Terminating Assets When Billing Management is Installed

If you have the Conga Billing package installed in your org, the billing impact is shown while terminating an asset. During billing, CPQ determines the refund value from the Cancellation Date captured in the asset and refunds to the customer accordingly. For more information, see Billing Concepts.

If Conga Billing is not installed, the system calculates the different values based on the calendar dates. For more information, see Terminating an Asset.

Prerequisites

- The fields displayed during termination depend on how your administrator has configured the **Asset Termination Fields** setting. For more information, see Configuring Installed Products Settings.
- CPQ enables you to edit selected fields on a cancelled line, based on the configured **Editable Fields For Cancelled Lines**. You can edit the end date on the cancelled line and reprice the cart to calculate the net price. You can also edit a custom currency field to pass credit amounts to downstream applications through the proposal or order. Apart from the fields you selected, the remaining fields are read-only fields. You cannot edit sensitive managed package fields such as list price related currency fields. For more information, see Configuring Installed Products Settings.
    - The values you edit at the bundle level cascade to the associated option lines.
    - The end date of a cancelled bundled header line overrides any change to the end date at the option level.
    - If a particular field is read-only at the bundle level, the field is read-only at the option line level also in the Cart.
- When the Conga Billing package is installed, it is recommended to display the Confirm Termination page instead of the Confirm Termination pop-up. In **Display Column Settings** for the **Display Type** *Asset Termination*, select all billing fields required. For more information, see Configuring Display Columns Settings.

To terminate an asset

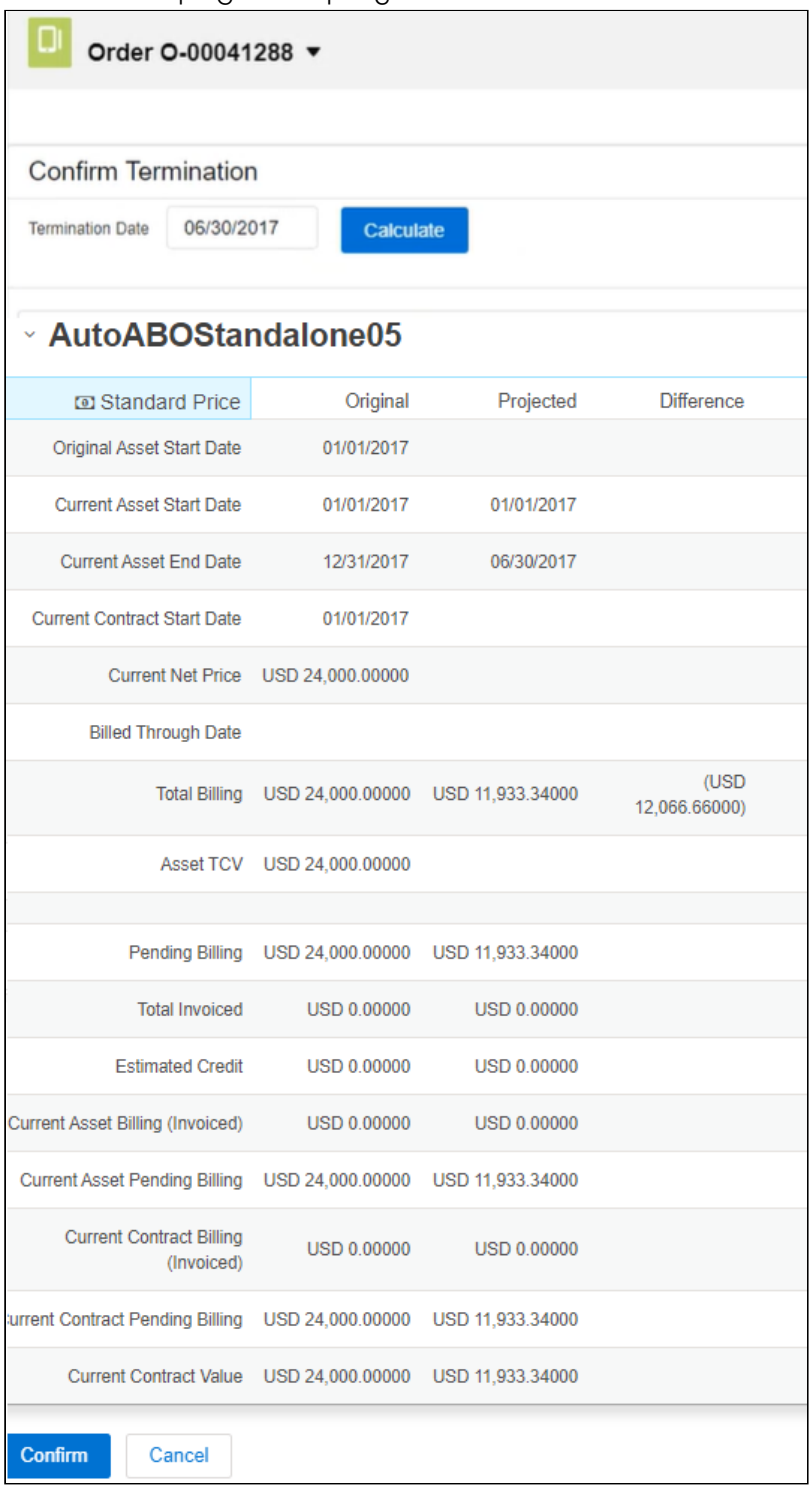
New Sale

1. From an account, create an opportunity.
2. Create a quote from the opportunity and configure products.
3. Add products to the cart and finalize it.
4. After your customer accepts the quote, an order is generated.
5. After you activate the order, assets are generated. The assets are active and visible on the account.

Termination

1. Create a quote to terminate assets.
2. Navigate to the Installed Products page.

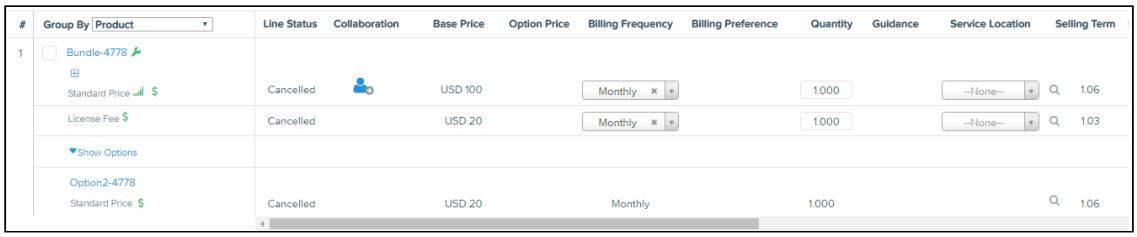
3. Select the assets that you want to cancel and click **Terminate**. The Confirm Termination page is displayed.



4. Enter the **Termination Date**.
   The termination date can be in the past, the current date, or any date in the future. You can use custom logic to validate the date entered for the cancellation. If the

validation fails, an error message is displayed.

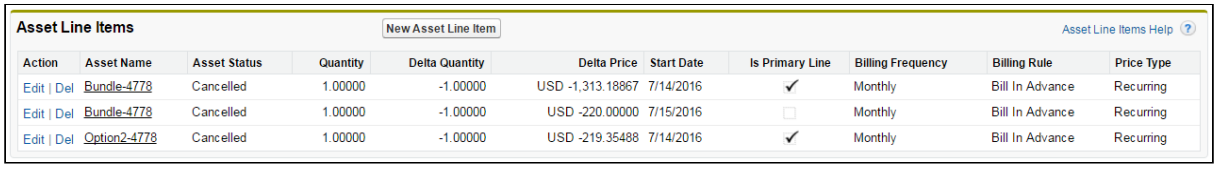
5. Click **Calculate** to calculate the difference in billing amount.
   CPQ displays values for different fields, such as the Pending Billing (Original = the amount you had paid, Projected = the amount to be credited/debited depending on the termination date, and Difference = the difference between the former field values). If your product has different Charge Types, the system displays the amount divided among the different Charge Types also. If your product is a Bundle and contains multiple Options, the system displays the amount divided among the Option products. Also, terminating a bundle cancels all the associated options configured inside that bundle.
   The Projected and Different amount is calculated simultaneously when you click **Calculate** using the Billing Schedules and Billing APIs. This gives you a quick overview of the difference you have to pay to or receive from the customer.
   For example, if your Asset, priced at USD 1200, contains the **Start Date** as 01/01/2016, **End Date** as 12/31/2016, and **Termination Date** as 07/31/2016, the amount populated in the different fields are:
   Original = USD 1200
   Projected = USD 300
   Difference = USD -900
6. Click **Confirm**.
   The status of the cancelled asset line items shows *Pending Cancelled* on the Installed Products page.
7. Go to the Cart and review the following fields and their values.
   - Net Price
   - Line Status
   - Selling Term
   - End Date



> You can view the change in the **Delta Quantity** and **Delta Price** on the Cart. For the above example, **Delta Price** is shown as USD -900.
> **Delta Quantity** and **Delta Price** is 0 for the following scenarios:
>
> i. Your Bundle or Standalone product has **Charge Type** = *One Time*.
> ii. You do not have Conga Billing installed in your org.

8. Click **Finalize**.
   After you accept the quote and activate the order for the terminated assets, you can view the impact on the following fields on the Account page:

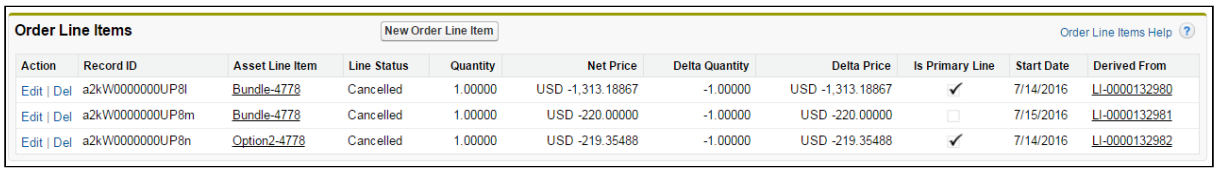
- For an Asset Line Item, **Asset Status** changes to *Cancelled*, **Delta Price** and **Delta Quantity** contain new values.



Asset Line Items — New Asset Line Item — Asset Line Items Help

Action	Asset Name	Asset Status	Quantity	Delta Quantity	Delta Price	Start Date	Is Primary Line	Billing Frequency	Billing Rule	Price Type
Edit | Del	Bundle-4778	Cancelled	1.00000	-1.00000	USD -1,313.18867	7/14/2016	✓	Monthly	Bill In Advance	Recurring
Edit | Del	Bundle-4778	Cancelled	1.00000	-1.00000	USD -220.00000	7/15/2016		Monthly	Bill In Advance	Recurring
Edit | Del	Option2-4778	Cancelled	1.00000	-1.00000	USD -219.35488	7/14/2016	✓	Monthly	Bill In Advance	Recurring

- For an Order Line Item, **Line Status** changes to *Cancelled*, **Delta Price** and **Delta Quantity** contain new values.



Order Line Items — New Order Line Item — Order Line Items Help

Action	Record ID	Asset Line Item	Line Status	Quantity	Net Price	Delta Quantity	Delta Price	Is Primary Line	Start Date	Derived From
Edit | Del	a2kW0000000UP8l	Bundle-4778	Cancelled	1.00000	USD -1,313.18867	-1.00000	USD -1,313.18867	✓	7/14/2016	LI-0000132980
Edit | Del	a2kW0000000UP8m	Bundle-4778	Cancelled	1.00000	USD -220.00000	-1.00000	USD -220.00000		7/15/2016	LI-0000132981
Edit | Del	a2kW0000000UP8n	Option2-4778	Cancelled	1.00000	USD -219.35488	-1.00000	USD -219.35488	✓	7/14/2016	LI-0000132982

After the assets are cancelled, CPQ selects the **Inactive** flag for the asset line items with *Cancelled* status so that these asset line items do not show on the Cart for the successive ABO actions.

To check the status of your assets, the **Approval Stage** of the Quote/Proposal must be *Accepted*. To view the assets, select the account of the proposal and navigate to Asset Line Items (Sold To) related list.

A new entry is also made in the Asset Transaction History related list. For more information, see Asset Transaction History.

Use Case: Same Day Cancellation

While terminating an asset, CPQ charges the customer for the period between the Start and End Date of an asset. If you have Conga Billing installed in your org, the billing impact is shown while terminating an asset. During billing, CPQ determines the refund value from the Cancellation Date captured in the asset and refunds to the customer accordingly. The refund amount differs based on the **Same Day Cancellation** setting. For more information on enabling this setting, see Billing System Properties.

- When the **Same Day Cancellation** is set to True, the asset cancellation is applicable on the same day. For example, while terminating an asset, you enter the **Termination Date** as 6/30/2017, the cancellation will be effective from 6/30/2017.

- If the **Same Day Cancellation** is set to False, the asset cancellation is applicable after a day. For example, while terminating an asset, you enter the **Termination Date** as 6/30/2017, the cancellation will be in effect a day later, that is 7/1/2017.

Same Day Cancellation is True

New Sale

You do a new sale of a product where:

Start Date	End Date	Selling Term	Net Price
01/01/2017	12/31/2017	12	24000

Asset Termination

You terminate the asset on 6/30/2017:

Start Date	End Date	Selling Term	Net Price	Delta Price
01/01/2017	6/30/2017	5.96667	11933.34	-12066.66 (refund)

Same Day Cancellation is False

New Sale

You do a new sale of a product where:

Start Date	End Date	Selling Term	Net Price
01/01/2017	12/31/2017	12	24000

Asset Termination

You terminate the asset on 6/30/2017:

Start Date	End Date	Selling Term	Net Price	Delta Price
01/01/2017	6/30/2017	6	12000	12000 (refund)

Use Case: Terminating Renewed Assets

If you terminate an asset after renewing it, the billing amount is calculated for the renewal period (not from the inception of the asset). CPQ displays the net price of the current year asset (renewed asset) on the Confirm Termination page.

> The termination date can be in the past, the current date, or any date in the future. You can use custom logic to validate the date entered for the cancellation.

New Sale

You do a new sale of a product where:

Start Date	End Date	Selling Term	Net Price
01/01/2019	12/31/2019	12	1200

Asset Renewal

You renew the asset where:

Start Date	End Date	Selling Term	Net Price
01/01/2020	10/31/2020	10	1000

Order Activation

After you activate the order:

Current Net Price of the Asset	Total Net Price of the Asset
1000	2200

Asset Termination

When you terminate the asset, CPQ displays the current net price of the asset as 1000 on the Confirm Termination page.

Use Case: Terminating an Asset with Ramp Lines

This use case describes how CPQ calculates the end date of an asset with ramps when you terminate the asset prematurely. For example, you sold a product with three ramps for three years. When you terminate the asset before the end date of the first ramp, CPQ sets the start date as the end date for the rest of future ramp lines and refunds full amount.

> The Same Day Cancellation is supported to work with *Billing Preference* as well as *Calendar* type of selling term calculation methods. For more information, see Use Case: Same Day Cancellation.

New Sale

You do a new sale of a product with three ramps:

Ramp	Start Date	End Date	Selling Term
1	1/1/2020	12/31/2020	12
2	1/1/2021	12/31/2021	12
3	1/1/2022	12/31/2022	12

Asset Termination

The following are different use cases of terminating assets with ramp lines.

> The **Same Day Cancellation** is set to *False* in these use cases. For more information on enabling this setting, see Billing System Properties.

Use Case 1

You terminate the asset with Terminate Date as 11/30/2020 (before the end date of the first ramp).

Ramp	Start Date	End Date	Refund
1	1/1/2020	11/30/2020	1 month

Ramp	Start Date	End Date	Refund
2	1/1/2021	1/1/2021	1 (full term refund)
3	1/1/2022	1/1/2022	1 (full term refund)

On the second and third ramp lines, the start will be equal to the end date and the full amount will be refunded.

Use Case 2

You terminate the asset with Terminate Date as 11/30/2021 (before the end date of the second ramp).

Ramp	Start Date	End Date	Refund
1	1/1/2020	12/31/2020	NA (because you consumed the entire term)
2	1/1/2021	11/30/2021	1 month refund
3	1/1/2022	1/1/2022	1 (full term refund)

On the third ramp, the start will be equal to the end date and the full amount will be refunded.

Use Case 3

You terminate the asset with Terminate Date as 11/30/2022 (before the end date of the third ramp).

Ramp	Start Date	End Date	Refund
1	1/1/2020	12/31/2020	NA (because you consumed the entire term)
2	1/1/2021	12/31/2021	NA (because you consumed the entire term)
3	1/1/2022	11/30/2022	1 month refund

> If Same Day Cancellation is set to *True*, the refund will be for 1 month and 1 day. For more information on how Same Day Cancellation works, see Use Case: Same Day Cancellation.

Suspending an Asset

You can suspend an asset for a particular period of time, which means the asset is temporarily not in effect. For example, ABC Corp is an electricity service provider who has installed meters at customer locations to measure the usage of the electricity. ABC Corp bills its customers based on their usage of the electricity. If a customer does not pay the bill for some time, a finance operations user from ABC Corp suspends the electricity service for that customer until the customer pays the due bill.

> When an asset is suspended, it is not available for any other ABO action (except *Terminate*) until you manually resume it. You can suspend assets only through Suspend APIs (SOAP and REST), but not on the UI. For more information, see Suspending Assets and CPQ Asset-Based Ordering APIs.

You must create and pass a business object (quote, agreement, or order) to be associated with the cart. When you execute the Suspend API code in the Developer Console, CPQ creates a transactional cart with the associated business object. After executing the API, the Asset Status field on asset line item is *Pending Suspension*. Upon order activation, the Asset Status field on asset line item is *Suspended*.

Key Points to Note

- The Pricing Status for the suspended asset line items on product configuration is *Complete*. CPQ does not perform pricing calculations on a cart if it has suspended assets.
- The Constraint Status Check for the suspended asset line items on product configuration is *Complete*. CPQ does not execute constraint rules on a cart if it has suspended assets.
- When you finalize the cart (using only the Finalize API), CPQ generates the respective line items for the associated business object. If the cart is a associated with a quote, agreement, or order, CPQ generates the proposal line items, agreement line items, or order line items respectively.
- A cart can only consist of line items with status *Suspended*. You cannot select line items with any other status because CPQ does not execute pricing or constraint rules on a cart if it has suspended assets.

Resuming an Asset

You can resume a suspended asset, which means bringing it back to effect and making it available for other ABO actions. For example, ABC Corp is an electricity service provider who has installed meters at customer locations to measure the usage of the electricity. ABC Corp bills its customers based on their usage of the electricity. Because a customer did not pay the bill for some time, a finance operations user from ABC Corp suspended the electricity service for that customer. Now that the customer has paid the due bills, the finance operations user resumes the electricity service for that customer.

> You can resume assets only through Resume APIs (SOAP and REST), but not on the UI. For more information, see Resuming Assets and CPQ Asset-Based Ordering APIs.

You must create and pass a business object (quote, agreement, or order) to be associated with the cart. When you execute the Resume API code in the Developer Console, CPQ creates a transactional cart with the associated business object. After executing the API, the Asset Status field on asset line item is *Pending Resume*. Upon order activation, the Asset Status field on asset line item is *Activated*.

Key Points to Note

- The Pricing Status for the activated (resumed) asset line items on product configuration is *Complete*. CPQ does not perform pricing calculations on a cart if it has activated assets.
- The Constraint Status Check for the resumed (resumed) asset line items on product configuration is *Complete*. CPQ does not execute constraint rules on a cart if it has resumed assets.
- When you finalize the cart (using only the Finalize API), CPQ generates the respective line items for the associated business object. If the cart is a associated with a quote, agreement, or order, CPQ generates the proposal line items, agreement line items, or order line items respectively.

Managing Assets through Contracts

CPQ allows you to manage assets through contracts. A contract or an agreement is a legally binding arrangement between two or more entities.

Prerequisites

- For the Assets and Contracts integration to work, the following packages must be available in your org:
    - Conga Contract Lifecycle Management
    - Conga Quote CLM Integration
    - Conga CLM Configuration Integration
- The **Auto Create Order** setting in **Comply System Properties** must be enabled if you want to create an order and asset as soon as an agreement is activated.

How asset and contract flows work

There are two process flows you must consider:

- Contract + Proposal Flow: In this flow, both Contract and Quote/Proposal are involved.
- Contract Flow: In this flow, only Contract is involved (no Quote/Proposal). The assets are created through contracts/agreements using the normal contract life cycle flow. After you finalize and activate a contract (containing agreement line items), the corresponding asset line items are created.

The following table explains the CPQ behavior when the **Auto Create Order** setting is configured.

Flow	Auto Create Order (Proposal System Properties)	Auto Create Order (Comply System Properties)	CPQ behavior
Contract + Proposal Flow	False	True	A Quote/Proposal is created followed by a Contract. The Asset Line Items and Order Line Items are created when you activate a Contract.
Contract Flow	False	True	Contract is created without a Quote/Proposal. The Asset and Order Line Items are created when you activate a Contract.

> It is recommended to refrain from enabling **Auto Activate Order** from **Comply System Properties** and **Proposal System Properties** simultaneously.
> The Conga Contract Lifecycle Management package version must be 8.4.0325 (8.325.1) or higher to use the features mentioned above.

Managing Assets in the Contract + Proposal Flow

This section explains how you can manage assets in the Contract + Proposal flow.

Prerequisite

Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.

To manage assets in the contract + proposal flow

1. In the Salesforce org, select **Apttus Proposal Management**.
2. On the **Proposals** tab, click **New**.
3. Enter requisite details, such as Proposal Name, Proposal Start Date, and Proposal End Date.
4. Associate an Account, an Opportunity, and a Price List to your record.
5. Click **Configure Products** to navigate to the Catalog and configure the required products with options and attributes.
6. After the configuration is finalized, click **Finalize** to navigate back to the Quote/Proposal detail page.
7. Click **Accept** to accept the finalized line items in your Quote/Proposal.
   After you have accepted the Quote/Proposal, **Create Agreement with Line Items** is active.
8. Click **Create Agreement with Line Items** to copy all Proposal Line Items to the Agreement Line Items. CPQ begins the agreement creation process.
9. Enter the mandatory details when prompted and save the agreement. The **Agreement Line Items** related list lists the agreement line items.
10. Click **Generate** to generate an agreement document.
11. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.

12. Refresh the org to see that the **Order** and **Order Line Items** related lists now contain records.
13. Go to the order record, enter the **Ready for Activation Date** and click **Save**. After you activate the order, assets are generated.

    > The asset line items are associated with the agreement, not with the proposal.

    The assets are visible on the account. The **Status** of all **Order Line Items** and **Asset Line Items** is *Activated*.

Changing Assets in the Contract + Proposal Flow

This section explains how you can change assets in the Contract + Proposal flow.

Prerequisite

- Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.
- Assets are already created.

To change assets in the contract + proposal flow

1. In the Salesforce org, select **Apttus Proposal Management**.
2. Create a proposal for asset change.
3. Click **Configure Products** and navigate to the Installed Products page.
4. Select the asset you want to update and click **Change**. On the cart page, **Line Status** changes to *Amended*.
5. You can change one or more of the following parameters:

Parameter	Description
Attribute Value	To change the Attribute value, click the wrench icon ( ) next to the asset. You will see the **Product Attributes** page. Select the new attribute value and click **Go to Pricing** to load the cart page.
Option	Expand the Show Options tree and select from the listing of the options to display all Product Options. From the resulting list, select the option with which you want to replace the original product option.

Parameter	Description
Quantity	Enter the number of units of the product your customer wants to purchase.
Adjustment Type	If you want to adjust the price of the asset, select an adjustment type.
Price	Enter a value that is the amount your customer is charged for the product without accounting for taxes and other charges.
Start Date	The date you want to start billing your customer for this purchase.
End Date	The date you want to stop billing your customer for this purchase.
Billing Frequency	Select one of the following options **Monthly** - To generate a bill once every month **Quarterly** - To generate a bill once every three months **Half-yearly**- To generate a bill once in the middle of a calendar or financial year **Yearly** - To generate a bill once every year **Usage** - To generate a bill based on usage For more information, see Billing Preferences.
Billing Rule	Select one of the following options **Bill in Advance** - To bill your customer before the product is delivered **Bill in Arrears** - To bill your customer after the product is delivered **Bill on Ready for Billing Date** - To bill your customer with a consolidate invoice, on a day of their choice.
Billing Preference	Select a predefined billing preference. For more information, see Billing Preferences.

Parameter	Description
Payment Term	Lookup and select a predefined Payment Term so your customers know that they must pay outstanding charges within a stipulated time. For more information, see Payment Term.

6. Click **Reprice** to view the changed price.
7. Click **Finalize** to navigate back to the quote. After you accept the quote, the **Create Agreement with Line Items** is active.
8. Click **Create Agreement with Line Items** to copy all Proposal Line Items to the Agreement Line Items. CPQ begins the agreement creation process.
9. Enter the mandatory details when prompted and save the agreement. The **Agreement Line Items** related list lists the agreement line items..

   > The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.

10. Click **Generate** to generate an agreement document.
11. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
    After you activate the order for the changed assets, the status of the changed asset changes from *New* to *Amended* on the account.

For more information, see Changing an Asset.

Renewing Assets in the Contract + Proposal Flow

This section explains how you can renew assets in the Contract + Proposal flow.

Prerequisite

- The **Renewal Business Object Type** is set to Proposal.
- Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.
- Assets are already created.

To renew assets in the contract + proposal flow

1. In the Salesforce org, select **Apttus Proposal Management**.
2. Create a proposal for asset renewal.

- You need to manually create a renewal quote only if you are renewing assets manually.
- If you are renewing assets in Auto Renewal Mode, the renewal quote is already created immediately after you activated the order during new sale. For more information, see Renewing Assets in Auto Renewal Mode.
- If you are renewing assets in OnDemand Renewal Mode, the renewal quote is created based on the defined **Renewal Lead Time** after you clicked a **custom button** on the account. For more information, see Renewing Assets in OnDemand Renewal Mode.

3. Click **Configure Products**.
   When you configure products in this renewal quote, CPQ takes you to the Cart page directly where assets with renewed Start Date, End Date, and Selling Term are available. The Line Status of assets is Renewed and asset pricing calculated according to the renewed Start Date, End Date, and Selling Term.

   On the new renewal quote, the **Start Date** = **Parent Quote's End Date** + 1 and **End Date** = **New Start Date** + Selling Term.

   The Approval Stage of the quote is marked back to Draft allowing you to perform any changes to the renewal configuration. Any change or update to the assets are reflected in the opportunity based renewal quote.
4. Add products to the cart if required.
5. Click **Finalize** to navigate back to the quote. After you accept the quote, the **Create Agreement with Line Items** is active.
6. Click **Create Agreement with Line Items** to copy all Proposal Line Items to the Agreement Line Items. CPQ begins the agreement creation process.
7. Enter the mandatory details when prompted and save the agreement. The **Agreement Line Items** related list lists the agreement line items..

   The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.
8. Click **Generate** to generate an agreement document.
9. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
   After you activate the order for renewed asset, the status of the renewed asset changes from *New* to *Renewed* on the account.

For more information, see Renewing an Asset.

Swapping Assets in the Contract + Proposal Flow

This section explains how you can swap assets in the Contract + Proposal flow.

Prerequisite

- Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.
- Assets are already created.

To swap assets in the contract + proposal flow

1. In the Salesforce org, select **Apttus Proposal Management**.
2. Create a proposal for asset swap.
3. Click **Configure Products** and navigate to the Installed Products page.
4. Select the assets that you want to swap.
5. Select the product from **Available Products** section.
6. Click **Purchase**. The Confirm Installed Products Swap page or the Confirm Swap pop-up is displayed.
7. From the **Effective End Date** lookup, select the date from when you want the swap to be effective, and click **Calculate**.
   The current asset will end on the effective end date you selected and swapped asset will start from the next day. If the **Start Date** of the swapped asset is a future date, a validation message is displayed to enter a valid **Start Date**. For example, an asset A with **Start Date** = 1 Jan, 2017 and **End Date** = 31 Jan, 2017 is swapped with a product B on June 5, 2016. Instead of taking June 6, 2016, as **Start Date** for product B, the system shows a validation message that the **End Date** of product B should be after the **Start Date** of product A.
8. View the Description and the Difference in the Original and the New Product.
9. To proceed with the Swap, click **Confirm**. Click **Back to Selection** to go back to the last page.

   > If you select multiple assets on the Installed Products page, the Swap button will be disabled.

10. Click **Finalize** to navigate back to the quote. After you accept the quote, the **Create Agreement with Line Items** is active.
11. Click **Create Agreement with Line Items** to copy all Proposal Line Items to the Agreement Line Items. CPQ begins the agreement creation process.

12. Enter the mandatory details when prompted and save the agreement. The **Agreement Line Items** related list lists the agreement line items..

    > The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.

13. Click **Generate** to generate an agreement document.
14. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
    After you activate the order for swapped asset, the status of the swapped asset changes from *New* to *Upgraded* on the account.

For more information, see Swapping an Asset.

Terminating Assets in the Contract + Proposal Flow

This section explains how you can terminate assets in the Contract + Proposal flow.

Prerequisite

- Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.
- Assets are already created.

To terminate assets in the contract + proposal flow

1. In the Salesforce org, select **Apttus Proposal Management**.
2. Create a proposal for asset termination.
3. Click **Configure Products** and navigate to the Installed Products page.
4. Select the assets that you want to cancel and click **Terminate**. The Confirm Termination pop-up is displayed.
5. Enter the **Termination Date** and click **Confirm**.
   The termination date can be in the past, the current date, or any date in the future. You can use custom logic to validate the date entered for the cancellation. If the validation fails, an error message is displayed.
   The status of the cancelled asset line items shows *Pending Cancellation* on the Installed Products page.
6. Go to the Cart and review the following fields and their values.
   - Net Price
   - Line Status
   - Selling Term

- End Date

7. Click **Finalize** to navigate back to the quote. After you accept the quote, the **Create Agreement with Line Items** is active.
8. Click **Create Agreement with Line Items** to copy all Proposal Line Items to the Agreement Line Items. CPQ begins the agreement creation process.
9. Enter the mandatory details when prompted and save the agreement. The **Agreement Line Items** related list lists the agreement line items..

   > The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.

10. Click **Generate** to generate an agreement document.
11. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
    After you activate the order for the terminated assets, the status of the terminated asset changes from *New* to *Cancelled* on the account.

For more information, see Terminating an Asset.

Managing Assets in the Contract Flow

This section explains how you can manage assets in the Contract flow.

Prerequisite

The custom setting **Auto Create Order** must be set to True under **Setup > Develop > Custom Settings > Comply System Properties**.

To manage assets in the contract flow

1. In the Salesforce org, select **Apttus Contract Management**.
2. On the **Agreements** tab, click **New**.
3. Enter required details, such as Agreement Name, Agreement Start Date, and Agreement End Date.
4. Select the **Single Transaction Adjustment** checkbox to control the adjustment of the billing based on the context in which the discount is applied. When you select this checkbox on an agreement, billing schedules are adjusted from the start of the contract term. For more information, see Calculating Billing Amount based on the Contract Term and the Single Transaction Adjustment Flag.
5. Associate an Account and a Price List to your agreement record.

6. Click **Configure Products** to navigate to the Catalog and configure the required products with options and attributes.
7. After the configuration is finalized, click **Finalize** to navigate back to the agreement. CPQ creates the agreement line items, which are visible under the **Agreement Line Items** related list.
8. Now that your line items finalized, click **Generate** to generate an agreement document.
9. After the negotiations with your customer is complete, click **Activate** to put the agreement in effect.
10. Refresh the org to see that the **Order** and **Order Line Items** related lists now contain records.
11. Go to your order record and enter the **Ready for Activation Date** and click **Save**. After you activate the order, assets are generated. On the account, the **Status** of all your **Order Line Items** and **Asset Line Items** is *Activated*.

Changing Assets in the Contract Flow

This section explains how you can change assets in the Contract flow.

Prerequisite

- Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.
- Assets are already created.

To change assets in the contract flow

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create an agreement for asset change.

   > The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.

3. Click **Configure Products** and navigate to the Installed Products page.

   > Using the **Amend** button on an agreement does not amend assets. You must use the **Configure Products** button on the agreement to change assets through the Cart. Also, CPQ allows you to modify all assets associated with an account, not just the assets associated with that particular agreement.

4. Select the asset you want to update and click **Change**. On the cart page, **Line Status** changes to *Amended*.
5. You can change one or more of the following parameters:

Parameter	Description
Attribute Value	To change the Attribute value, click the wrench icon ( ) next to the asset. You will see the **Product Attributes** page. Select the new attribute value and click **Go to Pricing** to load the cart page.
Option	Expand the Show Options tree and select from the listing of the options to display all Product Options. From the resulting list, select the option with which you want to replace the original product option.
Quantity	Enter the number of units of the product your customer wants to purchase.
Adjustment Type	If you want to adjust the price of the asset, select an adjustment type.
Price	Enter a value that is the amount your customer is charged for the product without accounting for taxes and other charges.
Start Date	The date you want to start billing your customer for this purchase.
End Date	The date you want to stop billing your customer for this purchase.
Billing Frequency	Select one of the following options **Monthly** - To generate a bill once every month **Quarterly** - To generate a bill once every three months **Half-yearly**- To generate a bill once in the middle of a calendar or financial year **Yearly** - To generate a bill once every year **Usage** - To generate a bill based on usage For more information, see Billing Preferences.

Parameter	Description
Billing Rule	Select one of the following options **Bill in Advance** - To bill your customer before the product is delivered **Bill in Arrears** - To bill your customer after the product is delivered **Bill on Ready for Billing Date** - To bill your customer with a consolidate invoice, on a day of their choice.
Billing Preference	Select a predefined billing preference. Select a predefined billing preference. For more information, see Billing Preferences.
Payment Term	Lookup and select a predefined Payment Term so your customers know that they must pay outstanding charges within a stipulated time. For more information, see Payment Term.

6. Click **Reprice** to view the changed price.
7. Click **Finalize** to navigate back to the agreement. CPQ creates the agreement line items.
8. Click **Generate** to generate an agreement document.
9. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
   After you activate the order for the changed assets, the status of the changed asset changes from *New* to *Amended* on the account.

For more information, see Changing an Asset.

Renewing Assets in the Contract Flow

Renewals are the most common and effective ways to retain existing customers and drive growth and sales for your business. Asset renewals eliminate the scope of pricing errors by extending the agreement. Therefore when you renew an asset, you basically regenerate the asset life-cycle for a new duration. CPQ supports the following types of renewals:

- Manual renewal of assets
- Auto renewal of assets

Renewing Assets Manually in the Contract Flow

CPQ allows you to renew an existing asset anytime you want, based on your business requirements.

To renew an asset

New Sale

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create an agreement.
3. Add products to the cart.
   If the administrator has configured the **Renewal Adjustment Type** and **Renewal Adjustment Amount** columns for the cart of the current flow, you will see those columns.
   a. In the **Renewal Adjustment Type** drop-down for a line item, select *% Uplift* by how much you want to uplift the asset price during renewal. In this case, enter the required percentage in the **Renewal Adjustment Amount** field of the line item.
   b. In the **Renewal Adjustment Type** drop-down for a line item, select *Uplift Amount* by how much you want to uplift the asset price during renewal. In this case, enter the required uplift amount in the **Renewal Adjustment Amount** field of the line item.

   > CPQ considers these renewal adjustments only during asset renewal. These values do not impact the current transaction.
4. Click **Finalize** to navigate back to the agreement. CPQ creates the agreement line items.
5. Click **Generate** to generate an agreement document.
6. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
7. Go to the order record, enter the **Ready for Activation Date** and click **Save**.
   After you activate the order, assets are generated. On the account, the **Status** of all your **Order Line Items** and **Asset Line Items** is *Activated*.

Renewal

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create an agreement for asset renewal.

   > The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.

3. Click **Configure Products** and navigate to the Installed Products page.

   > Using the **Renew** button on an agreement does not renew assets. You must use the **Configure Products** button on the agreement to renew assets through the Cart. Also, CPQ allows you to renew all assets associated with an account, not just the assets associated with that particular agreement.

4. Select the assets that you want to renew.
5. Click **Renew**.

   > If an asset is associated with a system-generated renewal agreement and the **Alert Asset Related To Renewal Cart** setting is enabled, a warning is displayed about the existence of a renewal agreement for this asset. The **Renewal Line Item** field on the asset indicates which renewal configuration is this asset currently associated with. Click **Proceed without delinking** to continue with manual renewal or **Cancel** to go to the Installed Products page for further action.

   If the asset is not associated with any system-generated renewal agreement or if the **Alert Asset Related To Renewal Cart** setting is disabled, CPQ either reloads the Installed Products page or displays the Confirm Renewal pop-up or the Confirm Renewal intermediate page depending on how your administrator has configured the **Cotermination Preferences During Renewal** and **Default Renewal Cotermination Option** settings. For more information, see Configuring Custom Settings.
6. Perform one of the following actions:
   - If the Installed Products page is reloaded, click **Go to Pricing**. The cart page is displayed.
   - If the Confirm Renewal intermediate page is displayed, select one from the following options to define the renewal date and click **Confirm**. The cart page is displayed.

- **Use Agreement End Date:** To renew the asset until the date agreement ends. The Agreement End Date is displayed next to the option.

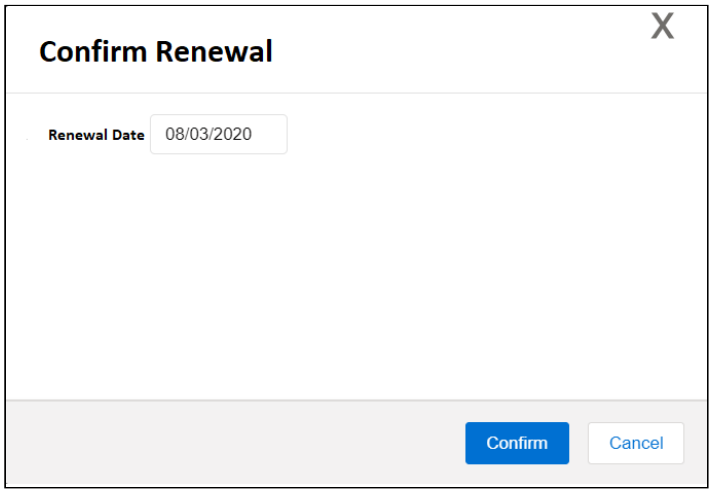
  > ⓘ If you perform asset renewal through the Asset Manager flow or CSR flow, the **Use Agreement End Date** option is not displayed on the Confirm Renewal intermediate page.

- **Retain the current Asset End Date:** To renew the current asset on the date it expires.
- **Use the Farthest Asset End Date:** To choose the farthest end date of all assets for renewal.

  > ⓘ This option is visible only if the asset has more than one renewable charge types or if the selected asset has multiple renewable lines with different end dates.

- **Enter Renewal Date:** To select a custom renewal date.

- If the Confirm Renewal pop-up is displayed, select a date in the **Renewal Date** field and click **Confirm**. The cart page is displayed.



7. On the cart, make changes to the asset if required.
   While finalizing the original agreement, if you have specified the renewal adjustment (%Uplift or Uplift Amount), you will see the change in the Net Price. Click **Confirm**.

   > If the renewal is loaded with asset pricing and if the renewal adjustment was specified during the original sale, the base price on the renewal transaction reflects the adjustment. If there is not asset pricing, the base price is the new list price of the asset.

8. Click **Reprice** to apply and load the asset with the changes on the cart page.
9. Click **Finalize** to navigate back to the agreement. CPQ creates the agreement line items.
10. Click **Generate** to generate an agreement document.
11. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
12. Go to the order record, enter the **Ready for Activation Date** and click **Save**.
13. After you activate the order, renewal assets are generated. On the Accounts page, you can verify that the status of the renewed asset changes from *New* to *Renewed*. You can trace the renewal activity and modifications from the **Order Line Items** and **Asset Transaction History** Related Lists.

> **Line Status** for one-time products will remain *Existing*.

Renewing Assets Automatically in the Contract Flow

CPQ allows you to auto-generate renewal agreements based on the fulfillment of order or the expiration of subscription. Through renewal agreements, you can forecast and analyze

the sales pipeline. With renewal agreements, you get the finer visibility into the pricing and configuration of assets within the renewal pipeline. The benefits of auto renewals are:

- Accurate forecasting on the asset state and sales pipeline
- Faster and simpler processing with automated renewal
- Automatic closure of opportunity on asset expiry

Types of Asset Renewal Modes

There are two types of renewal modes, defined in **Renewal Execution Mode**:

- Auto
- OnDemand

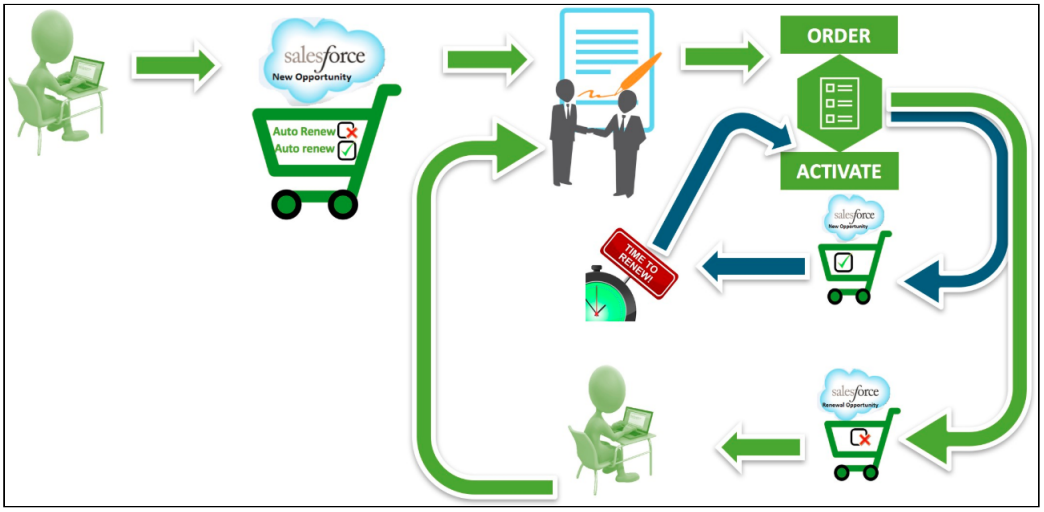
> It is recommended not to use these renewal modes interchangeably.

Renewing Assets in Auto Renewal Mode in the Contract Flow

When **Renewal Execution Mode** is set to *Auto*, CPQ automatically renews an existing agreement for all assets it has, as soon as the order is activated. You may want to renew an existing opportunity in Auto renewal mode when:

- You require renewal agreements for forecasting
- You must work on renewal agreements in advance

The following diagram describes the flow of **Auto** renewals.



Prerequisites

- The **Renewal Business Object Type** is set to Agreement.
- The **Renewal Execution Mode** is set to *Auto*.
- The **Renewal Group Fields** are defined.
- The AssetRenewalJobScheduler batch job is scheduled.

How Auto Renewal Mode Works

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create an agreement.
3. Add products to the cart and finalize it.
4. Click **Generate** to generate an agreement document.
5. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
6. After you activate the order, assets are generated. The assets are active and visible on the account.
7. CPQ creates auto renewal agreements immediately after you activate the order.

   By default, CPQ groups asset line items into renewal agreements by the **Auto Renew** flag and other **Renewal Group Fields** defined. There will be two renewed agreements: one for the asset lines with Auto Renew = True and the other for asset lines with Auto Renew = False.

   > Renewal agreement that has assets with Auto Renew = True: On the expiry date of the asset, CPQ automatically processes the renewal agreement and renews the asset for the next term. The agreement will be automatically set to Accepted without any intervention (this is called Touchless Renewal). Renewal agreement that has assets with Auto Renew = False: You need to manually process the agreement.

   On the Account page, the renewed asset groups are listed under Temp Renew Asset Groups in the New status. After the AssetRenewalJobScheduler job is executed, the status of asset groups changes to Completed and the renewal agreement is created. The name of the renewal agreement will be Renew:<original_agreement_id>-<name_of_price_list>-<agreement_end_date>.

   When the renewal agreement is generated, the **Renewal Line Item** field on the asset line item is updated with the latest renewal configuration ID on the renewal agreement. When the user tries to manually renew this asset, a warning is displayed about the existence of a renewal agreement for this asset.

8. When you configure products in this renewal agreement, CPQ takes you to the Cart page directly where assets with renewed Start Date, End Date, and Selling Term are available. The Line Status of assets is Renewed and asset pricing calculated according to the renewed Start Date, End Date, and Selling Term.

> On the new renewal agreement, the **Start Date** = **Parent agreement's End Date** + 1 and **End Date** = **New Start Date** + Selling Term.

The Approval Stage of the agreement is marked back to Draft allowing you to perform any changes to the renewal configuration. Any change or update to the assets are reflected in the opportunity based renewal agreement.

9. Add products to the cart if required.
10. Click **Finalize** to navigate back to the agreement.
11. Click **Generate** to generate an agreement document.
12. After you accept the agreement, an order is generated.

Renewing Assets in OnDemand Renewal Mode in the Contract Flow

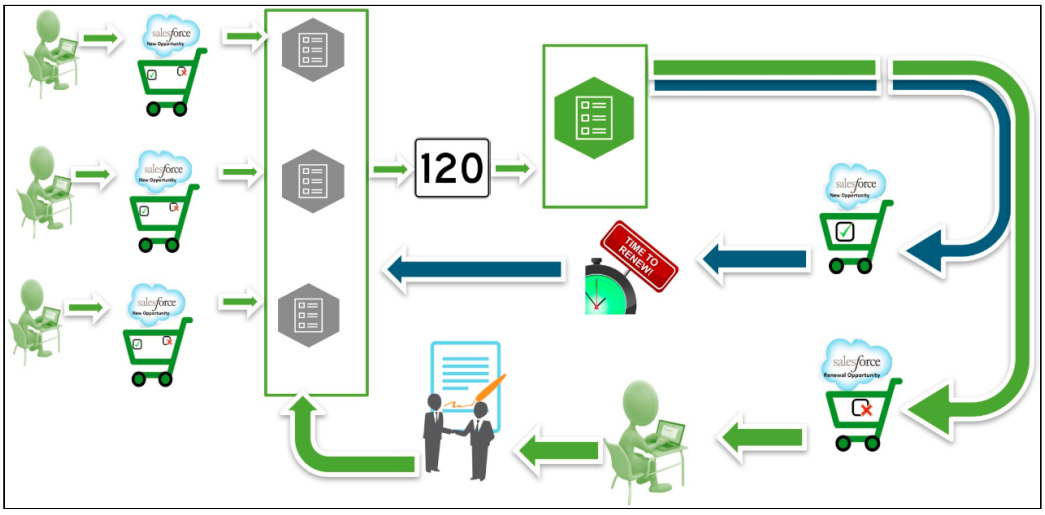
When **Renewal Execution Mode** is set to *OnDemand*, CPQ automatically renews all assets based on the defined **Renewal Lead Time**. CPQ supports Account mode and Product mode for OnDemand renewal of assets. However, these modes are mutually exclusive, which means you must use only one mode in one org. You may want to renew an existing opportunity in OnDemand renewal mode:

- During one-time legacy asset data migrations
- During one time renewal agreement creation for existing active asset base
- When minimal changes are required on the renewal agreements because of any ABO changes

Account Mode

This mode works based on the global **Renewal Lead Time** setting in Installed Product Settings. CPQ automatically renews all assets on the account (possibly from different agreements) based on the defined global **Renewal Lead Time** after you click a **custom button** on the account.

The following diagram describes the flow of **OnDemand** renewals. In this diagram, **Renewal Lead Time** is 120 days.



Over a period of time, you sell some products/services through different agreements, which might have varying end dates. After the original agreements are converted into orders and the order are activated, assets are generated. All assets are visible on the account. When you want to renew the assets that will expire in 120 days from today, you click the **custom button** on the account. CPQ compares the asset line items' end dates with the **Renewal Lead Time** defined and triggers the creation of renewal agreements for all those assets (from different agreements) that will expire in 120 days.

Prerequisites

- The **Renewal Business Object Type** is set to Agreement.
- The **Renewal Execution Mode** is set to OnDemand.
- A custom button is created on the account or a custom link is created on the quote.

  > ⓘ If the custom button is on the account, you can renew all assets in the account based on the defined Renewal Lead Time. If the custom link on the quote, you can renew assets from a quote based on the defined Renewal Lead Time.

- The **Renewal Lead Time** is defined.
- The **Renewal Group Fields** are defined.
- The *AssetRenewalJobScheduler* batch job is scheduled.

How OnDemand Renewal Mode Works

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create an agreement.
3. Add products to the cart and finalize it.
4. Click **Generate** to generate an agreement document.
5. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.

6. After you activate the order, assets are generated. The assets are active and visible on the account.
7. Click the **custom button** on the account when you want to trigger the creation of renewal agreements.
8. CPQ creates renewal agreements based on the defined **Renewal Lead Time**, considering the expiry date of assets (from different agreements).

   By default, CPQ groups asset line items by Account and the Price List. However, if other parameters are configured in **Renewal Group Fields**, CPQ groups asset line items accordingly. If some assets had **Auto Renew** flag set to True or False, there will be two renewed agreements: one for the asset lines with Auto Renew = True and the other for asset lines with Auto Renew = False.

   > Renewal agreement that has assets with Auto Renew = True: On the expiry date of the asset, CPQ automatically processes the renewal agreement and renews the asset for the next term. The agreement will be automatically set to Accepted without any intervention (this is called Touchless Renewal). Renewal agreement that has assets with Auto Renew = False: You need to manually process the agreement.

   After CPQ triggers renewal agreement creation, on the Account page, the renewed asset groups are listed under Temp Renew Asset Groups in the New status. After the *AssetRenewalJobScheduler* job is executed, the status of asset groups changes to Completed and the renewal agreement is created. The name of the renewal agreement will be Renew:<name_of_price_list>-<agreement_end_date>.

   When the renewal agreement is generated, the **Renewal Line Item** field on the asset line item is updated with the latest renewal configuration ID on the renewal agreement. When the user tries to manually renew this asset, a warning is displayed about the existence of a renewal agreement for this asset.

9. When you configure products in this renewal agreement, CPQ takes you to the Cart page directly where assets with renewed Start Date, End Date, and Selling Term are available. The Line Status of assets is Renewed and asset pricing calculated according to the renewed Start Date, End Date, and Selling Term.

   > On the new renewal agreement, the **Start Date** = **Parent agreement's End Date** + 1 and **End Date** = **New Start Date** + Selling Term.

The Approval Stage of the agreement is marked back to Draft allowing you to perform any changes to the renewal configuration. Any change or update to the assets are reflected in the opportunity based renewal agreement.

10. Add products to the cart if required.
11. Click **Finalize** to navigate back to the agreement.
12. Click **Generate** to generate an agreement document.
13. After you accept the agreement, an order is generated.

Product Mode

This mode works based on the product-level **Renewal Lead Time**. CPQ automatically renews all assets (possibly from different agreements across all accounts in the org) based on the defined product-level **Renewal Lead Time** after you execute a code in Developer Console.

Prerequisites

- The **Renewal Business Object Type** is set to Agreement.
- The **Renewal Execution Mode** is set to OnDemand.
- The **Renewal Lead Time** is defined for the required products. See Creating Products. For example, there are two products with the Renewal Lead Time of 20 and 40 days.
- The **Renewal Group Fields** are defined.
- The *AssetRenewalJobScheduler* batch job is scheduled.

How OnDemand Renewal Mode Works

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create agreements and configure products with Renewal Lead Time.
3. Add products to the cart and finalize it.
4. Click **Generate** to generate an agreement document.
5. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
6. After you activate the order, assets are generated.
7. Run the following code in Developer Console:

   ⚠ In the following code, (20,5,100,30} is used as an example Renewal Lead Time. CPQ allows you to use multiple lead times separated by comma.

```
List<Integer> productLeadTimes = new List<Integer>

{20,5,100,30}
```

```
;

Apttus_Config2.AssetRenewalSubmitController controller = new
 Apttus_Config2.AssetRenewalSubmitController(productLeadTimes);

// do submit
ID jobId = controller.doSubmitJob();
```

8. CPQ creates renewal agreements based on the defined **Renewal Lead Time**, considering the expiry date of assets (from different agreements across all accounts in the org). For example, if there are two products with the Renewal Lead Time of 20 and 40 days, only the product with Renewal Lead Time=20 is triggered and its OnDemand renewal agreement is created.

   By default, CPQ groups asset line items by Account and the Price List. However, if other parameters are configured in **Renewal Group Fields**, CPQ groups asset line items accordingly. If some assets had **Auto Renew** flag set to True or False, there will be two renewed agreements: one for the asset lines with Auto Renew = True and the other for asset lines with Auto Renew = False.

   > ⓘ Renewal agreement that has assets with Auto Renew = True: On the expiry date of the asset, CPQ automatically processes the renewal agreement and renews the asset for the next term. The agreement will be automatically set to Accepted without any intervention (this is called Touchless Renewal). Renewal agreement that has assets with Auto Renew = False: You need to manually process the agreement.

   After CPQ triggers renewal agreement creation, on the Account page, the renewed asset groups are listed under Temp Renew Asset Groups in the New status. After the *AssetRenewalJobScheduler* job is executed, the status of asset groups changes to Completed and the renewal agreement is created. The name of the renewal agreement will be Renew:<name_of_price_list>-<agreement_end_date>.

   When the renewal agreement is generated, the **Renewal Line Item** field on the asset line item is updated with the latest renewal configuration ID on the renewal agreement. When the user tries to manually renew this asset, a warning is displayed about the existence of a renewal agreement for this asset.
9. When you configure products in this renewal agreement, CPQ takes you to the Cart page directly where assets with renewed Start Date, End Date, and Selling Term are available. The Line Status of assets is Renewed and asset pricing calculated according to the renewed Start Date, End Date, and Selling Term.

> On the new renewal agreement, the **Start Date** = **Parent agreement's End Date** + 1 and **End Date** = **New Start Date** + Selling Term.

The Approval Stage of the agreement is marked back to Draft allowing you to perform any changes to the renewal configuration. Any change or update to the assets are reflected in the opportunity based renewal agreement.

10. Add products to the cart if required.
11. Click **Finalize** to navigate back to the agreement.
12. Click **Generate** to generate an agreement document.
13. After you accept the agreement, an order is generated.

Swapping Assets in the Contract Flow

This section explains how you can swap assets in the Contract flow.

Prerequisite

- Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.
- Assets are already created.

To swap assets in the contract flow

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create an agreement for asset swap.

> The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.

3. Click **Configure Products** and navigate to the Installed Products page.
4. Select the assets that you want to swap.
5. Select the product from **Available Products** section.
6. Click **Purchase**. The Confirm Installed Products Swap page or the Confirm Swap pop-up is displayed.
7. From the **Effective End Date** lookup, select the date from when you want the swap to be effective, and click **Calculate**.
   The current asset will end on the effective end date you selected and swapped asset will start from the next day. If the **Start Date** of the swapped asset is a future date, a validation message is displayed to enter a valid **Start Date**. For example, an asset A with **Start Date** = 1 Jan, 2017 and **End Date** = 31 Jan, 2017 is swapped with a product B

on June 5, 2016. Instead of taking June 6, 2016, as **Start Date** for product B, the system shows a validation message that the **End Date** of product B should be after the **Start Date** of product A.

8. View the Description and the Difference in the Original and the New Product.
9. To proceed with the Swap, click **Confirm**. Click **Back to Selection** to go back to the last page.

   > If you select multiple assets on the Installed Products page, the Swap button will be disabled.

10. Click **Finalize** to navigate back to the agreement. CPQ creates the agreement line items.
11. Click **Generate** to generate an agreement document.
12. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
    After you activate the order for swapped asset, the status of the swapped asset changes from *New* to *Upgraded* on the account.

For more information, see Swapping an Asset.

Terminating Assets in the Contract Flow

This section explains how you can terminate assets in the Contract flow.

Prerequisite

- Ensure that the custom setting **Auto Create Order** is set to True under **Setup > Develop > Custom Settings > Comply System Properties**.
- Assets are already created.

To terminate assets in the contract flow

1. In the Salesforce org, select **Apttus Contract Management**.
2. Create an agreement for asset termination.

   > The new agreement is not related to the agreement that was created during the new sale. CPQ does not support versioning of the original agreement.

3. Click **Configure Products** and navigate to the Installed Products page.

> Do not use the **Terminate** button on the agreement to terminate assets. If you do so, CPQ terminates only the agreement. In that case, you must terminate the corresponding assets manually unless you do partial termination of the assets using the **Amend** button on the agreement.

4. Select the assets that you want to cancel and click **Terminate**. The Confirm Termination pop-up is displayed.
5. Enter the **Termination Date** and click **Confirm**.
   The termination date can be in the past, the current date, or any date in the future. You can use custom logic to validate the date entered for the cancellation. If the validation fails, an error message is displayed.
   The status of the cancelled asset line items shows *Pending Cancellation* on the Installed Products page.
6. Go to the Cart and review the following fields and their values.
   - Net Price
   - Line Status
   - Selling Term
   - End Date
7. Click **Finalize** to navigate back to the agreement. CPQ creates agreement line items.
8. Click **Generate** to generate an agreement document.
9. After the negotiations are complete with your customer, click **Activate** to bring the agreement in effect.
   After you activate the order for the terminated assets, the status of the terminated asset changes from *New* to *Cancelled* on the account.

   > However, the agreement you created to terminate assets is still active. You must terminate it manually to cancel the agreement.

For more information, see Terminating an Asset.

Viewing the Asset Transaction History

The **Asset Transaction History** related list on the Asset Line Item detail page displays information for every Asset-based Order. For every ABO action (Change, Renew, Swap or Terminate), a new transaction and order record is created inside the Asset Transaction History related list.

In the Asset Transaction History related list, click the transaction ID to view the details.

The **Asset Line Item History** denotes the list of transactions made on the same Asset Line Item.

Billing for Assets

After you finalize a proposal and accept a customer's order, the products change from order line item to asset line item. Then, you create an Asset-based Order by creating a quote on the account that contains the installed products or assets. After you create the quote, go to product cart, click **Installed Products**, and select one of the following actions:

- Renew
- Terminate
- Swap
- Change

Every asset action on an asset line item on Installed Product page changes the existing **Billing Schedule**, **Invoice Status**, and the **Billing Cycle Start and End Date** for that asset line item or the Installed Product.

Renewal of an asset will impact the Billing Schedules because the asset **Start Date** and **End Date** will change. On renewal, the **Asset Status** will change to *Renewed* on the Asset Line Item. New Billing Schedules will be created based on the new **Start Date** and **End Date**.

When you change an asset, you can update values for fields such as the Net Price and Selling Term. Conga Billing, by default assigns the Start Date of the Asset Line Item to the effective date of the change.

Every changed or amended order has a new set of Billing Schedules that Conga Billing creates automatically and these new schedules depend on the Start and End dates that you define for the changed asset line item. When you change an asset, it is critical to align the Bill Cycle Start Date with the Billing Preference of the asset line item.

Let us now understand the impact of each asset action on Billing Schedules and how you can manage this change.

In the following section:

- Changing Billing Schedules When You Increase the Net Price of Assets
- Changing Billing Schedules When You Reduce the End Date of Assets

Impact of Asset Actions on Billing Schedules

If you use Conga Billing, you must consider the impact of each asset action, **Renew**, **Change**, **Swap** or **Terminate** has on the Billing and Invoicing of the purchased products. While the impact of asset actions on Billing Schedules are detailed in this guide, you must also consider the following definitions before you make any changes to an asset.

Field	Definition
Period Start Date	The date you want to start billing your customer for this purchase.
Period End Date	The date you want to stop billing your customer for this purchase.
Proration Period Treatment	The way of calculating the proration period. It must be set to Separate Period.
Billing Rule	The rule you defined to either **Bill in Advance** or **Bill in Arrears.**
Billing Plan	• **Periodic** - to bill your customer on fixed billing dates that represent custom billing periods that may or may not have a regular frequency. • **Milestone** - to distribute the total amount to be billed over multiple billing dates. As each milestone is successfully reached, you can bill your customer either a percentage of the entire cost or a pre-defined amount.
Billing Day of the Month	If you have configured **Billing Preference** with Account Billing Day of Month, select the day of the month when you want to generate the bill for this account. If you have configured **Billing Cycle Start** = Billing Day of Month, this field cannot be null.

Field	Definition
Billing Cycle Start	Select one of the following options: • **Billing Day of the Month** - Select this option to start the billing cycle on the billing day of the month. • **Period Start Date** - Select this option to enable Conga Billing to inherit this value from CPQ. • **Account Billing Day of the Month** - Select this option if you want to define the Billing Preference to use the Billing Day of the Month that you have defined on the Accounts page.

Scroll down to the Billing Schedule section on the Asset Line Item detail page to view the Billing and Usage schedules for that asset.

Billing Schedules for Renewed Assets

To define the **Renewal Date** on the Confirm Renewal page, select from one of the following options.

- **Contract End Date** - if you want to renew the asset on the date the contract expires.
- **Asset End Date** - if you want to renew the asset on the date the asset expires.
- **Farthest End Date** - if you want to choose the farthest end date out of all the selected Asset Line Items and apply this date to all the Asset Line Items.
- **Renewal Date** - use the calendar to define a custom date to renew the asset.

The date you enter here determines the billing period for the Renewed asset. The new Billing Schedules for a renewed asset are in the Pending Billing state by default. Each renewed asset has such Billing schedules that CPQ creates automatically when the asset is activated.

Billing Schedules for Cancelled Assets

When you cancel an asset, the cancellation is effective only after one day's time.

Example 1

You are a customer service representative for a telecommunications company and you cancel a customer's data subscription plan mid-cycle when the status of each Billing Schedule is **Pending Billing**.

Before you cancel the subscription, you see the following Billing Schedules.

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Pending Billing	100.00	
BS2	2/1/2015	2/28/2015	Pending Billing	100.00	
BS3	3/1/2015	3/31/2015	Pending Billing	100.00	
BS4	4/1/2015	4/30/2015	Pending Billing	100.00	

After you cancel the subscription, you see that status of schedules changes.

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Pending Billing	100.00	
BS2	2/1/2015	2/28/2015	**Superseded**	100.00	**Yes**
BS5	**2/1/2015**	**2/14/2015**	**Pending Billing**	**50.00**	
BS6	**2/15/2015**	**2/28/2015**	**Cancelled**	**50.00**	
BS3	3/1/2015	3/31/2015	**Cancelled**	100.00	
BS4	4/1/2015	4/30/2015	**Cancelled**	100.00	

In the above example, because the you cancelled the subscription mid-cycle, the original Billing Schedule is superseded and replaced by the following new Billing Schedules.

- BS1 is for the partial period that is still active and has not been billed.
- BS2 is for the partial period that has been cancelled.

Also, because the Billing Schedules for March and April are unbilled their status is set to *Cancelled.*

Example 2

Consider another example where you want to cancel a customer's data subscription plan mid-cycle when the status of some Billing Schedules is **Pending Billing** and others is **Invoiced**.

Before you cancel the subscription, you see the following Billing Schedules.

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	100.00	
BS2	2/1/2015	2/28/2015	Invoiced	100.00	
BS3	3/1/2015	3/31/2015	Invoiced	100.00	
BS4	4/1/2015	4/30/2015	Pending Invoiced	100.00	
BS5	5/1/2015	5/31/2015	Pending Billing	100.00	

After you cancel the subscription, you see that status of schedules changes.

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	100.00	
BS2	2/1/2015	2/28/2015	Invoiced	100.00	**Yes**
BS6	**2/15/2015**	**2/28/2015**	**Cancelled**	**50.00**	
BS7	**2/15/2015**	**2/28/2015**	**Pending Billing**	**-50.00**	
BS3	3/1/2015	3/31/2015	Invoiced	100.00	**Yes**
BS8	**3/1/2015**	**3/31/2015**	**Pending Billing**	**-100.00**	
BS4	4/1/2015	4/30/2015	**Cancelled**	100.00	

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS5	5/1/2015	5/31/2015	**Cancelled**	100.00	

In the second example, because you cancelled the subscription mid-cycle, the Billing Schedule for February is invoiced and is augmented with two new Billing Schedules.

- BS1 is for the partial period that was cancelled and is used to record that portion of the invoiced amount that was cancelled.
- BS2 is also for the partial period that was cancelled and it is used to generate an Invoice Line Item in order to reflect the credit for the portion of the invoiced amount that was cancelled.

Because the entire period for March is already *invoiced* it augments a single Billing Schedule that is used to generate an Invoice Line Item that then reflects the credit for the invoiced amount was cancelled. The Billing Schedules for April and March are *unbilled* their status is changed to *Cancelled*.

Billing Schedules for Incremented Assets

As a Sales rep, you can increase the quantity of an asset and bill your customer for the incremented asset as a separate asset. Separate billing schedules are created for the incremented asset.

Use Case: Creating Billing Schedules for an Incremented Asset

Description: This use case describes how billing schedules are created when you increase the quantity of an asset.

Suppose you are a customer service representative for a software company and you change the license quantity for an asset from 1 to 4. Before you change the asset quality, you see the following billing schedules:

Schedule	Period Start	Period End	Status	Fee Amount
BS1	1/1/2019	3/31/2019	Pending Billing	$90.00
BS2	4/1/2019	6/30/2019	Pending Billing	$90.00

Schedule	Period Start	Period End	Status	Fee Amount
BS3	7/1/2019	9/30/2019	Pending Billing	$90.00
BS4	10/1/2019	12/31/2019	Pending Billing	$90.00

To create billing schedules for the incremented assets

1. From an ABO Quote, click **Configure Products.**
2. On the cart page, click **Installed Products.**
3. Select your asset and click Change > Quantity.
4. In the **Change quantity by** field, increase the quantity by 3 by clicking the + icon. The **New Quantity** field displays the updated quantity immediately.
5. Select the appropriate **Cotermination** option. For details on Cotermination option, see Managing Asset Increment with Coterminate Lines.
6. Enter the Start Date and End Date.
7. Click **Go to Pricing**. An asset line item with status Incremented and quantity as 3 is created.
8. Finalize the Cart.
9. Accept the Quote/Proposal and activate the order.

Result:

- A new asset is created with Quantity as 3 and status as Incremented.
- 3 Billing Schedules are created as follows:

Schedule	Period Start	Period End	Status	Fee Amount
BS5	1/1/2019	3/31/2019	Pending Billing	$270.00
BS6	4/1/2019	6/30/2019	Pending Billing	$270.00
BS7	7/1/2019	9/30/2019	Pending Billing	$270.00

Usage Schedules for Cancelled Assets

Example 1

You are a customer service representative for a telecommunications company and you cancel a customer's usage based subscription plan mid-cycle when the status of each Usage Schedule is **Pending Billing**.

Before you cancel the subscription, the asset line item has the following Billing Schedules.

Billing Schedule	Period Start	Period End	Status	Usage Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	88.00	
BS2	2/1/2015	2/28/2015	Pending Billing	72.00	
BS3	3/1/2015	3/31/2015	Pending Billing	0.00	
BS4	4/1/2015	4/30/2015	Pending Billing	0.00	

Before you cancel the subscription, the asset line item has the following Usage Schedules.

Usage Schedule	Period Start	Period End	Status	Billing Schedule ID	Quantity	Superseded
US1	1/1/2015	1/31/2015	Pending Billing	BS1	30	
US2	2/1/2015	2/28/2015	Pending Billing	BS2	26	
US3	3/1/2015	3/31/2015	Pending Billing	BS3	0	
US4	4/1/2015	4/30/2015	Pending Billing	BS4	0	

After you cancel the subscription, the status of Billing Schedules change.

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	100.00	
BS2	2/1/2015	2/28/2015	**Superseded**	72.00	**Yes**
BS5	**2/1/2015**	**2/21/2015**	**Pending Billing**	**52.50**	
BS6	**2/22/2015**	**2/28/2015**	**Cancelled**	**19.50**	
BS3	3/1/2015	3/31/2015	**Cancelled**	0.00	
BS4	4/1/2015	4/30/2015	**Cancelled**	0.00	

After you cancel the subscription, the status of Usage Schedules change.

Usage Schedule	Period Start	Period End	Status	Billing Schedule ID	Quantity	Superseded
US1	1/1/2015	1/31/2015	Pending Billing	BS1	30	
US2	2/1/2015	2/28/2015	**Superseded**	BS2	26	**Yes**
US5	**2/1/2015**	**2/21/2015**	**Pending Billing**	**BS5**	**17**	
US6	**2/22/2015**	**2/28/2015**	**Cancelled**	**BS6**	**9**	
US3	3/1/2015	3/31/2015	**Cancelled**	BS3	0	
US4	4/1/2015	4/30/2015	**Cancelled**	BS4	0	

In the above example, because you cancelled the subscription mid-cycle, and Usage Schedule for the February is unbilled, the status is marked as *Superseded* and augmented be with the following Usage Schedules.

- BS1 is for the partial period that is still active and has not been *billed*. It reflects the aggregate amount of the *rated* Usage Inputs that have a date greater than or equal to 2/1 and less than or equal 2/21.
- US1 is for the partial period that is still active and has not been *billed*. It reflects the aggregate quantity of the *rated* Usage Inputs that have a date greater than or equal to 2/1 and less than or equal 2/21.
- BS2 is for the partial period that has been cancelled. It reflects the aggregate amount of the *rated* Usage Inputs that have a date greater than or equal to 2/22 and less than or equal 2/28.
- US2 is for the partial period that you cancelled. It reflects the aggregate quantity of the *rated* Usage Inputs that have a date greater than or equal to 2/22 and less than or equal 2/28.

> **Note**
>
> Every usage based asset that you terminate or change, also has Billing Schedules associated with that asset. The Usage Schedules are updated in a manner similar to the Billing Schedules.

Example 2

Consider another example where you want to cancel a customer's subscription plan mid-cycle when the status of some Usage Schedules is **Pending Billing** and others is **Invoiced**.

Before you cancel the subscription, the asset line item has the following Billing Schedules.

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	88.00	
BS2	2/1/2015	2/28/2015	Invoiced	72.00	
BS3	3/1/2015	3/31/2015	Invoiced	78.00	

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS4	4/1/2015	4/30/2015	Pending Billing	66.00	

Before you cancel the subscription, the asset line item has the following Usage Schedules.

Usage Schedule	Period Start	Period End	Status	Billing Schedule ID	Quantity	Superseded
US1	1/1/2015	1/31/2015	Pending Billing	BS1	30	
US2	2/1/2015	2/28/2015	Pending Billing	BS2	26	
US3	3/1/2015	3/31/2015	Pending Billing	BS3	31	
US4	4/1/2015	4/30/2015	Pending Billing	BS4	24	

After you cancel the subscription, you see that status of Billing Schedules change.

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	100.00	
BS2	2/1/2015	2/28/2015	Invoiced	72.00	**Yes**
BS5	**2/1/2015**	**2/28/2015**	**Pending Billing**	**-72.00**	
BS6	**2/1/2015**	**2/22/2015**	**Pending Billing**	**52.50**	
BS7	**2/22/2015**	**2/28/2015**	**Cancelled**	**19.50**	
BS3	3/1/2015	3/31/2015	Invoiced	78.00	**Yes**

Billing Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS8	**3/1/2015**	**3/31/2015**	**Pending Billing**	**-78.00**	
BS4	4/1/2015	4/30/2015	**Cancelled**	66.00	

After you cancel the subscription, you see that status of Usage Schedules change.

Usage Schedule	Period Start	Period End	Status	Billing Schedule ID	Quantity	Superseded
US1	1/1/2015	1/31/2015	Pending Billing	BS1	30	
US2	2/1/2015	2/28/2015	**Invoiced**	BS2	26	**Yes**
US5	**2/1/2015**	**2/21/2015**	**Pending Billing**	**BS6**	**17**	
US6	**2/22/2015**	**2/28/2015**	**Cancelled**	**BS7**	**9**	
US3	3/1/2015	3/31/2015	**Invoiced**	BS3	31	**Yes**
US4	4/1/2015	4/30/2015	**Cancelled**	BS4	24	

In the second example, because you cancelled the subscription mid-cycle, the Billing Schedule for February is invoiced and is augmented with two new Billing Schedules.

- BS1 is a credit for the amount that was *invoiced*.
- BS2 is for the partial period that was not cancelled and is used to charge for the portion of the invoiced amount was billed. It reflects the aggregate amount of the *rated* Usage Inputs that have a date greater than or equal to 2/1 and less than or equal 2/21.
- US1 is for the partial period that was not cancelled and is used to record (audit) that portion of the quantity amount was billed. It reflects the aggregate quantity of the *rated* Usage Inputs that have a date greater than or equal to 2/1 and less than or equal 2/21.

- BS3 is for the partial period that was cancelled and the amount the Account will not be charged for. It reflects the aggregate amount of the *rated* Usage Inputs that have a date greater than or equal to 2/22 and less than or equal 2/28.
- US2 is also for the partial period that has been cancelled. It reflects the aggregate quantity of the *rated* Usage Inputs that have a date greater than or equal to 2/22 and less than or equal 2/28.

Because the entire period for March is already *invoiced* it augments a single Billing Schedule that is used to generate an Invoice Line Item that then reflects the credit for the invoiced amount was cancelled. The Billing Schedules for April and March are *unbilled* their status is changed to *Cancelled*.

Changing Billing Schedules When You Decrease the Net Price of Assets

Example 1

You are a customer service representative for a software company and you decrease the Net Price of the customer's software subscription plan mid-cycle when the status of each Billing Schedule is **Pending Billing**.

Before you decrease the Net Price of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Pending Invoiced	100.00	
BS2	5/1/2015	5/31/2015	Pending Invoiced	100.00	
BS3	6/1/2015	6/30/2015	Pending Billing	100.00	

After you decrease the Net Price of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	**Superseded**	100.00	**Yes**

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS4	**4/1/2015**	**4/15/2015**	**Pending Billing**	**50.00**	
BS5	**4/16/2015**	**4/30/2015**	**Pending Billing**	**25.00**	
BS2	5/1/2015	5/31/2015	**Superseded**	100.00	**Yes**
BS6	**5/1/2015**	**5/31/2015**	**Pending Billing**	**50.00**	
BS3	6/1/2015	6/30/2015	**Superseded**	100.00	**Yes**
BS7	**6/1/2015**	**6/30/2015**	**Pending Billing**	**50.00**	

Because the amendment is *mid-cycle* and the April Billing Schedule is unbilled it will be *superseded* and *augmented* with the following new Billing Schedules.

- BS1 reflects the amount (at the old rate) to charge for the partial period before the amendment date.
- BS2 reflects the decreased amount for the partial period that was amended.

Also, the Billing Schedules for May and June are *unbilled* and are *superseded* and replaced by Billing Schedules that reflect the decreased amount.

Example 2

Consider another example where for a software company and you decrease the Net Price of the customer's software subscription plan mid-cycle when the status of some Billing Schedules is **Pending Billing** and others are **Invoiced**.

Before you decrease the Net Price of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	100.00	
BS2	5/1/2015	5/31/2015	Invoiced	100.00	
BS3	6/1/2015	6/30/2015	Pending Billing	100.00	

After you decrease the Net Price of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	100.00	**Yes**
BS4	**4/16/2015**	**4/30/2015**	**Pending Billing**	**-50.00**	
BS5	**4/16/2015**	**4/30/2015**	**Pending Billing**	**25.00**	
BS2	5/1/2015	5/31/2015	Invoiced	100.00	**Yes**
BS6	**5/1/2015**	**5/31/2015**	**Pending Billing**	**-50.00**	
BS3	6/1/2015	6/30/2015	**Superseded**	100.00	**Yes**
BS7	**6/1/2015**	**6/30/2015**	**Pending Billing**	**50.00**	

Because the amendment is *mid-cycle* and the April Billing Schedule is already invoiced it remains as invoiced and is *augmented* with the following new Billing Schedules.

- BS1 reflects the amount (at the old rate) that was invoiced for the partial period at/after the amendment date and will appear as a *credit* when an Invoice is generated.
- BS2 reflects the decreased amount for the partial period that was amended and will appear as a *charge* when an Invoice is generated.

Also, the Billing Schedule for May has been invoiced it will remain as invoiced and will be augmented with one new Billing Schedule. This new Billing Schedule contains the new amount to charge for and appears on the next Invoice generated for that Asset. The Billing Schedule for June is then unbilled (in draft state) it is superseded and replaced by a Billing Schedule that reflects the decreased amount.

Changing Billing Schedules When You Change the Billing Frequency of Assets

Example 1

You are a customer service representative for a software company and you must change the Billing frequency from Quarterly to Monthly for a customer's software subscription plan, mid-cycle when the status of each Billing Schedule is **Pending Billing**.

Before you change the Billing frequency of the subscription from Quarterly to Monthly, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	7/1/2015	9/30/2015	Invoiced	90.00	
BS2	10/1/2015	12/31/2015	Invoiced	90.00	
BS3	1/1/2016	3/31/2016	Pending Billing	90.00	

After you extend the End Date of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	7/1/2015	9/30/2015	Invoiced	90.00	**Yes**
BS4	**8/1/2015**	**9/30/2015**	**Pending Billing**	**-60.00**	
BS5	**8/1/2015**	**8/31/2015**	**Pending Billing**	**20.00**	
BS6	**9/1/2015**	**9/30/2015**	**Pending Billing**	**20.00**	
BS2	10/1/2015	12/31/2015	Invoiced	90.00	**Yes**
BS7	**10/1/2015**	**12/31/2015**	**Pending Billing**	**-90.00**	
BS8	**10/1/2015**	**10/31/2015**	**Pending Billing**	**20.00**	

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS9	**11/1/2015**	**11/30/2015**	**Pending Billing**	**20.00**	
BS10	**12/1/2015**	**12/31/2015**	**Pending Billing**	**20.00**	
BS3	1/1/2016	3/30/2016	**Superseded**	90.00	**Yes**
BS11	**1/1/2016**	**1/31/2016**	**Pending Billing**	**20.00**	
BS12	**2/1/2016**	**2/29/2016**	**Pending Billing**	**20.00**	
BS13	**3/1/2016**	**3/31/2016**	**Pending Billing**	**20.00**	
BS14	**4/1/2016**	**4/30/2016**	**Pending Billing**	**20.00**	

ⓘ The new *Term* is now 8/1/2015 – 4/30/2016 (9 months) and the new Net Price is 180.00, the Fee amount is 20.00 per month. When you transition from a less frequent Billing Frequency (like quarterly) to more a frequent Billing Frequency (like monthly), a *credit* Billing Schedule is created for each *whole period* Billing Schedule that is invoiced.

Because the billing frequency change is mid-cycle and the July-September Billing Schedule has been invoiced it will remain as invoiced and will be *augmented* with the following new Billing Schedules.

- BS4 reflects the amount for the partial period that is being amended (at the original billing frequency and rate) and will appear as a *credit* when an Invoice is generated.
- BS5 reflects the amount to charge for the first month within the amended partial period of the quarter.
- BS6 reflects the amount to charge for the second month within the amended partial period of the quarter.

Also, the October-December Billing Schedule has been invoiced it will remain as invoiced and will be *augmented* with the following new Billing Schedules.

- BS7 is a credit for the amount that was invoiced.
- BS8 reflects the amount to charge for the first month within the 2nd quarter.
- BS9 reflects the amount to charge for the second month within the 2nd quarter.
- BS10 reflects the amount to charge for the third month within the 2nd quarter.

The January-March Billing Schedule is unbilled and is marked as *Superseded* and augmented with the following new Billing Schedules.

- BS11 reflects the amount to charge for the first month within the 3rd quarter.
- BS12 reflects the amount to charge for the second month within the 3rd quarter.
- BS13 reflects the amount to charge for the third month within the 3rd quarter.

The End Date of the Asset Line Item is extended by 1 month to 4/30/2016 an additional Billing Schedule, BS14 is created to accommodate the extension.

Example 2

Consider another example where you must change the Billing frequency from Monthly to Quarterly, for a customer's software subscription plan, mid-cycle when the status of some Billing Schedules is **Pending Billing** and others is **Invoiced**.

Before you change the Billing frequency of the subscription from Monthly to Quarterly, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	3/1/2015	3/31/2015	Invoiced	100.00	
BS2	4/1/2015	4/30/2015	Invoiced	100.00	
BS3	5/1/2015	5/31/2015	Invoiced	100.00	
BS4	6/1/2015	6/30/2015	Invoiced	100.00	
BS5	7/1/2015	7/31/2015	Invoiced	100.00	
BS6	8/1/2015	8/31/2015	Pending Invoiced	100.00	

After you extend the End Date of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	3/1/2015	3/31/2015	Invoiced	100.00	
BS2	4/1/2015	4/30/2015	Invoiced	100.00	**Yes**
BS7	**4/16/2015**	**4/30/2015**	**Pending Billing**	**-50.00**	
BS8	**4/16/2015**	**5/31/2015**	**Pending Billing**	**50.00**	

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS3	5/1/2015	5/31/2015	Invoiced	100.00	**Yes**
BS4	6/1/2015	6/30/2015	Invoiced	100.00	**Yes**
BS9	**6/1/2015**	**8/31/2015**	**Pending Billing**	**100.00**	
BS5	7/1/2015	7/31/2015	Invoiced	100.00	**Yes**
BS6	8/1/2015	8/31/2015	**Superseded**	100.00	**Yes**

ⓘ The new Term is now 4/16 – 8/31 (4 ½ months) and the new Net Price is 450.00, the Fee amount is 300.00 every quarter. The amendment was performed on 4/16 with the Billing Cycle Start defined as the Period Start Date and the Calendar Cycle Start is June, the 1st period is from 4/16 to 5/31. After this period the quarters is aligned with 6/1 so the next period is from 6/1 to 8/1.

Because the billing frequency change is mid-cycle and the April Billing Schedule has been invoiced it will remain as invoiced and will be augmented with one new Billing Schedule. The new Billing Schedule reflects the amount for the partial period that is being amended (at the original billing frequency and rate) and will appear as a credit when an Invoice is generated.

Because the billing frequency change goes into effect 4/16 and the 16th does not align with the Start Period Date (3/1) a partial period will be created for the 1st new quarter. The Billing Schedule for May has already been invoiced, remains as invoiced, and is superseded by BS8. BS9 is created with a quarterly billing frequency that is from 6/1/2015 - 8/31/2015. It supersedes the 3 Billing Schedules for the months of June, July, and August. The Fee Amount is 300.00 but, because you Invoiced the schedules for June and July, the Fee amount is reduced to 100.00.

Changing Billing Schedules When You Extend the End Date of Assets

Example 1

You are a customer service representative for a software company and you extend the End Date of a customer's software subscription plan mid-cycle when the status of each Billing Schedule is **Pending Billing**.

Before you extend the End Date of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	200.00	
BS2	5/1/2015	5/31/2015	Invoiced	200.00	
BS3	6/1/2015	6/30/2015	Pending Invoiced	200.00	

After you extend the End Date of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	200.00	**Yes**
BS4	**4/16/2015**	**4/30/2015**	**Pending Billing**	**-100.00**	
BS5	**4/16/2015**	**4/30/2015**	**Pending Billing**	**50.00**	
BS2	5/1/2015	5/31/2015	Invoiced	200.00	**Yes**
BS6	**5/1/2015**	**5/31/2015**	**Pending Billing**	**-100.00**	
BS3	6/1/2015	6/30/2015	**Superseded**	200.00	**Yes**
BS7	**6/1/2015**	**6/30/2015**	**Pending Billing**	**100.00**	

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS8	7/1/2015	7/31/2015	Pending Billing	100.00	
BS9	8/1/2015	8/31/2015	Pending Billing	100.00	
BS10	9/1/2015	9/15/2015	Pending Billing	50.00	

Because the amendment is *mid-cycle* and the April Billing Schedule has been invoiced it will remain as invoiced and *augmented* with two new Billing Schedules.

- BS1 reflects the overpaid amount for the partial period that was amended and will appear as a *credit* when an Invoice is generated.
- BS2 reflects the new amount for the partial period that was amended and will appear as a *debit* when an Invoice is generated.

Also, the Billing Schedule for May has been invoiced it will be augmented with one new Billing Schedule. This new Billing Schedule contains a credit for the overpaid amount that was invoiced and will appear on the next Invoice generated for the corresponding Asset. The Billing Schedule for June is unbilled it will be superseded and replaced by a Billing Schedule that reflects the new amount. The **End Date** was extended by 2.5 months and 3 new Billing Schedules are created with the appropriate *fee amount*. The last Schedule created is only for half a month.

Example 2

Consider another example where you extend the End Date of the customer's software subscription plan mid-cycle when the status of some Billing Schedules is **Pending Billing** and others is **Invoiced**.

Before you extend the End Date of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	100.00	
BS2	5/1/2015	5/31/2015	Invoiced	100.00	
BS3	6/1/2015	6/30/2015	Pending Billing	100.00	

After you extend the End Date of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	200.00	
BS2	5/1/2015	5/31/2015	Invoiced	200.00	**Yes**
BS4	**5/1/2015**	**5/31/2015**	**Pending Billing**	**-100.00**	
BS3	6/1/2015	6/15/2015	**Superseded**	100.00	**Yes**
BS5	**6/1/2015**	**6/30/2015**	**Pending Invoiced**	**100.00**	
BS6	**7/1/2015**	**7/31/2015**	**Pending Invoiced**	**100.00**	

Note

The End Date of the Asset Line Item was extended by 1.5 months and Net Price was set to 300.00 for the new Term (5/1/2015 – 7/31/2015). This reduces the Fee from $200 per month to $100.00 per month.

Because the amendment is at the period start and the May Billing Schedule has been invoiced it will remain as invoiced and will be augmented with one new Billing Schedule. The new Billing Schedule reflects the overpaid amount for the whole period that was amended and will appear as a credit when an Invoice is generated. The Billing Schedule for June is unbilled it will be superseded and replaced by a Billing Schedule that reflects the new amount.

Because the original Schedule for June was for a partial period and the End Date of the Asset Line Item is extended by 1.5 months, the replacement Billing Schedule period is extended to a full period. Extending the End date by a period of 1.5 months results in the following changes.
The partial period for June is extended to a full period and the fee amount is updated accordingly. A new Billing Schedule is created for July with the appropriate fee amount.

Changing Billing Schedules When You Increase the Net Price of Assets

Example 1

You are a customer service representative for a software company and you increase the Net Price of the customer's software subscription plan mid-cycle when the status of each Billing Schedule is **Pending Billing**.

Before you increase the Net Price of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	3/1/2015	3/31/2015	Pending Invoiced	100.00	
BS2	4/1/2015	4/30/2015	Pending Invoiced	100.00	
BS3	5/1/2015	5/31/2015	Pending Billing	100.00	
BS4	6/1/2015	6/30/2015	Pending Billing	100.00	

After you increase the Net Price of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	3/1/2015	3/31/2015	Pending Invoiced	100.00	
BS2	4/1/2015	4/30/2015	**Superseded**	100.00	**Yes**
BS5	**4/1/2015**	**4/15/2015**	**Pending Billing**	**50.00**	
BS6	**4/16/2015**	**4/30/2015**	**Pending Billing**	**100.00**	
BS3	5/1/2015	5/31/2015	**Superseded**	100.00	**Yes**
BS7	**5/1/2015**	**5/31/2015**	**Pending Billing**	**200.00**	
BS4	6/1/2015	6/30/2015	**Superseded**	100.00	**Yes**

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS8	**6/1/2015**	**6/30/2015**	**Pending Billing**	**200.00**	

Because the amendment is *mid-cycle* and the April Billing Schedule is unbilled it will be *superseded* and *augmented* with the following new Billing Schedules.

- BS1 reflects the amount (at the old rate) to charge for the partial period before the amendment date.
- BS2 reflects the increased amount for the partial period that was amended.

Also, the Billing Schedules for May and June are *unbilled* and are *superseded* and replaced by Billing Schedules that reflect the increased amount.

Example 2

Consider another example where for a software company and you increase the Net Price of the customer's software subscription plan mid-cycle when the status of some Billing Schedules is **Pending Billing** and others is **Invoiced**.

Before you increase the Net Price of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	100.00	
BS2	5/1/2015	5/31/2015	Invoiced	100.00	
BS3	6/1/2015	6/30/2015	Pending Invoiced	100.00	

After you increase the Net Price of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	100.00	**Yes**
BS4	**4/16/2015**	**4/30/2015**	**Pending Billing**	**-50.00**	

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS5	**4/16/2015**	**4/30/2015**	**Pending Billing**	**100.00**	
BS2	5/1/2015	5/31/2015	Invoiced	100.00	**Yes**
BS6	**5/1/2015**	**5/31/2015**	**Pending Billing**	**100.00**	
BS3	6/1/2015	6/30/2015	**Superseded**	100.00	**Yes**
BS7	**6/1/2015**	**6/30/2015**	**Pending Billing**	**200.00**	

Because the amendment is *mid-cycle* and the April Billing Schedule is already invoiced it remains as invoiced and is *augmented* with the following new Billing Schedules.

- BS1 reflects the amount (at the old rate) that was invoiced for the partial period at/after the amendment date and will appear as a *credit* when an Invoice is generated.
- BS2 reflects the increased amount for the partial period that was amended and will appear as a *charge* when an Invoice is generated.

Also, the Billing Schedule for May has been invoiced it will remain as invoiced and will be augmented with one new Billing Schedule. This new Billing Schedule contains the new amount to charge for and appears on the next Invoice generated for that Asset. The Billing Schedule for June is then unbilled (in draft state) it is superseded and replaced by a Billing Schedule that reflects the increased amount.

Changing Billing Schedules When You Reduce the End Date of Assets

Example 1

You are a customer service representative for a software company and you shorten the End Date of a customer's software subscription plan mid-cycle when the status of each Billing Schedule is **Pending Billing**.

Before you shorten the End Date of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Pending Invoiced	100.00	
BS2	5/1/2015	5/31/2015	Pending Invoiced	100.00	
BS3	6/1/2015	6/30/2015	Pending Invoiced	100.00	
BS4	7/1/2015	7/31/2015	Pending Billing	100.00	
BS5	8/1/2015	8/31/2015	Pending Billing	100.00	

After you shorten the End Date of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	**Superseded**	100.00	**Yes**
BS6	**4/1/2015**	**4/15/2015**	**Pending Billing**	**50.00**	
BS7	**4/16/2015**	**4/30/2015**	**Pending Billing**	**112.50**	
BS2	5/1/2015	5/31/2015	**Superseded**	100.00	**Yes**
BS8	**5/1/2015**	**5/31/2015**	**Pending Billing**	**225.00**	
BS3	6/1/2015	6/30/2015	**Superseded**	100.00	**Yes**
BS9	**6/1/2015**	**6/15/2015**	**Pending Billing**	**112.50**	
BS10	**6/15/2015**	**6/30/2015**	**Cancelled**	**50.00**	
BS4	7/1/2015	7/31/2015	**Cancelled**	100.00	
BS5	8/1/2015	8/31/2015	**Cancelled**	100.00	

Because the **End Date** of the Asset Line Item is reduced by 2.5 months and the **Net Price** is set to 450.00, the monthly fee for the amended Term(4/15 - 6/15) is increased from 100 per month to 225 per month.

Because the amendment is mid-cycle and the April Billing Schedule is unbilled, it is then superseded and replaced with two new Billing Schedules.

- BS1 reflects the amount at the old rate for the partial period prior to the amendment date and is calculated as a *debit* when an Invoice is generated.
- BS2 reflects the new amount for the partial period that was amended and is calculated as a *debit* when an Invoice is generated.

Also, the Billing Schedule for May is unbilled and is superseded with one new Billing Schedule. This new Billing Schedule reflects the new monthly rate and is calculated in the next Invoice that you generate for the corresponding Asset. The Billing Schedule for June is unbilled and its period is shortened, it is superseded and replaced with the following new Billing Schedules.

- The 1st new Billing Schedule reflects the new amount for the partial period that was amended and is calculated as a *debit* when you generate an Invoice.
- The period in June is shortened and the 2nd new Billing Schedule reflects the amount for the partial period of June that is *dropped* because the End Date is shortened.

The Billing Schedules for July and August are unbilled and they are *dropped* because the End Date is shortened they are marked as *Cancelled.*

Example 2

Consider another example where you shorten the **End Date** of the customer's software subscription plan, mid-cycle when the status of some Billing Schedules is **Pending Billing** and others is **Invoiced**.

Before you shorten the **End Date** of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	100.00	
BS2	5/1/2015	5/31/2015	Invoiced	100.00	
BS3	6/1/2015	6/30/2015	Invoiced	100.00	
BS4	7/1/2015	7/31/2015	Invoiced	100.00	

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS5	8/1/2015	8/31/2015	Pending Invoiced	100.00	

After you shorten the End Date of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	4/1/2015	4/30/2015	Invoiced	100.00	**Yes**
BS6	**4/16/2015**	**4/30/2015**	**Pending Billing**	**-50.00**	
BS7	**4/16/2015**	**4/30/2015**	**Pending Billing**	**112.50**	
BS2	5/1/2015	5/31/2015	Invoiced	100.00	**Yes**
BS8	**5/1/2015**	**5/31/2015**	**Pending Billing**	**125.00**	
BS3	6/1/2015	6/30/2015	Invoiced	100.00	**Yes**
BS9	**6/1/2015**	**6/15/2015**	**Pending Billing**	**-50.00**	
BS10	**6/1/2015**	**6/15/2015**	**Pending Billing**	**112.50**	
BS11	**6/15/2015**	**6/30/2015**	**Pending Billing**	**-50.00**	
BS4	7/1/2015	7/31/2015	Invoiced	100.00	**Yes**
BS12	**7/1/2015**	**7/31/2015**	**Pending Billing**	**-100.00**	
BS5	8/1/2015	8/31/2015	**Cancelled**	100.00	

Note

The End Date of the Asset Line Item is reduced by 2.5 months and the Net Price is set to 450.00, the monthly fee for the amended Term(4/15 - 6/15) is increased from 100 per month to 225 per month.

Because the amendment is mid-cycle and the April Billing Schedule has been invoiced it will remain as invoiced and will be augmented with the following new Billing Schedules.

- BS1 reflects the amount already paid for the partial period that was amended and will appear as a *credit* when an Invoice is generated.
- BS2 reflects the new amount for the partial period that was amended and will appear as a *debit* when an Invoice is generated.

Because the period in June is shortened, the 3rd new Billing Schedule reflects a credit for the partial period of June after the End Date is shortened.

Example 3

Consider another example where you shorten the **End Date** make it the same as the **Start Date** of the customer's software subscription plan mid-cycle to the same as the **Start Date**, when the status of some Billing Schedules is **Pending Billing** and others is **Invoiced**.

Before you shorten the **End Date** and make it the same as the **Start Date** of the subscription, you see the following Billing Schedules.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	100.00	
BS2	2/1/2015	2/28/2015	Invoiced	100.00	
BS3	3/1/2015	3/31/2015	Invoiced	100.00	
BS4	4/1/2015	4/30/2015	Pending Invoiced	100.00	

After you shorten the End Date of the subscription, the number of and the status of each Billing Schedule changes.

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS1	1/1/2015	1/31/2015	Invoiced	100.00	
BS2	2/1/2015	2/28/2015	Invoiced	100.00	**Yes**
BS5	**2/8/2015**	**2/21/2015**	**Pending Billing**	**-50.00**	

Schedule	Period Start	Period End	Status	Fee Amount	Superseded
BS6	**2/8/2015**	**2/21/2015**	**Pending Billing**	**80.00**	
BS7	**2/22/2015**	**2/28/2015**	**Pending Billing**	**-25.00**	
BS3	3/1/2015	3/31/2015	Invoiced	100.00	**Yes**
BS8	**3/1/2015**	**3/31/2015**	**Pending Billing**	**-100.00**	
BS4	4/1/2015	4/30/2015	**Cancelled**	100.00	

The new Term in this scenario is from 2/8/2015 to 2/21/2015 and the Net Price will be 80.00.

Because the amendment is *mid-cycle* and the February Billing Schedule has been invoiced it will remain as invoiced and because the *end date* of the Asset has been reduced to February it will be *augmented* with the following new Billing Schedules.

- BS1 reflects the amount already paid for the partial period that was amended and will appear as a *credit* when an Invoice is generated.
- BS2 reflects the new amount for the partial period that was amended and will appear as a *debit* when an Invoice is generated.
- BS3 is to account for the end of the period that is being *dropped* and will appear as a *credit* when an Invoice is generated.

The Billing Schedule for March is invoiced and is *removed* because the End Date is shortened.

A new Billing Schedule is created to credit the amount that was already invoiced. Because the End Date is shortened and the period for August is unbilled, the Billing Schedule is marked as *Cancelled.*

Managing Services

The Service CPQ feature caters to customer requirements that are related to services for existing assets (equipment purchased from Conga). As a Sales Representative, you can quote an asset that is sold to customers and associate a service product with assets. You have the visibility to the end-to-end cycle through the following pages for service management:

- Installed Products page

- Service Catalog page
- Service Config page
- Service Cart page

This section covers the following topics:

Associating Services with Assets

Prerequisites

- You must have existing assets.
- Your administrator must have completed certain configurations. For more information on these configurations, see Configuring Service CPQ.

To associate a service with an asset

1. Create a quote.
2. Navigate to the Installed Products page.
3. On the **Purchased Products** tab, select the required assets (standalone, bundle, or options).

> **Validating Service Assets based on Criteria**
> You can select multiple equipment assets on the **Purchased Products** tab. Based on the selection criteria of equipment assets, CPQ displays validation messages to ensure that the right set of equipment is selected to associate a service. For example, you can select assets based on the location criterion and then proceed to associate a service.

4. Click one of the following options:
    - Click **Relate**. The Service Catalog is displayed. For more information, see Working with the Service Catalog.
    - Click **Relate Component**. The Relate Component pop-up is displayed. Select the required options of the asset bundle and click **Relate**. The Service Catalog is displayed.

      > **Relate or Relate Component**
      > - Use the **Relate** flow to associate services to a standalone or bundle equipment. The Related Line Item for the services on the Cart display only the standalone or bundle equipment that you selected here on the **Purchased Products** tab.
      > - Use the **Relate Component** flow to associate services to components of a bundle equipment. The Related Line Item for the services on the Cart display the components of the bundle that you selected here on the **Purchased Products** tab.

5. Select the required services:
    - Click **Add to Cart**. The Service Cart is displayed. For more information, see Working with the Service Cart.
    - Click **Configure**. The Config page is displayed.
        i. Configure the attributes or options.
        ii. Click **Go to Pricing**. The Service Cart is displayed.
6. View the Service Cart. You can see the service line items and the related line items with weightage details for each asset.
7. Finalize the cart.
8. After your customer accepts the quote, an order is generated.
9. After you activate the order, new service assets are generated. The service assets are active and visible on the account.
10. Navigate to the Installed products page, select an equipment asset or service asset on the **Purchased Products** tab, and then click the **Related Purchases** tab. The association between equipment asset and service asset is displayed.
    - If you selected an equipment asset on the **Purchased Products** tab, the associated service asset is displayed on the **Related Purchases** tab.
    - If you selected a service asset on the **Purchased Products** tab, the associated equipment asset is displayed on the **Related Purchases** tab.

Working with the Service Catalog

The Service Catalog works similar to the Product Catalog. The Service Catalog displays all products for which your administrator has set **Product Type** as *Service*. The services on the Service Catalog are displayed based on how your administrator has configured the service categories into offering, service categories, service bundles, standalone services and option services. You can see and configure the service offerings (entitlements) that are applicable to your assets. CPQ disables the non-eligible entitlements for selection on the Service Catalog. If you select an entitlement that is not applicable to any asset, CPQ displays an error message.

> If your administrator has configured constraint rules for services, the Service Catalog displays only the eligible service products that you can associate with the selected assets. When you relate an asset bundle, the eligibility rule is triggered at the option and bundle level. Based on the eligible purchases for the option of a service, you can do the following:
>
> - Sell services for a bundle or option
> - Select the options for which you want to add a service coverage

CPQ supports the catalog actions on the Service Catalog also. For example, searching for services, configuring service bundles, adding services to the cart, and changing the quantity of services. The Service Catalog also displays product description, product code, and charge types beside services. However, CPQ does not support Favorites in the Service CPQ flow.

You can perform the following actions on the Service Catalog:

1. Search for a particular service. For more information, see Searching Products from the Catalog.
2. Select a service and click **Add to Cart** to add it directly to the Cart. For more information, see Working with the Service Cart.
3. Click **Configure** next to a service that has attributes or options associated with it. The Config page is displayed. For more information, see Configuring Service Products.
4. Click the **Selected Assets** link to view the assets and their corresponding details. The Selected Assets pop-up displays the assets that you selected on the **Purchased Products** tab on the Installed Products page.
5. Click **Update** if you navigated to the Installed Products page from the Service Cart and modified asset selection. For more information, see Working with the Service Cart. After the update is complete, click the **View Cart** in the Mini-Cart to go to the Service Cart.

> Before the progress bar is loaded completely, do not to click **Update** on the Service Catalog. If you click **Update** before the progress bar loading is complete, CPQ allows you to add a related line item, but CPQ does not redistribute the **Weightage Percentage** and **Weightage Amount** field values on the Cart when you view **Related Line Items**.

Configuring Service Products

The Service Configuration page works similar to Product Configuration page. For more information, see Configuring Products from the Catalog. After you navigate from the Service Catalog to the Service Config page, you can perform the following actions:

1. Click the **Selected Assets** link to view the assets and their corresponding details. The Selected Assets pop-up displays the assets that you selected on the Installed Products page.
2. Configure attributes for your service from the **Product Attributes** tab if attributes are associated to your service product.
3. Select and configure options for your bundle service from the **Product Options** tab if options are associated to your service product.
    a. Click the icon next to the selected option. The Manage *<Service_Option_Name>* page with the assets you selected on the Installed Products page is displayed.
    b. Select required options of the asset bundle to which you want to associate the service option and click **Save**. You are navigated back to the Config page.

> After you select a service option, the icon appears after the pricing is complete. This Manage *<Service_Option_Name>* page helps you to see which assets are related to the service option.
>
> - If you selected a standalone asset on the Installed Products page, only that asset is displayed on the Manage *<Service_Option_Name>* page. If you clear the selected asset, CPQ displays an error message. Click **Cancel** to close the page because you cannot perform any other action.
> - If you selected more standalone assets on the Installed Products page, those assets are displayed on the Manage *<Service_Option_Name>* page. Clear some selected assets to

which you do not want to associate the service option and click **Save**.
- If you selected a bundle asset on the Installed Products page, the asset bundle and all options of the bundle asset are displayed on the Manage *<Service_Option_Name>* page. Select required options of the asset bundle to which you want to associate the service option and click **Save**.
- If you selected some options of the bundle asset on the Installed Products page (Relate Component scenario), only those selected options are displayed on the Manage *<Service_Option_Name>* page. Clear some selected options of the asset bundle to which you do not want to associate the service option and click **Save**.

c. Click the  icon next to the selected option under **Itemized Options** to view the assets and their corresponding details. The Selected Assets pop-up displays the assets that you selected on the Installed Products page.

4. Click **Go to Pricing**. CPQ calculates the price of products on the Config page and navigates you to the Service Cart page. For more information, see Working with the Service Cart.

> If your administrator has enabled the **Bypass Shopping Cart** setting, the **Go to Pricing** button is hidden on the Config page. However, you can still finalize the quote using the Mini-Cart.

Working with the Service Cart

The Service Cart works similar to Product Cart. For more information, see Working with the Cart. CPQ supports the cart grid actions on the Service Cart as well. After you navigate from the Service Catalog or the Service Config page to the Service Cart page, you can perform the following actions:

1. Modify the related line items that are associated with the service:
   - Hover the mouse on the service bundle name, click the more icon () and select **Related Line Items**. The asset bundle that you selected on the Installed Products page is displayed.
   - Click the  icon to expand the service bundle and hover the mouse on a service option name, click the more icon () and select **Related Line Items**. The asset component(s) that you selected on the Installed Products page is displayed.

2. Click **Add/Remove** if you want to change the assets associated to a specific line item only (service bundle or service option). The Installed Products page is displayed.
    a. Remove the selected asset if you do not want to associate it with the current service.
    b. Add more or different assets if required and click **Relate**. The Service Catalog is displayed.
    c. Click **Update** to confirm the service product association with the new asset.
    d. Click **View Cart** on the Mini-Cart. The Service Cart is displayed.
3. Click **Add/Remove at all levels** if you want to change the assets associated to the service bundle and all service options together. The Installed Products page is displayed.
    a. Remove the selected assets if you do not want to associate them with the current services.
    b. Add more or different assets if required and click **Relate**. The Service Catalog is displayed.
    c. Click **Update** to confirm the service product association with the new asset.
    d. Click **View Cart** on the Mini-Cart. The Service Cart is displayed.

    > When you use **Add/Remove** and **Add/Remove at all levels** buttons to navigate to the Service Catalog, filters defined in Search Filter (CPQ) are not applied.

4. Distribute the weightage of the service among related line items:
    a. From the **Weightage Type** drop-down, select *Percentage* or *Amount*.
    b. In the **Weightage Amount** field, enter a value.
        - If the Weightage Type = Percentage, the total weightage must be equal to 100.
        - If the Weightage Type = Amount, the total weightage must be equal to the net price of the line item.
    c. Click **Save**.

    > The weightage amount or percentage value drives the service allocation for the customer. For example, you can set 60:40 service allocation amongst two service products depending upon service implementation needs.

> If your administrator has set the global flag **isTriggerInitiatedUpdate** to *True*, CPQ skips the trigger logic, which allocates the net price of the service line item to all associated related line item records based on the **Weightage Type** and **Weightage Amount** fields. For more information, see Configuring the Execution of the RelatedLineItem Trigger.

5. Click **Split** to clone a configured service line. For more information, see Splitting the Related Line Items from Services.
6. Finalize the cart.

Splitting the Related Line Items from Services

The Split feature allows you to clone a service with some or all related line items using in the Service Cart. The Split feature is available only at the service bundle level. You can clone the service to discount or negotiate various equipment items differently. This feature splits the related line items based on the split criteria defined by the administrator.

In many scenarios, Sales Representatives belonging to a specific region have identical needs for negotiation. However, there may be scenarios where the Sales Representatives may not want to break the service line.

Prerequisites

The split criteria must be defined in the **Service Line Split Criteria** setting in Config System Properties. You can define split criteria that apply globally or create split criteria for a specific flow. In case the split criteria is defined for a specific flow, the **Split** button is not displayed in other flows. For more information, see Defining Service Line Split Criteria.

To split related line items from services

Perform the following steps to split a related line item from a service:

1. On the Service Cart page, hover the mouse on the service bundle, click the more icon ( ) and select **Related Line Items**.
2. Click **Split**.
3. Select one or more criteria and click **Split**.

After you click **Split**, the service is cloned. The related line items are removed from the original service and associated to the cloned service based on the split criteria. The price is recalculated based on the related line items associated with the service.

Modifying the Association of Assets with Services by Launching the Installed Products Page Directly

You can launch the Installed Products page directly for the Add/Remove flow (Service CPQ), where you can modify the association of assets with services.

Prerequisite

Your administrator must have configured a custom button to launch the Installed Products page directly from the custom cart page. For example, your administrator has used servicePLN=2 on the custom button. For more information, see Configuring a Custom Button to Launch the Installed Products Page Directly for the Add/Remove Flow.

To modify the association of assets with services

1. Create a quote.
2. Click **Configure Products**. The Catalog page is displayed.
3. Click **Installed Products**. The Installed Products page is displayed.
4. Select the required assets that you want to associate with service products.
   Example:
   - Standalone Asset 1
   - Standalone Asset 2
5. Click **Relate**. The Service Catalog is displayed.
6. Click **Add to Cart** for the service products that you want to associate with the selected assets.
   Example:
   - Service Product 1
   - Service Product 2
7. On the Mini-Cart, **View Cart**. The Service Cart page is displayed.
8. Finalize the cart (or save it). CPQ finalizes the cart and navigates you back to the quote.
9. Click the custom button (where the administrator has used the service parameter, servicePLN=2). CPQ launches the Installed Products page directly with Standalone Asset 1 and Standalone Asset 2 selected by default. You can see the service parameter in the URL now. On the Mini-Cart, you can see Service Product 1 Service Product 2.

10. Modify the association of assets with Service Product 2:
    - Add more assets.
    - Remove the assets you will no longer associate with the service.
11. Click **Relate**. The Service Catalog with the service product that you passed in the parameter is displayed. In this case, Service Product 2 is displayed.
12. Click **Update**.

Use Case: Bundle to Bundle

Description

This use case describes how to associate a service bundle with an asset bundle. The following table lists a sample asset bundle and a service bundle.

Asset Bundle 1	Service Bundle 1
Asset Bundle Option 1	Service Bundle Option 1
Asset Bundle Option 2	Service Bundle Option 2
Asset Bundle Option 3	Service Bundle Option 3
Asset Bundle Option 4	Service Bundle Option 4

Association

1. Create a quote.
2. Navigate to the Installed Products page.
3. On the **Purchased Products** tab, select **Asset Bundle 1**.
4. Click **Relate**. The Service Catalog is displayed.
5. Select **Service Bundle 1** and click **Configure**. The Config page is displayed.
6. Click **Go to Pricing**. The Service Cart is displayed.
7. Hover the mouse on **Service Bundle 1**, click the more icon (⋮) and select **Related Line Items**. You can see **Asset Bundle 1**.
8. Finalize the cart.
9. After your customer accepts the quote, an order is generated.
10. After you activate the order, new service asset **Service Bundle 1** is generated. The service asset is active and visible on the account.
11. Navigate to the Installed products page. You can see the association between the service asset and equipment asset on the **Purchased Products** tab and the **Related Purchases** tab.

Use Case: Bundle to Components

Description

This use case describes how to associate a service bundle with components of an asset bundle. The following table lists a sample asset bundle and a service bundle.

Asset Bundle 1	Service Bundle 1
Asset Bundle Option 1	Service Bundle Option 1
Asset Bundle Option 2	Service Bundle Option 2
Asset Bundle Option 3	Service Bundle Option 3
Asset Bundle Option 4	Service Bundle Option 4

Association

1. Create a quote.
2. Navigate to the Installed Products page.
3. On the **Purchased Products** tab, select **Asset Bundle 1**.
4. On the **Relate** drop-down, click **Relate Component**. The Relate Component pop-up is displayed.
5. Select **Asset Bundle Option 1** and **Asset Bundle Option 2**, and click **Relate**. The Service Catalog is displayed.
6. Select **Service Bundle 1** and click **Configure**. The Config page is displayed.
7. Select **Service Bundle Option 1** and **Service Bundle Option 2**.
8. Click **Go to Pricing**. The Service Cart is displayed.
9. Hover the mouse on **Service Bundle 1**, click the more icon (⋮) and select **Related Line Items**. You can see **Asset Bundle Option 1** and **Asset Bundle Option 2**.
10. Click the  icon to expand **Service Bundle 1** and hover the mouse on **Service Bundle Option 1**, click the more icon (⋮) and select **Related Line Items**. You can see **Asset Bundle Option 1** and **Asset Bundle Option 2**.
11. Click the  icon to expand **Service Bundle 1** and hover the mouse on **Service Bundle Option 2**, click the more icon (⋮) and select **Related Line Items**. You can see **Asset Bundle Option 1** and **Asset Bundle Option 2**.
12. Finalize the cart.
13. After your customer accepts the quote, an order is generated.

14. After you activate the order, new service assets **Service Bundle 1**, **Service Bundle Option 1**, and **Service Bundle Option 2** are generated. The service assets are active and visible on the account.
15. Navigate to the Installed products page. You can see the association between the service assets and equipment assets on the **Purchased Products** tab and the **Related Purchases** tab.

Use Case: Bundle to Bundle and Components

Description

This use case describes how to associate a service bundle with an asset bundle and its components.

> ⓘ You can associate a service bundle or a service option with either an equipment asset bundle or an equipment asset option at a time, but not with both.

The following table lists a sample asset bundle and a service bundle.

Asset Bundle 1	Service Bundle 1
Asset Bundle Option 1	Service Bundle Option 1
Asset Bundle Option 2	Service Bundle Option 2
Asset Bundle Option 3	Service Bundle Option 3
Asset Bundle Option 4	Service Bundle Option 4

Association

1. Create a quote.
2. Navigate to the Installed Products page.
3. On the **Purchased Products** tab, select **Asset Bundle 1**.
4. Click **Relate**. The Service Catalog is displayed.
5. Select **Service Bundle 1** and click **Configure**. The Config page is displayed.
6. Select **Service Bundle Option 1**.
    a. Click the  icon next to it. The Manage Service Bundle Option 1 page with the following assets is displayed.
        - Asset Bundle 1
        - Asset Bundle Option 1
        - Asset Bundle Option 2

        - Asset Bundle Option 3
        - Asset Bundle Option 4
    b. Click the checkbox next to **Asset Bundle 1** until you clear the selection.
    c. Select **Asset Bundle Option 1** and click **Save**. The Config page is displayed.
7. Select **Service Bundle Option 2**.
    a. Click the icon next to it. The Manage Service Bundle Option 2 page with the following assets is displayed.
        - Asset Bundle 1
        - Asset Bundle Option 1
        - Asset Bundle Option 2
        - Asset Bundle Option 3
        - Asset Bundle Option 4
    b. Click the checkbox next to **Asset Bundle 1** until you clear the selection.
    c. Select **Asset Bundle Option 2** and click **Save**. The Config page is displayed.
8. Click **Go to Pricing**. The Service Cart is displayed.
9. Hover the mouse on **Service Bundle 1**, click the more icon ( ) and select **Related Line Items**. You can see **Asset Bundle 1**.
10. Click the icon to expand **Service Bundle 1** and hover the mouse on **Service Bundle Option 1**, click the more icon ( ) and select **Related Line Items**. You can see **Asset Bundle Option 1**.
11. Click the icon to expand **Service Bundle 1** and hover the mouse on **Service Bundle Option 2**, click the more icon ( ) and select **Related Line Items**. You can see **Asset Bundle Option 2**.
12. Finalize the cart.
13. After your customer accepts the quote, an order is generated.
14. After you activate the order, new service asset **Service Bundle 1** with service options is generated. The service asset is active and visible on the account.
15. Navigate to the Installed products page. You can see the association between the service asset and equipment asset on the **Purchased Products** tab and the **Related Purchases** tab.

Use Case: A Product with Parent and Child Price Lists

This use case describes how the Catalog page and the Cart page show different price lists when a product is associated with both parent and child price lists.

1. Create a price list ABC Group (this is the parent price list).
2. Create another price list ABC Group North (this is a child price list where **Based on Price List** = ABC Group).
3. Create a product ABC P-1000 (this product is associated with both parent and child price lists).
4. Create a quote with the child price list (ABC Group North).
5. Navigate to the Catalog page and search for the product ABC P-1000. The price of this product on the Catalog page is derived from the parent price list ABC Group.

ⓘ Though you have created the quote with the child price list, the Product Configuration has the parent price list. Therefore, CPQ is deriving the price of the product from the parent price list. After the pricing engine is executed, CPQ derives the correct price for the product on the Cart page, which is from the child price list.

Glossary

This section covers the following topics:

- Approval Stages
- Proposal Actions
- New Related Lists

Approval Stages

This section describes the purpose of the different Approval Stages.

Field Value	Description
Draft	New Quote Created.
Approval Required	Changes to the Quote or Quote Line Items have triggered a need for approval.
In Review	Quote is in the process of being Approved or Denied.
Approved	Quote has been approved by management.
Generated	Quote document has been generated.

Field Value	Description
Presented	Quote document was presented to the end customer, either via email or manually.
Accepted	Quote was accepted by the end customer.
Denied	Quote has been denied by management.

Proposal Actions

This section describes the actions available at various stages of the proposal and their significance.

> These actions are fields and they are configurable by the System Administrator per your business process requirements.

Configure Products

Action	
Description	Starts a new configuration process for the Quote, or reconfigure an existing set of products on the Quote.
Availability	If approval stage is *Draft*.

Generate Proposal

Action	
Description	Generates a Proposal document and attaches it to the Quote/Proposal record.
Availability	Always Available if Ready to Generate is set to *true*.

Present Proposal

Action	
Description	Presents the attached document to the Quote to the end customer through an email.
Availability	Always Available if Ready to Present is set to *true*.

Accept Proposal

Action	
Description	Accept refers to an end customer *Accepting* the final proposal. This button synchronizes all Summary lines with the Opportunity Products for the Related Opportunity.
Availability	If approval stage is *Presented*.

Make Primary

Action	
Description	Sets this Quote to primary and all other Quotes for the Opportunity non Primary. Also enables the Synchronize with Opportunity button.
Availability	If Primary flag is *false*.

Synchronize with Opportunity

Action	
Description	Synchronizes the finalized quote with the Opportunity, each time the cart is Finalized. If you are using automatic synchronization, you should remove this from the page layout.
Availability	If Primary flag is *true*.

Create Agreement with Line Items

Action	
Description	Once accepted, the final step is to create the contract.
Availability	If approval stage is *Accepted*.

New Related Lists

This section describes the various related lists enabled after the draft proposal record is created.

Name	Description
Actions	Lists the various actions you can execute for this stage of the quoting process. A list of actions with the description of their function is available in Proposal Actions.
Line Items	Contains the detailed line items (bundles, options, and standalone) of a finalized configuration. This corresponds to the Detail tab in configuration.
Summary	Contains the summary view (standalone and bundles) of a finalized configuration. This corresponds to the Summary tab in configuration.
Summary Groups	Contains the totals for the finalized configuration according to the product hierarchy including a Grand Total.
Notes & Attachments	Enables you to attach any reference document or file relevant to a quote/proposal. The proposal document when generated from the template is stored by the application in this section.
Open Activities	Enables you to create and assign new task corresponding to a quote.

Name	Description
Approval History	Initiate and track approvals corresponding to the quote. Note: Approvals can be initiated only if the corresponding approval processes have been set up by the System Administrator.
Activity History	Lists an Audit history of significant changes during the life of the quote.

CPQ for SOAP API Developers

This section explains the SOAP APIs provided by Conga Configuration, Pricing, and Quoting (CPQ).

Topic	Description
What's Covered	This section walks the API developers through the list of SOAP APIs provided by Conga.
Primary Audience	• API developers
IT Environment	Refer to the latest *Conga CPQ Release Notes* for information on System Requirements and Supported Platforms.
Updates	For a comprehensive list of updates to this section for each release, see the What's New in CPQ Documentation topic.
Other Resources	Refer to *Conga CPQ Release Notes* for information on system requirements and supported platforms, new features and enhancements, resolved issues, and known issues for a specific release.

This section describes the following topics:

- CPQ Web Service (Apttus_Config2)
- CPQ Web Service(Apttus_CPQApi)
- Batch Update Service
- Quote/Proposal Config Web Service
- Favorite Configuration Global Service
- Quote Collaboration Service
- Asset Service
- Batch Job Service
- Proposal Web Service

Before using Conga CPQ, you must be familiar with the following:

- Basic knowledge of Salesforce
- Basic knowledge of SOAP APIs
- Salesforce and Conga terms and definitions

Select one of the following topics for more information:

Getting Started as a Developer

This section describes the following topics:

API Supported Packages

Refer to the *Conga CPQ Release Notes* for the package details of the latest release.

Document Setup

The CPQ API reference Guide is divided into two sections: *API Reference* and *Scenarios*.

API Reference	The *API reference* section details the APIs that you can use to manipulate Conga objects through API calls and passing parameters. The API reference section also includes some code samples.
Scenarios	The *Scenarios* section details examples of the APIs you require to complete a specific task such as, adding products and constraint rules to the cart. The scenarios are classified by theme. You can refer to the generic examples of scenarios to identify the calls you can use to achieve your objective.

API Standards and Development Platforms

CPQ APIs are based on Salesforce APIs and use the same standards and platforms

Standards

Name	Reference
Simple Object Access Protocol (SOAP) 1.1	http://www.w3.org/TR/2000/NOTE-SOAP-20000508
Web Service Description Language (WSDL) 1.1	http://www.w3.org/TR/2001/NOTE-wsdl-20010315
WS-I Basic Profile 1.1	http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

Development Platforms

CPQ SOAP API works with standard SOAP development environments. For a list of compatible development platforms, see Salesforce Developer Force API details.

Field Types

CPQ APIs use a subset of the supported data and field types on Salesforce.

The following table lists the APIs that Conga provides. For a comprehensive list of all field types supported by Salesforce, see Salesforce Data Types.

Type	Description
Boolean	The **Boolean** field has a *true*(or 1) or *false*(or 0) value.
Data object	The **Data Object** field is an ID type and is represented by `CPQ.nnDO` in this document.

Type	Description
Date	The **Date** field contains date values only and do not contain relevant time values. Time in a date field is always set to midnight in the UTC time zone. If you want a timestamp you must use a **dateTime** field.
Decimal	The **Decimal** field provides an exact numeric value and you can arbitrarily size the precision and scale of the value.
ID	The **ID** field is an alphanumeric field that acts like the primary key for a specific record associated with an object. The ID value includes a three-character code that identifies which object the record is associated with. The ID for a specific record does not change. For some objects, this field may also be a `reference` type value, which contains the ID value for a related record. They are identified by field names ending in 'Id', such as `priceListId`. The ID field acts like foreign keys and their values can be changed using an `update()` call.
Integer	The **Integer** field contains whole numbers only. There are no digits after the decimal.
List	The **List** field includes a fixed set of values from which you must select a single value. Picklists are available as drop-down lists. If a picklist is unrestricted, the API does not limit entries to only currently active values.
String	The **String** field contains text and may have differing length restrictions based on the data you store in the specific field. For instance, `City` may be limited to 50 characters, while `AddressLine1` is limited to 255 characters.

Recommendations

Following are the recommendations for CPQ SOAP APIs on the Salesforce Platform:

- You must use the Public Admin APIs to integrate the CPQ master data.
  In a nutshell, the master data includes products, rules, and pricing setup as a whole.
- You cannot use generic APIs for any of the master datasets.
  CPQ does not provide support for any data that is integrated through the generic API mechanism for client implementation.

Integrating Conga with External Systems

Additional steps are required when you choose to integrate CPQ on Salesforce with external applications, customer portals, or other critical business systems. Because CPQ Web Services are hosted on Salesforce, you should familiarize yourself with the Salesforce SOAP API and processes surrounding integration and best practices detailed here: https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/sforce_api_quickstart_intro.htm

> **Best Practices**
>
> It is recommended that you work with Professional Services to design and implement your integration. Use this documentation for basic integration steps and to reference CPQ Web Service calls.

The following basic steps are required to get started using the CPQ Web Services API.

1. **Generate the Enterprise or Partner WSDL** – Integration with data stored in Salesforce requires you to first point your browser to the Salesforce Enterprise or Partner WSDL. This WSDL is generally provided by Professional Services. Refer to Salesforce Documentation for complete instructions on generating the Web Service WSDL.
2. **Generate the Web Services WSDL** – After you have connected to Web Service, go to your organization and download the WSDL for the appropriate Web Service (CPQ Web Service, Batch Update, Proposal Web Service, and so on).
3. **Import the WSDL Files Into Your Development Platform** – After you have generated the WSDL files, you can import them into your development platform. Conga does not provide instructions for the import process. Refer to Salesforce documentation or documentation related to your development platform.
4. **Connect to Conga** – Before you can begin using CPQ Web Services, you must first authenticate to Conga using the login() API.

Generating the Conga Web Services WSDL

Before you can import Conga SOAP Web Service into your development or testing platform, you must generate and download the Conga WSDL for the appropriate Web Service.

The example provided here uses SoapUI.

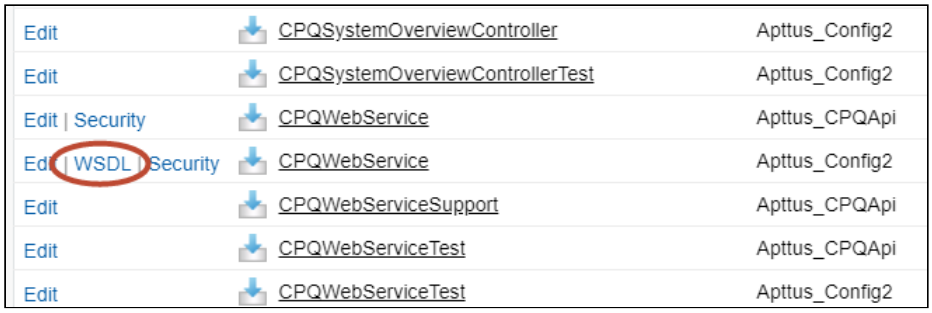
There a known bug in the WSDL Generator on Salesforce that does not include several field types, so it is recommended to update the WSDL file after you have generated it but *before* importing it into your development platform. You can find the details for any workaround tasks here:

- https://success.salesforce.com/issues_view?id=a1p3A000000eatxQAA&title=generated-wsdl-for-apex-webservices-is-malformed
- https://success.salesforce.com/issues_view?id=a1p300000008XKUAA2

When updating generated WSDL, make sure that the target namespace for any schema you add points to the correct Web Service (for example, schemas/class/Apttus_QPConfig/QPConfigWebService). If you are still having trouble, please ask Conga Professional Services for a modified WSDL for the Web Services you are using.

To generate the Conga Web Services WSDL

1. Log in to the Salesforce organization that contains your Conga records and data (sandbox or production).
2. Go to **Setup > Develop > Apex Classes** (on Lightning, go to **Setup > Custom Code > Apex Classes**.
3. Find the Web Service you want to generate the WSDL for (for example, **CPQWebService**).
4. Click the **WSDL** link to generate the WSDL. The WSDL XML is generated and displayed in a new tab.



5. Right-click on the page and select **View Page Source**. Copy the XML content to any text editor.
6. Save the file with the extension **.wsdl**.
7. Open SoapUI (or wherever is required on your development platform).
8. Create a new SOAP project and import the Conga Web Services WSDL. All methods under that Web Service are now available to call.

> Refer to the **Request/Response XML** section for any API in this reference to get the structure of the request and any prerequisite calls required for any API.

Connecting to Conga

After you have downloaded the Enterprise or Partner WSDL, call the **login()** method to obtain a session ID from your org that you can use when calling CPQ Web Services. After authenticating, you can use the same session ID until it either expires or your logout or login again.

The example provided here uses SoapUI, an API testing tool which can be downloaded for free here: https://www.soapui.org/.

Prerequisite: To authenticate with Conga, please make sure to have your production or test org credentials on hand (username and password).

To connect to Conga Web Service using SoapUI

1. Open SoapUI. Go to **File > New SOAP Project**.
2. Enter a name for the project.
3. Click **Browse**. Navigate to the saved Enterprise or Partner WSDL file that you downloaded and click **Open**.
4. Click **OK** to close the project window.
5. From the Navigation panel to the left, highlight the project folder and click to expand. Click to expand the **SoapBinding**. The list of methods that comprise the Enterprise or Partner services are displayed.
6. Scroll down and right click on **login**. Double-click on an existing **Request**. The request window opens in the SoapUI interface.

> If you are doing this for the first time, you need to right-click on the **login** method and select **New Request**.

7. Select and delete all content following the **<soapenv:Header>** tag and the **</soapenv:Header>** tag.
8. Enter the username for your org (must have appropriate privileges) between the **<urn:username>** and **</urn:username>** tags.
9. Enter the username for your org (must have appropriate privileges) between the **<urn:password>** and **</urn:password>** tags.

The request should look like the following:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:partner.soap.sforce.com">
   <soapenv:Header>
   </soapenv:Header>
   <soapenv:Body>
      <urn:login>
         <urn:username>username@example.com</urn:username>
         <urn:password>password</urn:password>
      </urn:login>
   </soapenv:Body>
</soapenv:Envelope>
```

10. From the upper-left corner of the window, click the **Run** () icon. The response is generated in the right-hand window.
11. Make note of the **serverURL** and the **sessionId** returned by the server. You will use the information to make Conga Web Service calls.

Generic API Operations in Salesforce (SOAP)

The following methods are provided in the Enterprise or Partner WSDL as generic core API calls that allow you primarily to perform CRUD actions on standard and custom Salesforce objects (also referred to as "data setup"). Refer to Salesforce Developer documentation for complete information on calling these methods. The examples provided on this page will aid you in performing CRUD actions and querying information from the Salesforce database to use in Conga API calls.

Retrieving the Server URL and Session Id (Login)

All API calls to your organization require you to authenticate with a valid Session Id.

Method: **login()**

Request: Pass the **username** and **password** for the user providing authentication.

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="
urn:partner.soap.sforce.com">
   <soapenv:Header>
```

```
   </soapenv:Header>
   <soapenv:Body>
      <urn:login>
         <urn:username>username</urn:username>
         <urn:password>password</urn:password>
      </urn:login>
   </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **serverURL** and **session Id**.

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ur
n:partner.soap.sforce.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <loginResponse>
         <result>
            <serverUrl>serverURL</serverUrl>
            <sessionId>sessionId</sessionId>
         </result>
      </loginResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Retrieving an Opportunity

Method: **query()**

Request: Pass the **session Id** and **Opportunity Name**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>session Id</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
```

```
    <soapenv:Body>
        <urn:query>
            <urn:queryString>select id from opportunity where
name='opportunityName'</urn:queryString>
        </urn:query>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Opportunity Id** and **Size** (indicates number of Ids)

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:enterprise.soap.sforce.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>4973</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <queryResponse>
            <result>
                <done>true</done>
                <queryLocator xsi:nil="true"/>
                <records xsi:type="sf:Opportunity">
                    <sf:Id>006Z000000Gr9ODIAZ</sf:Id>
                </records>
                <size>1</size>
            </result>
        </queryResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Creating a Price List

Method: **create()**

Request: Pass the **session Id** and **Price List Name**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com"
    xmlns:urn1="urn:sobject.enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>session Id</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:create>
            <urn:sObjects xsi:type="Apttus_Config2__PriceList__c"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <Name>Price List Name</Name>
            </urn:sObjects>
        </urn:create>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Price List Id** with status = 'true' if the Price List was successfully created.

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>4983</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
```

```
        <createResponse>
            <result>
                <id>a1DZ0000002i9IWMAY</id>
                <success>true</success>
            </result>
        </createResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving a Price List

Method: **query()**

Request: Pass the **session Id** and **Price List Name**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>${#Project#SessionID}</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:query>
            <urn:queryString>select Id from Apttus_Config2__PriceList__c where
         name='priceListName'
         </urn:queryString>
        </urn:query>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Price List Id**.

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ur
n:partner.soap.sforce.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
   <soapenv:Header>
```

```
      <LimitInfoHeader>
         <limitInfo>
            <current>637</current>
            <limit>5000000</limit>
            <type>API REQUESTS</type>
         </limitInfo>
      </LimitInfoHeader>
   </soapenv:Header>
   <soapenv:Body>
      <queryResponse>
         <result xsi:type="QueryResult">
            <done>true</done>
            <queryLocator xsi:nil="true"/>
            <records xsi:type="sf:sObject">
               <sf:type>Apttus_Config2__PriceList__c</sf:type>
               <sf:Id>a1DZ0000002mg5nMAA</sf:Id>
            </records>
            <size>1</size>
         </result>
      </queryResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Creating a Product Category

Method: **create()**

Request: Pass the **session Id**, **Category Name**, and **Category Label**.

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="
urn:enterprise.soap.sforce.com" xmlns:urn1="urn:sobject.enterprise.soap.sforce.com">
   <soapenv:Header>
      <urn:SessionHeader>
         <urn:sessionId>session Id</urn:sessionId>
      </urn:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <urn:create>
```

```
         <urn:sObjects xsi:type="Apttus_Config2__ClassificationName__c" xmlns:xsi="ht
tp://www.w3.org/2001/XMLSchema-instance">
           <Name>Category Name</Name>
           <Apttus_Config2__HierarchyLabel__c>Category Label</
Apttus_Config2__HierarchyLabel__c>
         </urn:sObjects>
      </urn:create>
   </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Category Id** with status = 'true' if the Category was successfully created.

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>5011</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <createResponse>
            <result>
                <id>a0oZ0000002u94bIAA</id>
                <success>true</success>
            </result>
        </createResponse>
    </soapenv:Body>
</soapenv:Envelope>undefined</soapenv:Body>undefined</soapenv:Envelope>
```

Retrieving a Category

Method: **query()**

Request: Pass the **session Id** and **Category Name**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>${#Project#SessionID}</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:query>
            <urn:queryString>select Id, Name from
Apttus_Config2__ClassificationName__c where
         name='Category Name'
         </urn:queryString>
        </urn:query>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Category Id**, **Category Name**, and **Size** (indicates number of Ids).

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:enterprise.soap.sforce.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>5017</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <queryResponse>
            <result>
```

```
                <done>true</done>
                <queryLocator xsi:nil="true"/>
                <records xsi:type="sf:Apttus_Config2__ClassificationName__c">
                    <sf:Id>a0oe0000005l37UAAQ</sf:Id>
                    <sf:Name>AutoQuoteOfferings</sf:Name>
                </records>
                <size>1</size>
            </result>
        </queryResponse>
    </soapenv:Body>
</soapenv:Envelope>undefined</soapenv:Body>undefined</soapenv:Envelope>undefined</
soapenv:Envelope>
```

Creating a Standalone Product

Method: **create()**

Request: Pass the **session Id**, **Product Name**, and **Configuration Type**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com"
    xmlns:urn1="urn:sobject.enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>${#Project#SessionID}</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:create>
            <urn:sObjects xsi:type="Product2"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <Name>${#Project#createStandloneProductName}</Name>
                <Apttus_Config2__ConfigurationType__c>Standalone</
Apttus_Config2__ConfigurationType__c>
            </urn:sObjects>
        </urn:create>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Product Id** with status = 'true' if the Product was successfully created.

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>5033</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <createResponse>
            <result>
                <id>01tZ0000004ypoaIAA</id>
                <success>true</success>
            </result>
        </createResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving a Standalone Product

Method: **query()**

Request: Pass the **session Id** and **Product Name**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>${#Project#SessionID}</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
```

```
        <urn:query>
            <urn:queryString>select Id, Name from Product2 where name='Standalone
Product Name'</urn:queryString>
        </urn:query>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Standalone Product Id**, **Product Name**, and **Size** (indicates the number of Ids).

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:partner.soap.sforce.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sf="urn:sobject.partner.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>638</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <queryResponse>
            <result xsi:type="QueryResult">
                <done>true</done>
                <queryLocator xsi:nil="true"/>
                <records xsi:type="sf:sObject">
                    <sf:type>Product2</sf:type>
                    <sf:Id>01tZ0000004lUXeIAM</sf:Id>
                    <sf:Name>Auto_API_StandaloneProduct3</sf:Name>
                </records>
                <size>1</size>
            </result>
        </queryResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Creating a Bundle Product

Method: **create()**

Request: Pass the **session Id**, **Product Name**, and **Configuration Type**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com"
    xmlns:urn1="urn:sobject.enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>${#Project#SessionID}</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:create>
            <urn:sObjects xsi:type="Product2"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <Name>Bundle Product Name</Name>
                <Apttus_Config2__ConfigurationType__c>Bundle</
Apttus_Config2__ConfigurationType__c>
            </urn:sObjects>
        </urn:create>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Product Id** with status = 'true' if the Bundle Product was successfully created.

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
```

```
                <current>5033</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <createResponse>
            <result>
                <id>01tZ0000004yrtyaIAA</id>
                <success>true</success>
            </result>
        </createResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving a Bundle

Method: **query()**

Request: Pass the **session Id** and **Product Name**.

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="
urn:partner.soap.sforce.com">
   <soapenv:Header>
      <urn:SessionHeader>
         <urn:sessionId>00DZ000000NAEIA!
ASAAQHmIRqgn4R9Oi1yQjWTIVk4UZmsDe_.eK0Z9z6qLij7Tu.L_Yo8dRA_p8OmhKMeRs4uzCZTadIgQ9fEbD
KciXEQYRyaA</urn:sessionId>
      </urn:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <urn:query>
            <urn:queryString>select Id, Name from Product2 where name='Bundle Product
Name'
         </urn:queryString>
      </urn:query>
   </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Bundle Id** and **Product Name**, and **Size** (indicates the number of Ids).

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="urn:partner.soap.sforce.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sf="urn:sobject.partner.soap.sforce.com">
   <soapenv:Header>
      <LimitInfoHeader>
         <limitInfo>
            <current>643</current>
            <limit>5000000</limit>
            <type>API REQUESTS</type>
         </limitInfo>
      </LimitInfoHeader>
   </soapenv:Header>
   <soapenv:Body>
      <queryResponse>
         <result xsi:type="QueryResult">
            <done>true</done>
            <queryLocator xsi:nil="true"/>
            <records xsi:type="sf:sObject">
               <sf:type>Product2</sf:type>
               <sf:Id>01tZ0000004lUXtIAM</sf:Id>
               <sf:Name>Auto_API_BundleProduct1</sf:Name>
            </records>
            <size>1</size>
         </result>
      </queryResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Creating an Option Product

Method: **create()**

Request: Pass the **session Id**, **Product Name**, and **Configuration Type**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com"
```

```
    xmlns:urn1="urn:sobject.enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>${#Project#SessionID}</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:create>
            <urn:sObjects xsi:type="Product2"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <Name>Option Product Name</Name>
                <Apttus_Config2__ConfigurationType__c>Option</
Apttus_Config2__ConfigurationType__c>
            </urn:sObjects>
        </urn:create>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Product Id** with status = 'true' if the Option Product was successfully created.

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>5033</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <createResponse>
            <result>
                <id>01tZ0008984yrterIAA</id>
                <success>true</success>
            </result>
        </createResponse>
    </soapenv:Body>
```

```
</soapenv:Envelope>
```

Retrieving an Option

Method: **query()**

Request: Pass the **session Id** and **Product Name**.

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="
urn:partner.soap.sforce.com">
   <soapenv:Header>
      <urn:SessionHeader>
         <urn:sessionId>00DZ000000NAEIA!
ASAAQHmIRqgn4R9Oi1yQjWTIVk4UZmsDe_.eK0Z9z6qLij7Tu.L_Yo8dRA_p8OmhKMeRs4uzCZTadIgQ9fEbD
KciXEQYRyaA</urn:sessionId>
      </urn:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <urn:query>
            <urn:queryString>select Id, Name from Product2 where name='Option Product
Name'
         </urn:queryString>
      </urn:query>
   </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Option Id**, **Product Name**, and **Size** (indicates the number of Ids).

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ur
n:partner.soap.sforce.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
   <soapenv:Header>
      <LimitInfoHeader>
         <limitInfo>
            <current>648</current>
            <limit>5000000</limit>
            <type>API REQUESTS</type>
```

```
            </limitInfo>
         </LimitInfoHeader>
      </soapenv:Header>
      <soapenv:Body>
         <queryResponse>
            <result xsi:type="QueryResult">
               <done>true</done>
               <queryLocator xsi:nil="true"/>
               <records xsi:type="sf:sObject">
                  <sf:type>Product2</sf:type>
                  <sf:Id>01tZ0000004lUYNIA2</sf:Id>
                  <sf:Name>Auto_API_OptionProduct2</sf:Name>
               </records>
               <size>1</size>
            </result>
         </queryResponse>
      </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Component Id for a Product, Bundle, or Option

Method: **query()**

Request: Pass the **session Id** and **Product Id**.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>${#Project#SessionID}</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:query>
            <urn:queryString>select Id from Apttus_Config2__ProductOptionComponent__c
where
         Apttus_Config2__ComponentProductId__c = 'Product Id'</urn:queryString>
        </urn:query>
    </soapenv:Body>
</soapenv:Envelope>
```

Response: Returns the **Component Product Id**.

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="urn:partner.soap.sforce.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sf="urn:sobject.partner.soap.sforce.com">
    <soapenv:Header>
        <LimitInfoHeader>
            <limitInfo>
                <current>649</current>
                <limit>5000000</limit>
                <type>API REQUESTS</type>
            </limitInfo>
        </LimitInfoHeader>
    </soapenv:Header>
    <soapenv:Body>
        <queryResponse>
            <result xsi:type="QueryResult">
                <done>true</done>
                <queryLocator xsi:nil="true"/>
                <records xsi:type="sf:sObject">
                    <sf:type>Apttus_Config2__ProductOptionComponent__c</sf:type>
                    <sf:Id>a1YZ0000003pII9MAM</sf:Id>
                </records>
                <size>1</size>
            </result>
        </queryResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

API Reference

In this section the following topics are described:

- Proposal Web Service
- Batch Update Service
- CPQ Web Service(Apttus_CPQApi)
- CPQ Web Service (Apttus_Config2)

- CPQ Admin Web Service
- Batch Job Service
- Constraint Web Service 2
- Quote Collaboration Service
- Favorite Configuration Global Service
- Quote/Proposal Config Web Service
- Asset Service
- Remote CPQ Admin Controller
- Asset Web Services
- Merge Web Service

Proposal Web Service

The Proposal web service APIs account for the standard actions to account, opportunity, and quotes/proposals.

You can invoke APIs in Proposal Service using the following command:

```
Apttus_Proposal.ProposalWebService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the Proposal web service APIs to complete the following tasks:

- Retrieving Proposal Field Values from Account
- Retrieving Proposal Field Values from Opportunity
- Retrieving Field Values from Proposal
- Creating Proposal Line Items from Opportunity
- Updating a Proposal
- Cloning Email Template
- Cloning Document Templates
- Getting the Proposal Document Output Format
- Copying Attachments
- Creating a Task After Proposal is Presented
- Retrieving Name of the First Template
- Retrieving Templates in a Proposal
- Deleting an Email Template

Retrieving Proposal Field Values from Account

This API fetches the field values of the proposal object that are required to create a proposal from an account. You can use the fetched values to create a new proposal.

API	Signature
`createProposalFromAccount`	*webService static Apttus_Proposal__Proposal__c createProposalFromAccount(Id accountId, Id recordTypeId)*

Request Parameter		
Name	**Type**	**Description**
accountID	ID	The ID of the account you need to retrieve values from.
recordTypeId	ID	Record Type of the proposal object

Response - Apttus_Proposal__Proposal__c		
Field	**Type**	**Description**
RecordTypeId	ID	Record Type of the proposal object
Apttus_Proposal__Proposal_Name__c	String	The name of the proposal.
Apttus_Proposal__Account__c	ID	The Id of the account associated with the proposal.
Apttus_Proposal__Primary_Contact__c	ID	The primary contact associated with the proposal.
Apttus_Proposal__Description__c	String	The description of the proposal.

Response - Apttus_Proposal__Proposal__c		
Field	**Type**	**Description**
Apttus_Proposal__Primary__c	Boolean	Indicates whether the proposal is the primary quote to update the related opportunity.
Apttus_Proposal__ReadyToGenerate__c	Boolean	Indicates whether the proposal is ready for generation.
Apttus_Proposal__ReadyToPresent__c	Boolean	Indicates whether the proposal is ready for presentation.
OwnerId	ID	The ID of the proposal object.
CurrencyIsoCode	String	The currency defined for the proposal.

Code Sample

The sample code below enables you to retrieve the proposal field values of the account associated with the account ID that you provide. You can use the standard *createRecord* API to create the proposal.

```
/**
 * The below method demonstrates fetch the values proposal field values from an existing account
 */
public static Apttus_Proposal__Proposal__c createProposal(Id accountID) {
Apttus_Proposal__Proposal__c newProposalSO = new Apttus_Proposal__Proposal__c();
ID proposalRecordTypeID = [select ID FROM recordType where sObjectType='Apttus_Proposal__Proposal__c' AND Name = ‘Proposal’ LIMIT 1].Id;
newProposalSO = Apttus_Proposal.ProposalWebService.createProposalFromAccount(accountID, proposalRecordTypeID);
    return newProposalSO;
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

- Retrieve Proposal information
- Create Proposal record

API Prerequisites

None.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/
ProposalWebService">
    <soapenv:Header>
        <prop:SessionHeader>
            <prop:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</prop:sessionId>
        </prop:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <prop:createProposalFromAccount>
            <prop:accountId>001Z000001UtoRL</prop:accountId>
        </prop:createProposalFromAccount>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <createProposalFromAccountResponse>
            <result xsi:type="Apttus_Proposal__Proposal__c">
                <Id xsi:nil="true"/>
                <Apttus_Proposal__Account__c>001Z000001V13ovIAB</
Apttus_Proposal__Account__c>
                <Apttus_Proposal__Primary__c>false</Apttus_Proposal__Primary__c>
                <Apttus_Proposal__Proposal_Name__c>Auto_API_Account</
Apttus_Proposal__Proposal_Name__c>
                <Apttus_Proposal__ReadyToGenerate__c>true</
Apttus_Proposal__ReadyToGenerate__c>
                <Apttus_Proposal__ReadyToPresent__c>true</
Apttus_Proposal__ReadyToPresent__c>
                <CurrencyIsoCode>USD</CurrencyIsoCode>
                <OwnerId>0050U000000r444QAA</OwnerId>
            </result>
        </createProposalFromAccountResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Creating a Proposal

Use data from the Response to create the proposal. The following example uses the *createRecord()* method to create the Proposal object.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com"
    xmlns:urn1="urn:sobject.enterprise.soap.sforce.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:create>
```

```
            <urn:sObjects xsi:type="urn1:Apttus_Proposal__Proposal__c">
                <urn1:Apttus_Proposal__Proposal_Name__c>Auto_API_Account</urn1:Apttus
_Proposal__Proposal_Name__c>
                <urn1:Apttus_Proposal__Account__c>001Z000001V13ovIAB</urn1:Apttus_Pro
posal__Account__c>
                <urn1:Apttus_Proposal__Primary__c>false</urn1:Apttus_Proposal__Primar
y__c>
                <urn1:Apttus_Proposal__ReadyToGenerate__c>true</urn1:Apttus_Proposal_
_ReadyToGenerate__c>
                <urn1:Apttus_Proposal__ReadyToPresent__c>true</urn1:Apttus_Proposal__
ReadyToPresent__c>
                <urn1:CurrencyIsoCode>USD</urn1:CurrencyIsoCode>
                <urn1:Apttus_Proposal__Opportunity__c>006Z000000Gr9OD</urn1:Apttus_Pr
oposal__Opportunity__c>
                <urn1:Apttus_QPConfig__PriceListId__c>a1De0000001yPXQ</urn1:Apttus_QP
Config__PriceListId__c>
            </urn:sObjects>
        </urn:create>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <createProposalFromAccountResponse>
            <result xsi:type="Apttus_Proposal__Proposal__c">
                <Id xsi:nil="true"/>
                <Apttus_Proposal__Account__c>001Z000001V13ovIAB</
Apttus_Proposal__Account__c>
                <Apttus_Proposal__Primary__c>false</Apttus_Proposal__Primary__c>
                <Apttus_Proposal__Proposal_Name__c>Auto_API_Account</
Apttus_Proposal__Proposal_Name__c>
                <Apttus_Proposal__ReadyToGenerate__c>true</
Apttus_Proposal__ReadyToGenerate__c>
                <Apttus_Proposal__ReadyToPresent__c>true</
Apttus_Proposal__ReadyToPresent__c>
                <CurrencyIsoCode>USD</CurrencyIsoCode>
```

```
            <OwnerId>0050U000000r444QAA</OwnerId>
        </result>
    </createProposalFromAccountResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Retrieving Proposal Field Values from Opportunity

This API fetches the field values of the proposal object that are required to create a proposal from an opportunity. You can use the fetched values to create a new proposal.

API	Signature
createProposalFromOpportunity	*webService static Apttus_Proposal__Proposal__c createProposalFromOpportunity(Id opptyId, Id recordTypeId)*

Request Parameter		
Name	**Type**	**Description**
opptyID	ID	The ID of the opportunity you need to retrieve values from.
recordTypeId	ID	Record Type of the proposal object

Response - Apttus_Proposal__Proposal__c		
Field	**Type**	**Description**
RecordTypeId	ID	Record Type of the proposal object
Apttus_Proposal__Proposal_Name__c	String	The name of the proposal.

Response - Apttus_Proposal__Proposal__c		
Field	**Type**	**Description**
Apttus_Proposal__Account__c	ID	The Id of the account associated with the proposal.
Apttus_Proposal__Opportunity__c	ID	The Id of the opportunity associated with the proposal.
Apttus_Proposal__Primary_Contact__c	ID	The primary contact associated with the proposal.
Apttus_Proposal__Description__c	String	The description of the proposal.
Apttus_Proposal__Primary__c	Boolean	Indicates whether the proposal is the primary quote to up
Apttus_Proposal__ReadyToGenerate__c	Boolean	Indicates whether the proposal is ready for generation.
Apttus_Proposal__ReadyToPresent__c	Boolean	Indicates whether the proposal is ready for presentation.
OwnerId	ID	The ID of the proposal object.
CurrencyIsoCode	String	The currency defined for the proposal.

Code Sample

The sample code below enables you to retrieve the proposal field values of the opportunity associated with the ID that you provide. You can use the standard *createRecord* API to create the proposal.

```
1  /**
2   * The below method demonstrates how to fetch the values proposal field
   values from an existing opporutnity
3   */
4  public static Apttus_Proposal__Proposal__c createProposal(Id
   opportunityID) {
5  Apttus_Proposal__Proposal__c newProposalSO = new
    Apttus_Proposal__Proposal__c();
```

```
6   ID proposalRecordTypeID = [select ID FROM recordType where
    sObjectType='Apttus_Proposal__Proposal__c' AND Name = 'Proposal' LIMIT 1].
    Id;
7   newProposalSO =
    Apttus_Proposal.ProposalWebService.createProposalFromOpportunity(opportuni
    tyID, proposalRecordTypeID);
8       return newProposalSO;
9   }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

- Retrieve Proposal information
- Create Proposal record

API Prerequisites

None.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService">
   <soapenv:Header>
      <prop:SessionHeader>
         <prop:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</prop:sessionId>
      </prop:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <prop:createProposalFromOpportunity>
         <prop:opptyId>006Z000000Gr9OD</prop:opptyId>
         <prop:recordTypeId>012i00000001ECb</prop:recordTypeId>
      </prop:createProposalFromOpportunity>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <createProposalFromOpportunityResponse>
            <result xsi:type="Apttus_Proposal__Proposal__c">
                <Id xsi:nil="true"/>
                <Apttus_Proposal__Account__c>001Z000001V13ovIAB</
Apttus_Proposal__Account__c>
                <Apttus_Proposal__Opportunity__c>006Z000000Gr9ODIAZ</
Apttus_Proposal__Opportunity__c>
                <Apttus_Proposal__Primary__c>false</Apttus_Proposal__Primary__c>
                <Apttus_Proposal__Proposal_Name__c>Auto_API_Opportunity</
Apttus_Proposal__Proposal_Name__c>
                <Apttus_Proposal__ReadyToGenerate__c>true</
Apttus_Proposal__ReadyToGenerate__c>
                <Apttus_Proposal__ReadyToPresent__c>true</
Apttus_Proposal__ReadyToPresent__c>
                <CurrencyIsoCode>USD</CurrencyIsoCode>
                <OwnerId>0050U000000r444QAA</OwnerId>
                <RecordTypeId>012i00000001ECbAAM</RecordTypeId>
            </result>
        </createProposalFromOpportunityResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Create Proposal

Use data from the Response to create the proposal. The following example uses the *createRecord()* method to create the Proposal object.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:urn="urn:enterprise.soap.sforce.com"
    xmlns:urn1="urn:sobject.enterprise.soap.sforce.com"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:create>
            <urn:sObjects xsi:type="urn1:Apttus_Proposal__Proposal__c">
                <urn1:Apttus_Proposal__Proposal_Name__c>Auto_API_Opportunity</urn1:Ap
ttus_Proposal__Proposal_Name__c>
                <urn1:Apttus_Proposal__Account__c>001Z000001V13ovIAB</urn1:Apttus_Pro
posal__Account__c>
                <urn1:Apttus_Proposal__Primary__c>false</urn1:Apttus_Proposal__Primar
y__c>
                <urn1:Apttus_Proposal__ReadyToGenerate__c>true</urn1:Apttus_Proposal_
_ReadyToGenerate__c>
                <urn1:Apttus_Proposal__ReadyToPresent__c>true</urn1:Apttus_Proposal__
ReadyToPresent__c>
                <urn1:CurrencyIsoCode>USD</urn1:CurrencyIsoCode>
                <urn1:Apttus_Proposal__Opportunity__c>006Z000000Gr9OD</urn1:Apttus_Pr
oposal__Opportunity__c>
                <urn1:Apttus_QPConfig__PriceListId__c>a1De0000001yPXQ</urn1:Apttus_QP
Config__PriceListId__c>
            </urn:sObjects>
        </urn:create>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <createProposalFromOpportunityResponse>
            <result xsi:type="Apttus_Proposal__Proposal__c">
                <Id xsi:nil="true"/>
                <Apttus_Proposal__Account__c>001Z000001V13ovIAB</
Apttus_Proposal__Account__c>
```

```
                <Apttus_Proposal__Primary__c>false</Apttus_Proposal__Primary__c>
                <Apttus_Proposal__Proposal_Name__c>Auto_API_Account</
Apttus_Proposal__Proposal_Name__c>
                <Apttus_Proposal__ReadyToGenerate__c>true</
Apttus_Proposal__ReadyToGenerate__c>
                <Apttus_Proposal__ReadyToPresent__c>true</
Apttus_Proposal__ReadyToPresent__c>
                <CurrencyIsoCode>USD</CurrencyIsoCode>
                <OwnerId>0050U000000r444QAA</OwnerId>
            </result>
        </createProposalFromOpportunityResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Field Values from Proposal

This API fetches the field values of the proposal object that are required to clone from an existing proposal. You can use the fetched values to clone the proposal.

API	Signature
cloneProposal	*webService static Apttus_Proposal__Proposal__c cloneProposal(Id originalId)*

Request Parameter		
Name	**Type**	**Description**
proposalIds	ID	The ID of the proposal you want to clone.

Response - Apttus_Proposal__Proposal__c		
Field	**Type**	**Description**
RecordTypeId	ID	Record Type of the proposal object
Apttus_Proposal__Proposal_Name__c	String	The name of the proposal.
Apttus_Proposal__Account__c	ID	The Id of the account associated with the proposal.
Apttus_Proposal__Primary_Contact__c	ID	The primary contact associated with the proposal.
Apttus_Proposal__Description__c	String	The description of the proposal.
Apttus_Proposal__Primary__c	Boolean	Indicates whether the proposal is the primary quote to up
Apttus_Proposal__ReadyToGenerate__c	Boolean	Indicates whether the proposal is ready for generation.
Apttus_Proposal__ReadyToPresent__c	Boolean	Indicates whether the proposal is ready for presentation.
OwnerId	ID	The ID of the proposal object.
CurrencyIsoCode	String	The currency defined for the proposal.

Code Sample

The sample code below enables you to retrieve the proposal field values from the proposal associated with the ID that you provide. You can use the standard *createRecord* API to create the new proposal.

```
1  /**
2   * The below code demonstrates how to fetch the fields values from an
   existing proposal
3   */
4  Apttus_Proposal__Proposal__c proposalSO = new
    Apttus_Proposal__Proposal__c();
```

```
proposalSO =
Apttus_Proposal.ProposalWebService.cloneProposal(proposalName);
System.debug(proposalSO);
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

None.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/
ProposalWebService">
    <soapenv:Header>
        <prop:SessionHeader>
            <prop:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</prop:sessionId>
        </prop:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <prop:cloneProposal>
            <prop:originalId>a0eZ0000005pF3U</prop:originalId>
        </prop:cloneProposal>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <cloneProposalResponse>
            <result xsi:type="Apttus_Proposal__Proposal__c">
                <Id xsi:nil="true"/>
                <ABOTerminate__c>_HL_ENCODED_/apex/
Apttus_QPConfig__ProposalConfiguration?id=a0eZ0000005pF3U&flow=ABOTerminate_HL__IM1_/
resource/Apttus_QPConfig__Button_Configure_IM2_Configure Products_IM3__HL__self_HL_</
ABOTerminate__c>
                <Apttus_Proposal__Account__c>001Z000001UeYiiIAF</
Apttus_Proposal__Account__c>
                <Apttus_Proposal__Amount__c>0.00</Apttus_Proposal__Amount__c>
                <Apttus_Proposal__Approval_Stage__c>Draft</
Apttus_Proposal__Approval_Stage__c>
                <Apttus_Proposal__Generate__c>_HL_ENCODED_/apex/
Apttus_Proposal__ProposalGenerate?id=a0eZ0000005pF3U_HL__IM1_/resource/
Apttus_Proposal__Button_Generate_IM2_Generate_IM3__HL__self_HL_</
Apttus_Proposal__Generate__c>
                <Apttus_Proposal__Grand_Total__c>0.00</
Apttus_Proposal__Grand_Total__c>
                <Apttus_Proposal__MakePrimary__c>_HL_ENCODED_/apex/
Apttus_Proposal__ProposalMakePrimary?id=a0eZ0000005pF3U_HL__IM1_/resource/
Apttus_Proposal__Button_MakePrimary_IM2_Make Primary_IM3__HL__self_HL_</
Apttus_Proposal__MakePrimary__c>
                <Apttus_Proposal__Net_Amount__c>0.00</Apttus_Proposal__Net_Amount__c>
                <Apttus_Proposal__Opportunity__c>006Z000000Gr9ODIAZ</
Apttus_Proposal__Opportunity__c>
                <Apttus_Proposal__Payment_Term__c>Net 30 Days</
Apttus_Proposal__Payment_Term__c>
                <Apttus_Proposal__Present__c>_HL_ENCODED_/apex/
Apttus_Proposal__ProposalPresent?id=a0eZ0000005pF3U_HL__IM1_/resource/
Apttus_Proposal__Button_Present_IM2_Present_IM3__HL__self_HL_</
Apttus_Proposal__Present__c>
                <Apttus_Proposal__Preview__c>_HL_ENCODED_/apex/
Apttus_Proposal__ProposalGenerate?id=a0eZ0000005pF3U&action=Preview_HL__IM1_/
resource/Apttus_Proposal__Button_Preview_IM2_Preview_IM3__HL__self_HL_</
Apttus_Proposal__Preview__c>
                <Apttus_Proposal__Primary__c>false</Apttus_Proposal__Primary__c>
                <Apttus_Proposal__Proposal_Name__c>Test_CR003</
Apttus_Proposal__Proposal_Name__c>
                <Apttus_Proposal__ReadyToGenerate__c>true</
Apttus_Proposal__ReadyToGenerate__c>
```

```
                <Apttus_Proposal__ReadyToPresent__c>true</
Apttus_Proposal__ReadyToPresent__c>
                <Apttus_Proposal__Request_Approval__c>_HL_ENCODED_/servlet/
servlet.Integration?lid=01N700000009u2c&eid=a0eZ0000005pF3U_HL__IM1_/servlet/
servlet.ImageServer?
oid=00D70000000JXTN&id=01570000000e6nU_IM2_Submit_IM3__HL__blank_HL_</
Apttus_Proposal__Request_Approval__c>
                <Apttus_Proposal__Sales_Tax_Amount__c>0.00</
Apttus_Proposal__Sales_Tax_Amount__c>
                <Apttus_Proposal__SendProposal__c>_HL_ENCODED_/apex/
Apttus_Proposal__DocGen?id=a0eZ0000005pF3U&context=proposal&name=Q-01308003_HL__IM1_/
resource/Apttus_Proposal__Button_SendProposal_IM2_Send Proposal_IM3__HL__self_HL_</
Apttus_Proposal__SendProposal__c>
                <Apttus_QPAsset__ConfigureWithAssets__c>_HL_ENCODED_/apex/
Apttus_QPAsset__ProposalConfiguration?id=a0eZ0000005pF3U_HL__IM1_/resource/
Apttus_QPConfig__Button_Configure_IM2_Configure Products_IM3__HL__self_HL_</
Apttus_QPAsset__ConfigureWithAssets__c>
                <Apttus_QPConfig__AutoActivateOrder__c>false</
Apttus_QPConfig__AutoActivateOrder__c>
                <Apttus_QPConfig__AutoCreateBill__c>false</
Apttus_QPConfig__AutoCreateBill__c>
                <Apttus_QPConfig__AutoCreateRevenue__c>false</
Apttus_QPConfig__AutoCreateRevenue__c>
                <Apttus_QPConfig__BillingPreferenceId__c>a2ti0000000Fh9SAAS</
Apttus_QPConfig__BillingPreferenceId__c>
                <Apttus_QPConfig__ConfigureNG__c>_HL_ENCODED_/apex/
Apttus_QPConfig__ProposalConfiguration?id=a0eZ0000005pF3U&flow=NGDefault_HL__IM1_/
resource/Apttus_QPConfig__Button_Configure_IM2_Configure Products_IM3__HL__self_HL_</
Apttus_QPConfig__ConfigureNG__c>
                <Apttus_QPConfig__Configure__c>_HL_ENCODED_/apex/
Apttus_QPConfig__ProposalConfiguration?id=a0eZ0000005pF3U&flow=Default_HL__IM1_/
resource/Apttus_QPConfig__Button_Configure_IM2_Configure Products_IM3__HL__self_HL_</
Apttus_QPConfig__Configure__c>
                <Apttus_QPConfig__DisableCartVersioning__c>false</
Apttus_QPConfig__DisableCartVersioning__c>
                <Apttus_QPConfig__IsAutoAccepted__c>false</
Apttus_QPConfig__IsAutoAccepted__c>
                <Apttus_QPConfig__IsSystemGenerated__c>false</
Apttus_QPConfig__IsSystemGenerated__c>
                <Apttus_QPConfig__IsTaskPending__c>false</
Apttus_QPConfig__IsTaskPending__c>
                <Apttus_QPConfig__PriceListId__c>a1De0000001yPXQEA2</
Apttus_QPConfig__PriceListId__c>
```

```
                <Apttus_QPConfig__SourceChannel__c>Direct</
Apttus_QPConfig__SourceChannel__c>
                <Apttus_QPConfig__SyncAssetChangesToQuote__c>true</
Apttus_QPConfig__SyncAssetChangesToQuote__c>
                <Apttus_QPConfig__UseType__c>Main</Apttus_QPConfig__UseType__c>
                <AsyncFinalize_CartGrid__c>_HL_ENCODED_/apex/
Apttus_QPConfig__ProposalConfiguration?
id=a0eZ0000005pF3U&flow=AutoCartGrid&asyncFinalize=true_HL__IM1_/resource/
Apttus_QPConfig__Button_Configure_IM2_Configure Products_IM3__HL__self_HL_</
AsyncFinalize_CartGrid__c>
                <AutoCartGrid__c>_HL_ENCODED_/apex/
Apttus_QPConfig__ProposalConfiguration?id=a0eZ0000005pF3U&flow=AutoCartGrid_HL__IM1_/
resource/Apttus_QPConfig__Button_Configure_IM2_Configure Products_IM3__HL__self_HL_</
AutoCartGrid__c>
                <AutoCartReadOnly__c>_HL_ENCODED_/apex/
Apttus_QPConfig__ProposalConfiguration?
id=a0eZ0000005pF3U&flow=AutoCartGrid&mode=readOnly_HL__IM1_/resource/
Apttus_QPConfig__Button_Configure_IM2_Configure Products_IM3__HL__self_HL_</
AutoCartReadOnly__c>
                <CurrencyIsoCode>USD</CurrencyIsoCode>
                <IsDeleted>false</IsDeleted>
                <Net_Amount__c>0.00</Net_Amount__c>
                <Numeric_Range_Field__c>1</Numeric_Range_Field__c>
                <OwnerId>0050U000000r444QAA</OwnerId>
                <Quote_Checkbox_21145__c>false</Quote_Checkbox_21145__c>
                <RecordTypeId>012i00000001ECbAAM</RecordTypeId>
                <Region__c>India</Region__c>
                <SystemModstamp>2020-05-20T19:52:43.000Z</SystemModstamp>
            </result>
        </cloneProposalResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Creating a Proposal

Use data from the Response to create the proposal. The following uses the *createRecord()* method to create the Proposal object.

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
    xmlns:urn="urn:partner.soap.sforce.com"
    xmlns:urn1="urn:sobject.partner.soap.sforce.com">
    <soapenv:Header>
        <urn:SessionHeader>
            <urn:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</urn:sessionId>
        </urn:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <urn:create>
            <urn:sObjects>
                <urn1:type>Apttus_Proposal__Proposal__c</urn1:type>
                <urn1:Apttus_Proposal__Account__c>001Z000001UeYiiIAF</urn1:Apttus_Pro
posal__Account__c>
                <urn1:Apttus_Proposal__Approval_Stage__c>Draft</urn1:Apttus_Proposal_
_Approval_Stage__c>
                <urn1:Apttus_Proposal__Opportunity__c>006Z000000Gr9ODIAZ</urn1:Apttus
_Proposal__Opportunity__c>
                <urn1:Apttus_Proposal__Payment_Term__c>Net 30 Days</urn1:Apttus_Propo
sal__Payment_Term__c>
                <urn1:Apttus_Proposal__Primary__c>false</urn1:Apttus_Proposal__Primar
y__c>
                <urn1:Apttus_Proposal__Proposal_Name__c>Test_CR003-copy2</urn1:Apttus
_Proposal__Proposal_Name__c>
                <urn1:Apttus_Proposal__ReadyToGenerate__c>true</urn1:Apttus_Proposal_
_ReadyToGenerate__c>
                <urn1:Apttus_Proposal__ReadyToPresent__c>true</urn1:Apttus_Proposal__
ReadyToPresent__c>
                <urn1:Apttus_QPConfig__AutoActivateOrder__c>false</urn1:Apttus_QPConf
ig__AutoActivateOrder__c>
                <urn1:Apttus_QPConfig__AutoCreateBill__c>false</urn1:Apttus_QPConfig_
_AutoCreateBill__c>
                <urn1:Apttus_QPConfig__AutoCreateRevenue__c>false</urn1:Apttus_QPConf
ig__AutoCreateRevenue__c>
                <urn1:Apttus_QPConfig__BillingPreferenceId__c>a2ti0000000Fh9SAAS</
urn1:Apttus_QPConfig__BillingPreferenceId__c>
                <urn1:Apttus_QPConfig__DisableCartVersioning__c>false</urn1:Apttus_QP
Config__DisableCartVersioning__c>
                <urn1:Apttus_QPConfig__IsAutoAccepted__c>false</urn1:Apttus_QPConfig_
_IsAutoAccepted__c>
                <urn1:Apttus_QPConfig__IsSystemGenerated__c>false</urn1:Apttus_QPConf
ig__IsSystemGenerated__c>
```

```
                <urn1:Apttus_QPConfig__IsTaskPending__c>false</urn1:Apttus_QPConfig__
IsTaskPending__c>
                <urn1:Apttus_QPConfig__PriceListId__c>a1De0000001yPXQEA2</urn1:Apttus
_QPConfig__PriceListId__c>
                <urn1:Apttus_QPConfig__SourceChannel__c>Direct</urn1:Apttus_QPConfig_
_SourceChannel__c>
                <urn1:Apttus_QPConfig__SyncAssetChangesToQuote__c>true</urn1:Apttus_Q
PConfig__SyncAssetChangesToQuote__c>
                <urn1:Apttus_QPConfig__UseType__c>Main</urn1:Apttus_QPConfig__UseType
__c>
                <urn1:CurrencyIsoCode>USD</urn1:CurrencyIsoCode>
                <urn1:Numeric_Range_Field__c>1</urn1:Numeric_Range_Field__c>
                <urn1:OwnerId>0050U000000r444QAA</urn1:OwnerId>
                <urn1:Quote_Checkbox_21145__c>false</urn1:Quote_Checkbox_21145__c>
                <urn1:RecordTypeId>012i00000001ECbAAM</urn1:RecordTypeId>
                <urn1:Region__c>India</urn1:Region__c>
            </urn:sObjects>
        </urn:create>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ur
n:partner.soap.sforce.com">
   <soapenv:Header>
   </soapenv:Header>
   <soapenv:Body>
      <createResponse>
         <result>
            <id>a0eZ0000005pFF1IAM</id>
            <success>true</success>
         </result>
      </createResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Creating Proposal Line Items from Opportunity

This API creates proposal line items from opportunity line items, creates a task for the action and invokes custom the callback which extends the interface

CustomClass.IQuoteLifecycleCallback2. This API is a callback action after the proposal is created from **createProposalFromOpportunity** API.

API	Signature
afterCreateFromOpportunity	*webService static Boolean afterCreateFromOpportunity(Id proposalId)*

Request Parameter		
Name	**Type**	**Description**
proposalId	ID	The ID of the existing proposal.
Response Parameters		
Field	**Type**	**Description**
isSuccess	Boolean	Indicates whether the callback was successfully executed. The values are *true* and *false*.

Code Sample

The sample code below enables you to create proposal line items from the opportunity and create a new task for given proposalID.

```
/**
 * The below code creates proposal line  items from opportunity and create task for the proposal as callback actions.
 * proposal from createProposalFromOpportunity API.
 */
public Boolean afterCreateFromOpportunity(String proposalName) {
    Boolean callBackStatus = false;
    Apttus_Proposal__Proposal__c proposalSO = [SELECT Id
                FROM Apttus_Proposal__Proposal__c
```

```
9                 WHERE Name =: proposalName LIMIT 1];
10      callBackStatus  =
Apttus_Proposal.ProposalWebService.afterCreateFromOpportunity(proposalSO.I
d);
11      return callBackStatus;
12  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Proposal from an Opportunity

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/
ProposalWebService">
   <soapenv:Header>
      <prop:SessionHeader>
         <prop:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</prop:sessionId>
      </prop:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <prop:afterCreateFromOpportunity>
         <prop:proposalId>a0eZ0000005pF3U</prop:proposalId>
      </prop:afterCreateFromOpportunity>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
```

```
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService">
    <soapenv:Body>
        <afterCreateFromOpportunityResponse>
            <result>true</result>
        </afterCreateFromOpportunityResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Updating a Proposal

This API updates proposal object field values.

API	Signature
updateProposal	*webService static Boolean updateProposal(Apttus_Proposal__Proposal__c proposalSO)*

Request Parameter		
Name	**Type**	**Description**
proposalSO	Apttus_Proposal__Proposal__c	The proposal SObject

Response - Apttus_Proposal_Proposal_c		
Field	**Type**	**Description**
updateStatus	Boolean	Indicated whether the update was successful. The values returned are *true* and *false*.

Code Sample

The sample below enables you to update the proposal object field values.

```
/**
 * The below method demonstrates how to update description for already created proposal
 */
public Boolean updateProposal(String proposalName)
{
    Boolean updateStatus = false;
    Apttus_Proposal__Proposal__c proposalSO = [SELECT Id FROM Apttus_Proposal__Proposal__c WHERE Name = :proposalName LIMIT 1];

    // assign new value to the proposal fields one by one.
    proposalSO.Apttus_Proposal__Description__c = 'Updated through updateProposal API';
    updateStatus = Apttus_Proposal.ProposalWebService.updateProposal(proposalSO);
    return updateStatus;
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService">
    <soapenv:Header>
        <prop:SessionHeader>
            <prop:sessionId>00DZ000000NAEIA!ASAAQBtZlu4.YeuaMdf45TpRUpKEgxgWi30zUbCO161kylPP_RfOfumgPPciWIOobZst4.nxzgnmtZOs3hLjbUCWziADgmbo</prop:sessionId>
        </prop:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
```

```
        <prop:updateProposal>
            <prop:proposalSO>
                <prop:Id>a0eZ0000005pG5N</prop:Id>
                <prop:Apttus_QPConfig__PriceListId__c>a1DZ0000002mg5n</prop:Apttus_QP
Config__PriceListId__c>
                <prop:Apttus_Proposal__Account__c>001Z000001UtoRL</prop:Apttus_Propos
al__Account__c>
            </prop:proposalSO>
        </prop:updateProposal>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService">
    <soapenv:Body>
        <updateProposalResponse>
            <result>true</result>
        </updateProposalResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Cloning Email Template

Clones an email template.

Prerequisite:

The email template you are cloning must exist in your org.

API	Signature
cloneEmailTemplate	*webService static Id cloneEmailTemplate(Id originalId)*

Request Parameter		
Name	**Type**	**Description**
originalId	ID	The ID of email template you want to clone.

Response - Apttus_Proposal_Proposal_c		
Field	**Type**	**Description**
newTemplateId	ID	The ID of the cloned email template.

Code Sample

The sample code below enables you to clone an email template and get the ID of the cloned email template.

```
/**
 * The below code demonstrates how to clone from existing email template.
 */
Public Id cloneMailTemplate(Id originalID) {
    Id newTemplateId =
Apttus_Proposal.ProposalWebService.cloneEmailTemplate(originalID);
    return newTemplateId;
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

None.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/
ProposalWebService">
    <soapenv:Header>
        <prop:SessionHeader>
            <prop:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</prop:sessionId>
        </prop:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <prop:cloneEmailTemplate>
            <prop:originalId>a09i000000TGec8</prop:originalId>
        </prop:cloneEmailTemplate>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService">
    <soapenv:Body>
        <cloneEmailTemplateResponse>
            <result>00XZ0000000o333MAA</result>
        </cloneEmailTemplateResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Cloning Document Templates

This API clones all the attachments and files associated with the source email template and associate them with the newly cloned email template. This API is a callback API of *cloneEmailTemplate* API.

API	Signature
afterEmailTemplateClone	*webService static Boolean afterEmailTemplateClone(Id originalId, Id cloneId)*

Request Parameter		
Name	**Type**	**Description**
originalId	ID	The ID of email template you want to clone.
cloneId	ID	The ID of the cloned proposal where you want to clone the attachments or files.

Response Parameter		
Field	**Type**	**Description**
isSuccess	Boolean	Indicates whether the cloning pf the document template was successful. The values are *true* and *false*.

Code Sample

The sample code below allows you to clone an email template and attachments or files from the parent email template and associate to the newly cloned email template.

```
/**
 * The below code clone email template and the callback function of cloning the document template from the source
 * and associate with the cloned email template. It returns boolean based on the status of after clone action.
 */
public Boolean AfterCloneEmailTemplate(Id  SrourceID) {
   Boolean afterCloneAPIStatus = false;
   Id clonedId = Apttus_Proposal.ProposalWebService.cloneEmailTemplate(SrourceID);
   afterCloneAPIStatus = Apttus_Proposal.ProposalWebService.afterEmailTemplateClone(SrourceID, clonedId);
   return afterCloneAPIStatus;
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Cloning Email Template

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService">
    <soapenv:Header>
        <prop:SessionHeader>
            <prop:sessionId>00DZ000000NAEIA!ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPfmSookHTttOu</prop:sessionId>
        </prop:SessionHeader>
```

```
    </soapenv:Header>
    <soapenv:Body>
        <prop:afterEmailTemplateClone>
            <prop:originalId>a09i000000TGec8</prop:originalId>
            <prop:cloneId>a09Z000000E2y3l</prop:cloneId>
        </prop:afterEmailTemplateClone>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService">
    <soapenv:Body>
        <afterEmailTemplateCloneResponse>
            <result>true</result>
        </afterEmailTemplateCloneResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Getting the Proposal Document Output Format

This API retrieves the proposal document output format for a specific user profile.

API	Signature
getDefaultOutputFormat	*webService static Apttus_Proposal__Proposal_Document_Output_Format__c getDefaultOutputFormat(Id profileId, Id recordTypeId)*

Request Parameter		
Name	**Type**	**Description**
profileId	ID	The ID of the profile.
recordTypeId	ID	Record Type of the proposal object

Response - Apttus_Proposal__Proposal__c		
Field	**Type**	**Description**
Id	ID	Record ID of the proposal.
Name	String	The name of the proposal
Apttus_Proposal__ Profile_Id__c	ID	The Id of the profile.
Apttus_Proposal__Output _Format__c	String	The document output format (For example PDF, RTF or DOC or DOCX)
Apttus_Proposal__Include Watermark__c	Boolean	Indicates whether a watermark is added to the document.
Apttus_Proposal__Allow_ Override__c	Boolean	Indicates whether you are allowed to override the default output format during the quote/proposal document generation process.
Apttus_Proposal__AllowO verrideWatermark__c	Boolean	Indicates whether you are allowed to override the default watermark setting during the quote/proposal document generation process.
Apttus_Proposal__Propos al_Type_Id__c	String	The type of proposal
CurrencyIsoCode	String	The currency defined for the proposal.

Code Sample

The sample code below enabled you to retrieve the proposal document output format for the given user profile and record type name.

```
/**
 * The below code demonstrates how to get the document output format for given profile name.
 */
Public Static Id getProfileID (String profileName)
{
    return [SELECT id FROM profile WHERE Name =: profileName LIMIT 1].Id;
}
public Id getProposalRecordTypeId(String proposalRecordTypeName)
{
    return [SELECT ID
            FROM recordType
            WHERE sObjectType='Apttus_Proposal__Proposal__c'
            AND Name =: proposalRecordTypeName
            LIMIT 1].Id;
}
Public static Apttus_Proposal__Proposal_Document_Output_Format__c getDocumentOutputFormat (String profileName, String recordTypeName )
{
    ID profileID = getProfileID(profileName);
    ID proposalRecordTypeId = getProposalRecordTypeId(recordTypeName);
    Apttus_Proposal__Proposal_Document_Output_Format__c outputFormatSO = new Apttus_Proposal__Proposal_Document_Output_Format__c();
    outputFormatSO = Apttus_Proposal.ProposalWebService.getDefaultOutputFormat();
    return outputFormatSO;
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

None.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:prop="http://soap.sforce.com/schemas/class/Apttus_Proposal/
ProposalWebService">
    <soapenv:Header>
        <prop:SessionHeader>
            <prop:sessionId>00DZ000000NAEIA!
ASAAQKosATSpsGeD7FUh4RDI18xnzIFPe80Mk89ejrAmDrCBY2lmyJKQjKvuwj3TvT71r6g_epvbo6FeqKUPf
mSookHTttOu</prop:sessionId>
        </prop:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <prop:getDefaultOutputFormat>
            <prop:profileId>00eZ0000000Uhkn</prop:profileId>
            <prop:recordTypeId>012i00000001ECb</prop:recordTypeId>
        </prop:getDefaultOutputFormat>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_Proposal/ProposalWebService" xmlns:xsi="htt
p://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <getDefaultOutputFormatResponse>
         <result xsi:type="Apttus_Proposal__Proposal_Document_Output_Format__c">
            <Id>a0bZ0000008cMKnIAM</Id>
            <Apttus_Proposal__AllowOverrideWatermark__c>true</
Apttus_Proposal__AllowOverrideWatermark__c>
            <Apttus_Proposal__Allow_Override__c>true</
Apttus_Proposal__Allow_Override__c>
            <Apttus_Proposal__IncludeWatermark__c>true</
Apttus_Proposal__IncludeWatermark__c>
```

```
            <Apttus_Proposal__Output_Format__c>DOCX</
Apttus_Proposal__Output_Format__c>
            <Apttus_Proposal__Profile_Id__c>00eZ0000000UhknIAC</
Apttus_Proposal__Profile_Id__c>
            <Apttus_Proposal__Proposal_Type_Id__c>012i00000001ECbAAM</
Apttus_Proposal__Proposal_Type_Id__c>
            <Name>a0bZ0000008cMKn</Name>
         </result>
      </getDefaultOutputFormatResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Copying Attachments

This API copies an attachment to the destination record.

API	Signature
copyAttachment	*webService static Boolean copyAttachment(Id destParentId, Id attId)*

Request Parameter		
Name	**Type**	**Description**
destParentId	ID	The ID of the destination parent record.
attId	ID	The ID of the attachment you want to copy.

Response Parameter		
Field	**Type**	**Description**
isSuccess	Boolean	Indicates whether the copy was successful

Code Sample

The sample code below enables you to clone an attachment from any parent record and assign it to the destination SObject record.

```
/**
 * The below code demonstrates how to copy from existing attachment and associate with given parent ID.
 */
public Boolean createAttachment (String attachmentName, String proposalName)
{
    Boolean isSuccess;
    Id proposalSOID = [SELECT Id FROM Apttus_Proposal__Proposal__c WHERE Name = :proposalName LIMIT 1].Id;
    Id attachmentID = [SELECT Id FROM Attachment WHERE Name = :attachmentName LIMIT 1].Id;
    isSuccess = Apttus_Proposal.ProposalWebService.copyAttachment(proposalSOID, attachmentID);
    return isSuccess;
}
```

Creating a Task After Proposal is Presented

This API changes the approval status of a proposal to *Presented* and creates a task.

API	Signature
afterPresent	*webService static Boolean afterPresent(Id proposalId)*

Request Parameter		
Name	**Type**	**Description**
proposalId	ID	The ID of the proposal

Response Parameter		
Field	**Type**	**Description**
isSuccess	Boolean	Indicates whether the approval status was changed and task creation was successful

Code Sample

The sample code below enables you to change the approval status of a proposal to *Presented* and create a task for the given proposal.

```
/**
 * The below code demonstrates the task creation after changing approval status to 'Presented'
 */
public Boolean afterPresent(String proposalName)
{
    Boolean isSuccess;
    Id proposalSOID = [SELECT Id FROM Apttus_Proposal__Proposal__c WHERE Name = :proposalName LIMIT 1].Id;
    isSuccess = Apttus_Proposal.ProposalWebService.afterPresent(proposalSOID);
    return isSuccess;
}
```

Retrieving Name of the First Template

The API retrieves the name of the first template associated with the given proposal.

API	Signature
selectFirstTemplateNameForProposal	*webService static String selectFirstTemplateNameForProposal(Id proposalId)*

Request Parameter		
Name	**Type**	**Description**
proposalId	ID	The ID of the proposal.

Response Parameter		
Field	**Type**	**Description**
templateName	String	The name of the first template.

Code Sample

The sample code below enables you to get the first template name associated to the given proposal.

```
/**
 * The below code gets first template name.
 */
public String selectFirstTemplateNameForProposal(String proposalName)
{
    String templateName;
    Id proposalSOID = [SELECT Id FROM Apttus_Proposal__Proposal__c WHERE Name = :proposalName LIMIT 1].Id;
    templateName = Apttus_Proposal.ProposalWebService.selectFirstTemplateNameForProposal(proposalSOID);
```

```
        return templateName;
    }
```

Retrieving Templates in a Proposal

The API retrieves the templates associated with the given proposal.

API	Signature
getTemplatesForProposal	*webService static List getTemplatesForProposal(Id proposalId, String proposalType)*

Request Parameter		
Name	**Type**	**Description**
proposalId	ID	The ID of the proposal.
proposalType	String	The type of template to query

Response Parameter		
Field	**Type**	**Description**
templateSOs	List<Apttus__APTS_Template__c>	The list of templates matching the configured query criteria.

Code Sample

The sample code below enables you to retrieve the list of templates for the given proposal.

```
/**
 * The below code demonstrates how to get list of template records for the given proposal
 */
public List<Apttus__APTS_Template__c> getTemplatesForProposal(String proposalName)
{
    Id proposalSOID = [SELECT Id FROM Apttus_Proposal__Proposal__c WHERE Name =: proposalName LIMIT 1].Id;
    List<Apttus__APTS_Template__c> templateSOs = new List<Apttus__APTS_Template__c>();
    templateSOs = Apttus_Proposal.ProposalWebService.getTemplatesForProposal(proposalSOID, 'Proposal');
    return templateSOs;
}
```

Deleting an Email Template

The API deletes an email template.

API	Signature
deleteEmailTemplate	*webService static Boolean deleteEmailTemplate(Id templateId)*

Request Parameter		
Name	**Type**	**Description**
templateId	ID	The ID of the email template sobject you want to delete.

Response Parameter		
Field	**Type**	**Description**
isDeleted	Boolean	Indicates whether the email template was deleted.

Code Sample

The sample code below enables you to delete an email template and return the status of the delete action.

```
/**
 * The below code demonstrates how to delete existing email template.
 */
Public Boolean deleteMailTemplate(Id templateId)
{
    Boolean isDeleted =
Apttus_Proposal.ProposalWebService.deleteEmailTemplate(templateId);
    return isDeleted;
}
```

Batch Update Service

The Batch Update Services enable you to update the single and multiple Category Views.

You can invoke APIs in Batch Update Service using the following command:

```
Apttus_CPQApi.BatchUpdateService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

In this section:

- Updating Category Hierarchy for a single Category
- Updating Category Hierarchy for multiple Categories

Updating Category Hierarchy for a single Category

This method allows you to run the Category maintenance for a single category. This is applicable to only those categories which have products associated to them. You can invoke this API when a user clicks **Update View** button in the Category Hierarchy.

API	Signature
updateCategoryView	*static Id updateCategoryView(Id hierarchyId)*

Request Parameter		
Name	**Type**	**Description**
hierarchyId	ID	This is the ID of the category hierarchy record.

Response Parameter		
Name	**Type**	**Description**
apexBatchJobId	ID	The ID of the Batch Job run.

Code sample

The code described below is used to run the Category Maintenance after you associate new products or modify the existing products under a category. You can also use this API when associate a Price List Item to a Category, under the related list, Price List Category. This method takes up the ID of the category hierarchy record and returns the Batch job ID after success.

```
1  public void updateCategoryView(String categoryName)
2  {
3      List<Apttus_Config2_ClassificationName_c> categoryList = [SELECT ID
   FROM Apttus_Config2_ClassificationName_c WHERE Name =: categoryName];
4      String hierarchyId = categoryList[0].ID;
5      ID apexBatchJobId =
   Apttus_CpqApi.BatchUpdateService.updateCategoryView(hierarchyId);
```

```
    ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,
'Apex Batch Job Id : ' + apexBatchJobId));

}
```

Updating Category Hierarchy for multiple Categories

This method allows you to run the Category maintenance for multiple categories. This is applicable to only those categories which have products associated to them. You can invoke this API when a user clicks **Update View** button in the Category Hierarchy.

API	Signature
updateCategoryViews	*static Id updateCategoryViews(Set hierarchyIds)*

Request Parameter

Name	Type	Description
hierarchyIds	Set<ID>	This is the set containing the IDs of the category hierarchy records.

Response Parameter

Name	Type	Description
apexBatchJobId	ID	The ID of the Batch Job run.

Code sample

The code described below is used to run the Category Maintenance for multiple Category Hierarchies after you associate new products or modify the existing products under multiple categories. You can also use this API when associate a Price List Item to a Category, under the related list, Price List Category. This method takes up the set of IDs of the category hierarchy records and returns the Batch job ID after success.

```
public void updateCategoryViews(List<String> categoryNames)
{
```

```
    Set<ID> hierarchyIds = new Set<ID>();
    for(String categoryName : categoryNames)
    {
        List<Apttus_Config2_ClassificationName_c> categoryList = [SELECT ID FROM Apttus_Config2_ClassificationName_c WHERE Name =: categoryName];
        hierarchyIds.add(categoryList[0].ID);
    }

    ID apexBatchJobId = Apttus_CpqApi.BatchUpdateService.updateCategoryViews(hierarchyIds);

    ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info, 'Apex Batch Job Id : ' + apexBatchJobId));
}
```

CPQ Web Service(Apttus_CPQApi)

You can invoke APIs in CPQ Web Service(Apttus_CPQApi) from the following command:

```
Apttus_CPQApi.CPQWebService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the CPQ web service APIs to complete the following tasks:

- Creating a Cart from a Quote
- Retrieving Categories for a Price List
- Retrieving Products and List Prices for a Price List
- Retrieving Products and List Prices for a Price List and Category
- Retrieving Products and List Prices For a Price List and Search Text
- Retrieving Products and List Prices For a Price List Category and Search Text
- Retrieving Option Groups, Options, and List Prices for a Price List Product
- Adding Products to a Cart
- Adding a Bundle to a Cart
- Adding Custom Bundles
- Adding Options to a Bundle
- Cloning Bundle Line Items on the Cart
- Cloning Line Items on the Cart
- Cloning Option Line Items on the Cart
- Comparing Products

Creating a Cart from a Quote

This API creates a cart for the quote or proposal referenced by QuoteID. The quote or proposal must be associated with a price list.

API	Signature
createCart	*webService static Apttus_CPQApi.CPQ.CreateCartResponseDO createCart(Apttus_CPQApi.CPQ.CreateCartRequestDO request)*

Parameters			
Name	**Type**	**Required?**	**Description**
request	Apttus_CPQApi.CPQ.CreateCartRequestDO	Yes	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.CreateCartRequestDO		
Field	**Type**	**Description**
Properties	List<Apttus_CPQApi.CPQ.PropertyDO>	The list of properties applicable to the cart
QuoteID	ID	The id of the quote/proposal to be associated with the cart.

Data Object - Apttus_CPQApi.CPQ.PropertyDO		
Field	**Type**	**Description**
Name	String	Specify the features applicable to the cart. Applicable values are: • useAdvancedApproval: Enables Advanced Approval for a cart. • useDealOptimizer: Enables Deal Optimizer for a cart
Value	String	The applicable values are true or false. Specifying the value as true enables the feature for a cart.

Response Data Object - Apttus_CPQApi.CPQ.CreateCartResponseDO		
Field	**Type**	**Description**
CartId	ID	The ID of the newly created cart object

Code Sample

The sample below enables you to create a cart for a valid quote with a Quote ID. Using the sample below, you can search for a valid quote using a quote number. If a quote exists with the quote number entered, you can create a cart using the createCart API or you will be prompted with a message to enter a valid quote number. You can invoke this API in use cases when you want to show a cart page based on the quote. For example for a realized opportunity, you can create a quote. Based on a valid quote ID, you can create a cart using this API.

```
1
2  /**
3   * The below method demonstrates how to create a cart for a quote
4   */
5  public static void createCart(String quoteNumber)
6  {
```

```
    List<Apttus_Proposal__Proposal__c> quote = [SELECT Id FROM Apttus_Proposal__Proposal__c WHERE Name = :quoteNumber LIMIT 1];

    if(!quote.isEmpty())
    {

        // Create the request object
        Apttus_CPQApi.CPQ.CreateCartRequestDO request = new Apttus_CPQApi.CPQ.CreateCartRequestDO();
        List<Apttus_Config2.Property> Properties = new List<Apttus_Config2.Property>();
        Properties.add(new Apttus_Config2.Property('useAdvancedApproval','false'));
        Properties.add(new Apttus_Config2.Property('isCartVersioningDisabled,'true'));

        request.QuoteId = quote.get(0).Id;
        request.Properties = Properties;
        // Excute the createCart routine
        Apttus_CPQApi.CPQ.CreateCartResponseDO response = Apttus_CPQApi.CPQWebService.createCart(request);
        System.debug('Cart has been successfully created. CartId = ' + response.CartId);
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

None.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00D4P000000z7dE!
AQQAQLOWqt9rvkoE67lP.NXIamuJl0mjoOR6a_l0vkPoATAH.gKFFRWManncKXN0KGY92akqlxsMOzeOlRfEr
g4V2LtI03XQ</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:createCart>
            <cpq:request>
                <cpq1:QuoteId>a0c4P00000GKQzL</cpq1:QuoteId>
            </cpq:request>
        </cpq:createCart>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:CreateCartResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
    <soapenv:Body>
        <createCartResponse>
            <result>
                <CreateCartResponseDO:CartId>a1OZ0000002YK9rMAG</CreateCartResponseDO:
CartId>
            </result>
        </createCartResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Categories for a Price List

This API is used to get the unique list of categories for all the products within the specified price list.

API	Signature
getCategoriesForPriceList	*webService static Apttus_CPQApi.CPQ.CategorySearchResultDO getCategoriesForPriceList(Id priceListId)*

Parameters			
Name	**Type**	**Required?**	**Description**
priceListId	ID	Yes	The id of the price list

Response Data Object - Apttus_CPQApi.CPQ.CategorySearchResultDO		
Field	**Type**	**Description**
Categories	List<Apttus_CPQApi.CPQ.CategoryDO>	The list of category data objects.
HasCategories	Boolean	Indicates if there are categories for the price list.

Data Object - Apttus_CPQApi.CPQ.CategoryDO		
Field	**Type**	**Description**
CategoryId	ID	The Id of the category.
ChildCategories	List<Apttus_CPQApi.CPQ.CategoryDO>	The list of child category data objects associated with the category.
HasChildCategories	Boolean	Indicates if the category has child categories.
Name	String	The category name.
ParentCategoryId	ID	The Id of the parent category.
ProductCount	Integer	The number of products within a category.

Code Sample

The code sample below enables you to search for categories based on the pricelist ID. All the categories and subcategories associated with the price list are displayed. You can search for categories associated with a price list till the nth level. You can provide a search field on the cart page that enables the user to search for products by categories. For example, your cart page has multiple categories, such as Laptop, camera, Desktop, Accessories and so on. The user should be able to search for products by category by entering category name such as laptop. Use this API to invoke the categories and its associated sub-categories of products and then display it to the user.

```
1  public void executeSearch()
2  {
3  //Query for fetching pricelist id by querying price list name
4       List<Apttus_Config2__PriceList__c> priceListItemList = [select id
5       from Apttus_Config2__PriceList__c
6       where  name = :priceListName limit 1];
7
8  //If an id is returned in the list execute the getCategoriesforpricelist
   API.
9        if(priceListItemList.size() > 0)
```

```
10          {
11              priceListId = priceListItemList[0].ID;
12              Apttus_CPQApi.CPQ.CategorySearchResultDO result =
    Apttus_CPQApi.CPQWebService.getCategoriesForPriceList(priceListId);
13              lstwrap = New List<CategoryWrapperClass>();
14              For( Apttus_CPQApi.CPQ.CategoryDO catresult :
    result.Categories)
15              {
16                  CategoryWrapperClass wrap = New CategoryWrapperClass();
17                  wrap.CategoryId = catresult.CategoryId;
18                  wrap.categoryName = catresult.name;
19                  lstwrap.add(wrap);
20                  //If a category has sub categories fetch the sub-category
    name and id
21                  if(catresult.HasChildCategories)
22                  {
23                      For( Apttus_CPQApi.CPQ.CategoryDO subcatresult :
    catresult.ChildCategories)
24                      {
25                          CategoryWrapperClass subwrap = New
    CategoryWrapperClass();
26                          subwrap.CategoryId = subcatresult.CategoryId;
27                          subwrap.categoryName = subcatresult.name;
28                          lstwrap.add(subwrap);
29                          // If the sub category has child categories fetch
    the sub category name and ID.
30                          if(subcatresult.HasChildCategories)
31                          {
32                              For( Apttus_CPQApi.CPQ.CategoryDO
    subsubcatresult : subcatresult.ChildCategories)
33                              {
34                                  CategoryWrapperClass subsubwrap = New
    CategoryWrapperClass();
35                                  subsubwrap.CategoryId =
    subsubcatresult.CategoryId;
36                                  subsubwrap.categoryName =
    subsubcatresult.name;
37                                  lstwrap.add(subsubwrap);
38                              }
39                          }
40                      }
41                  }
42              }
```

```
43          }
44      //If no Price List exists with the searched string name execute the
    else condition
45      else
46      {
47          lstwrap = New List<CategoryWrapperClass>();
48          ApexPages.addMessage(new
     ApexPages.Message(ApexPages.severity.info, 'Category Name not found.
    Please enter valid Category Name.'));
49      }
50  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQN7k3xhhvMe.j.8gpR.ijcGq77HJeF4MEDfbxbBAqZ8r4WWNTv3OVb6o1bjtHJLbq5mvHcuAH_ie6sTC3
DzrWgnLdUuD</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:getCategoriesForPriceList>
            <cpq:priceListId>a1DZ0000002mg5n</cpq:priceListId>
        </cpq:getCategoriesForPriceList>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:CategorySearchResultDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getCategoriesForPriceListResponse>
            <result>
                <CategorySearchResultDO:Categories>
                    <CategorySearchResultDO:CategoryId>a0nZ0000003x6pLIAQ</
CategorySearchResultDO:CategoryId>
                    <CategorySearchResultDO:HasChildCategories>false</
CategorySearchResultDO:HasChildCategories>
                    <CategorySearchResultDO:Name>Auto_API_Category</
CategorySearchResultDO:Name>
                    <CategorySearchResultDO:ParentCategoryId xsi:nil="true"/>
                    <CategorySearchResultDO:ProductCount>15</CategorySearchResultDO:P
roductCount>
                </CategorySearchResultDO:Categories>
                <CategorySearchResultDO:HasCategories>true</CategorySearchResultDO:Ha
sCategories>
            </result>
        </getCategoriesForPriceListResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Products and List Prices for a Price List

This API is used to get the list of all the products within the specified price list. This API automatically takes into consideration product visibility rules and will enforce them if applicable. For more information, see Configuring Product Visibility.

> ⓘ This API works only when you have associated a price list and price list items to a category. This API also considers the based on price list scenario.

API	Signature
getProductsForPriceList	*webService static Apttus_CPQApi.CPQ.ProductSearchResultDO getProductsForPriceList(Id priceListId)*

Parameters		
Name	**Type**	**Description**
priceListId	ID	The id of the price list.

Response Data Object - Apttus_CPQApi.CPQ.ProductSearchResultDO		
Field	**Type**	**Description**
HasProducts	Boolean	This returns true if the list of product data objects is not empty.
Products	List<Apttus_CPQApi.CPQ.ProductDO>	The list of product data objects.

Data Object - Apttus_CPQApi.CPQ.ProductDO		
Field	**Type**	**Description**
ContentUrl	String	The product content location.
Description	String	The product description.
HasPrices	Boolean	Indicates if there are list prices for the product.

Data Object - Apttus_CPQApi.CPQ.ProductDO		
Field	**Type**	**Description**
HasProperties	Boolean	Indicates if there are properties set for the product.
ImageUrl	String	The location of the image, if there is one, associated with the product.
Name	String	The product name.
Prices	List<Apttus_CPQApi.CPQ.PriceDO>	The list of price data objects.
ProductCode	String	The product code.
ProductId	ID	The Id of the product.
Property	List<Apttus_CPQApi.CPQ.PropertyDO>	The list of property data objects.

Data Object - Apttus_CPQApi.CPQ.PriceDO		
Field	**Type**	**Description**
ChargeType	String	The charge type.
PriceItem	Apttus_Config2__PriceListItem__c object	The price list items for a price list.
Value	Decimal	The list price.

Data Object - Apttus_CPQApi.CPQ.PropertyDO		
Field	**Type**	**Description**
Name	String	The name of the property.
Value	Decimal	The value of a property.

Code Sample

Using the sample you can show the user the list of products for a particular price list. For example if you have a price list created for a particular company, the user can search the price list by name and view all the products associated with that price list. For example, when a user searches for badger price list and clicks search, invoke the getProductsForPriceList API in which you pass the pricelistID as a parameter. Fetch and then display all the product components to the user such as Machinery, standard price, quantity and name and description of the product.

```
1  public void getProductsForPriceList()
2  {
3      //If the priceList name by which the user searches the price list
   exists fetch the ID.
4      if(priceListId == null || priceListId== '0')
5      {
6          List<Apttus_Config2__PriceList__c> priceListItemList = [select id
   from Apttus_Config2__PriceList__c where  Name = :priceListName limit 1];
7          //If the priceListItem exists, the list size >0 assign the ID at
   the 0th position of the list to priceListId
8          if(priceListItemList.size() > 0)
9          {
10             priceListId = priceListItemList[0].ID;
11         }
12         else
13         {
14             lstProductwrapAll = New List<ProductWrapperClass>();
15             lstProductwrap = New List<ProductWrapperClass>();
16             return;
17         }
18     }
19
```

```
20      //Fetch id
21      Apttus_CPQApi.CPQ.ProductSearchResultDO productResult =
    Apttus_CPQApi.CPQWebService.getProductsForPriceList(priceListId);
22      productCount = 'Product Count: ' + productResult.Products.size();
23      lstProductwrapAll = New List<ProductWrapperClass>();
24      lstProductwrap = New List<ProductWrapperClass>();
25
26      //For the fetched pricelistID fetch and display the following
27      For(Apttus_CPQApi.CPQ.ProductDO catresult : productResult.Products)
28      {
29          ProductWrapperClass wrap = New ProductWrapperClass ();
30          wrap.ProductId = catresult.ProductId;
31          wrap.ProductCode= catresult.ProductCode;
32          wrap.ProductName=catresult.Name;
33          wrap.Description=catresult.Description;
34          wrap.ImageUrl=catresult.ImageUrl;
35          wrap.ContentUrl=catresult.ContentUrl;
36          wrap.HasPrices=catresult.HasPrices;
37          wrap.Prices=catresult.Prices;
38          wrap.Quantity=1;
39          lstProductwrapAll.add(wrap);
40      }
41  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00DZ000000NAEIA!
ASAAQN7k3xhhvMe.j.8gpR.ijcGq77HJeF4MEDfbxbBAqZ8r4WWNTv3OVb6o1bjtHJLbq5mvHcuAH_ie6sTC3
DzrWgnLdUuD</cpq:sessionId>
```

```
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:getProductsForPriceList>
         <cpq:priceListId>a1De0000001yPXQ</cpq:priceListId>
      </cpq:getProductsForPriceList>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:ProductSearchResultDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getProductsForPriceListResponse>
            <result>
                <ProductSearchResultDO:HasProducts>true</ProductSearchResultDO:HasPro
ducts>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>AutoQuoteBundle1</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Ce0000003fQ1QEAU</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
```

```
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>100.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1De0000001yPXQEA2</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__ProductId__c>01te0000005NGpDAAW</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01te0000005NGpDAAW</Id>
                                <Name>AutoQuoteBundle1</Name>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000538767</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>100.00000</ProductSearchResultDO:
Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode xsi:nil="true"/>
                    <ProductSearchResultDO:ProductId>01te0000005NGpDAAW</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>AutoQuoteBundle2</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
```

```
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Ce0000003fQ2EEAU</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>100.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1De0000001yPXQEA2</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__ProductId__c>01te0000005NGpNAAW</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01te0000005NGpNAAW</Id>
                                <Name>AutoQuoteBundle2</Name>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000538772</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>100.00000</ProductSearchResultDO:
Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode xsi:nil="true"/>
                    <ProductSearchResultDO:ProductId>01te0000005NGpNAAW</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
            </result>
        </getProductsForPriceListResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Products and List Prices for a Price List and Category

This API is used to get the list of products and related list prices for a specified category in the price list. This API automatically takes into consideration product visibility rules and will enforce them if applicable. For more information, see Configuring Product Visibility.

API	Signature
getProductsForPriceListCategory	*webService static Apttus_CPQApi.CPQ.ProductSearchResultDO getProductsForPriceListCategory(Id priceListId, Id categoryId)*

Parameters		
Name	**Type**	**Description**
priceListId	ID	The id of the price list.
categoryID	ID	The id of the category.

Response Data Object - Apttus_CPQApi.CPQ.ProductSearchResultDO		
Field	**Type**	**Description**
HasProducts	Boolean	This returns true if the list of product data objects is not empty.
Products	List<Apttus_CPQApi.CPQ.ProductDO>	The list of product data objects.

Data Object - Apttus_CPQApi.CPQ.ProductDO		
Field	**Type**	**Description**
ContentUrl	String	The product content location.
Description	String	The product description.
HasPrices	Boolean	Indicates if there are list prices for the product.
HasProperties	Boolean	Indicates if there are properties set for the product.
ImageUrl	String	The location of the image, if there is one, associated with the product.
Name	String	The product name.
Prices	List<Apttus_CPQApi.CPQ.PriceDO>	The list of price data objects.
ProductCode	String	The product code.
ProductId	ID	The Id of the product.
Property	List<Apttus_CPQApi.CPQ.PropertyDO>	The list of property data objects.

Data Object - Apttus_CPQApi.CPQ.PriceDO		
Field	**Type**	**Description**
ChargeType	String	The charge type.

Data Object - Apttus_CPQApi.CPQ.PriceDO		
Field	**Type**	**Description**
PriceItem	Apttus_Config2__PriceListItem__c object	The price list items for a price list.
Value	Decimal	The list price.

Data Object - Apttus_CPQApi.CPQ.PropertyDO		
Field	**Type**	**Description**
Name	String	The name of the property.
Value	Decimal	The value of a property.

Code Sample

The sample below enables you to search for categories using price list name. Once the user selects the category and proceeds to search, pass the pricelistID and categoryID as parameters to the API. The user can select and view the products associated with that category and it components. For example, user enters the price list name and selects the category-Hardware. All the products associated with the Hardware category, such as Laptop are displayed. You can display all the fields associated to that product.

```
public void getProductList()
{
    categoryId ='';
    for(CategoryWrapperClass wrap: lstwrap )
    {
        if(wrap.selected == true)
        {
            categoryId = categoryId + wrap.CategoryId +',';
        }
    }
```

```
11      //If no category is selected prompt the user with an appropriate error
    message.
12      if(categoryId.Trim()=='')
13      {
14          ApexPages.addMessage(new
     ApexPages.Message(ApexPages.severity.info, 'Please select at least one
    Category.'));
15          lstProductwrap = New List<ProductWrapperClass>();
16          lstProductwrapAll = New List<ProductWrapperClass>();
17      }
18      //Pass the pricelistID and categoryID as parameters to the API
19      else
20      {
21          Apttus_CPQApi.CPQ.ProductSearchResultDO productResult =
    Apttus_CPQApi.CPQWebService.getProductsForPriceListCategory(priceListId,ca
    tegoryId.Substring(0,categoryId.length()-1));
22
23          productCount = 'Product Count: ' + productResult.Products.size();
24
25          lstProductwrap = New List<ProductWrapperClass>();
26          lstProductwrapAll = New List<ProductWrapperClass>();
27          For(Apttus_CPQApi.CPQ.ProductDO catresult :
    productResult.Products)
28          //Fetch and display the following product fields to the user
29          {
30              ProductWrapperClass wrap = New ProductWrapperClass ();
31              wrap.ProductId = catresult.ProductId;
32              wrap.ProductCode= catresult.ProductCode;
33              wrap.ProductName=catresult.Name;
34              wrap.Description=catresult.Description;
35              wrap.ImageUrl=catresult.ImageUrl;
36              wrap.ContentUrl=catresult.ContentUrl;
37              wrap.HasPrices=catresult.HasPrices;
38              wrap.Prices=catresult.Prices;
39              wrap.Quantity=1;
40              lstProductwrapAll.add(wrap);
41          }
42
43      }
44  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQN7k3xhhvMe.j.8gpR.ijcGq77HJeF4MEDfbxbBAqZ8r4WWNTv3OVb6o1bjtHJLbq5mvHcuAH_ie6sTC3
DzrWgnLdUuD</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:getProductsForPriceListCategory>
            <cpq:priceListId>a1DZ0000002mg5n</cpq:priceListId>
            <cpq:categoryId>a0nZ0000003x6pL</cpq:categoryId>
        </cpq:getProductsForPriceListCategory>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:ProductSearchResultDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getProductsForPriceListCategoryResponse>
            <result>
```

```
                <ProductSearchResultDO:HasProducts>true</ProductSearchResultDO:HasPro
ducts>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>Auto_API_BundleProduct1</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1CZ000000KOraoMAD</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>10.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1DZ0000002mg5nMAA</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__ProductId__c>01tZ0000004lUXtIAM</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01tZ0000004lUXtIAM</Id>
                                <Name>Auto_API_BundleProduct1</Name>
                            </Apttus_Config2__ProductId__r>
```

```
                            <Name>PI-0000539885</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>10.00000</ProductSearchResultDO:
Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode xsi:nil="true"/>
                    <ProductSearchResultDO:ProductId>01tZ0000004lUXtIAM</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>Auto_API_BundleProduct2</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1CZ000000KOrajMAD</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>10.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1DZ0000002mg5nMAA</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
```

```
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__ProductId__c>01tZ0000004lUXyIAM</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01tZ0000004lUXyIAM</Id>
                                <Name>Auto_API_BundleProduct2</Name>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000539884</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>10.00000</ProductSearchResultDO:
Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode xsi:nil="true"/>
                    <ProductSearchResultDO:ProductId>01tZ0000004lUXyIAM</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
            </result>
        </getProductsForPriceListCategoryResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Products and List Prices For a Price List and Search Text

This API is used to get the list of products that match search text criteria you used for the products in the price list. This API automatically takes into consideration product visibility rules and will enforce them if applicable. For more information, see Configuring Product Visibility.

API	Signature
getProductsForSearchText	*webService static Apttus_CPQApi.CPQ.ProductSearchResultDO getProductsForSearchText(Id priceListId, String searchText)*

Parameters		
Name	**Type**	**Description**
priceListId	ID	The id of the price list.
searchText	String	The search terms that will be used to retrieve the products for the price list.

Response Data Object - Apttus_CPQApi.CPQ.ProductSearchResultDO		
Field	**Type**	**Description**
HasProducts	Boolean	This returns true if the list of product data objects is not empty.
Products	List<Apttus_CPQApi.CPQ.ProductDO>	The list of product data objects.

Data Object - Apttus_CPQApi.CPQ.ProductDO		
Field	**Type**	**Description**
ContentUrl	String	The product content location.
Description	String	The product description.
HasPrices	Boolean	Indicates if there are list prices for the product.
HasProperties	Boolean	Indicates if there are properties set for the product.

Data Object - Apttus_CPQApi.CPQ.ProductDO		
Field	**Type**	**Description**
ImageUrl	String	The location of the image, if there is one, associated with the product.
Name	String	The product name.
Prices	List<Apttus_CPQApi.CPQ.PriceDO>	The list of price data objects.
ProductCode	String	The product code.
ProductId	ID	The Id of the product.
Property	List<Apttus_CPQApi.CPQ.PropertyDO>	The list of property data objects.

Data Object - Apttus_CPQApi.CPQ.PriceDO		
Field	**Type**	**Description**
ChargeType	String	The charge type.
PriceItem	Apttus_Config2__PriceListItem__c	The price list items for a price list.
Value	Decimal	The list price.

Data Object - Apttus_CPQApi.CPQ.PropertyDO		
Field	**Type**	**Description**
Name	String	The name of the property.
Value	String	The value of a property.

Code Sample

Using the sample below you can search for products using the price list id and product name search string. For example, you can provide to search fields for the user one for price list and the other for Search Text. Once the user enters the price list name, fetch the price list ID using the SOQL query. The user then types in the Product Name and clicks Search. Invoke the API where you pass the priceListID and the corresponding search string as parameters. The response object returns the product object fields. For example, if the user enters the name of the Hardware price list and the name of the product, such as Laptop and clicks Search, invoke the API and pass the priceListId and the product name as parameters of the API and fetch and display the product information.

```
class ProductWrapperClass
{
    Id ProductId;
    String ProductCode;
    String ProductName;
    String Description;
    String ImageUrl;
    String ContentUrl;
    Boolean HasPrices;
    List<Apttus_CPQApi.CPQ.PriceDO> Prices;
}

public void getProductsForSearchText(Id priceListId, String productName) {


    //Pass the pricelist ID and the Product Name entered in the search text as parameters to the API

    Apttus_CPQApi.CPQ.ProductSearchResultDO productResult = Apttus_CPQApi.CPQWebService.getProductsForSearchText(priceListId, productName);
    String productCount = 'Product Count: ' + productResult.Products.size();
    List<ProductWrapperClass> lstProductwrapAll = New List<ProductWrapperClass>();
    For(Apttus_CPQApi.CPQ.ProductDO catresult : productResult.Products)

    {
        ProductWrapperClass wrap = New ProductWrapperClass();
```

```
25          wrap.ProductId = catresult.ProductId;
26          wrap.ProductCode= catresult.ProductCode;
27          wrap.ProductName=catresult.Name;
28          wrap.Description=catresult.Description;
29          wrap.ImageUrl=catresult.ImageUrl;
30          wrap.ContentUrl=catresult.ContentUrl;
31          wrap.HasPrices=catresult.HasPrices;
32          wrap.Prices=catresult.Prices;
33          lstProductwrapAll.add(wrap);
34      }
35      ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,
    'Product Count : ' + productCount));
36
37      ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,
    'Product Data Object : ' + lstProductwrapAll));
38
39  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00D4P000000z7dE!
AQQAQGL6XYy.QsjqQQ6RTBnh.1ApTbiqkGAdVz9BS70lxobcyXgHHplmGXAe7p_cf6ziWJ8tpQt_4Q4Bi2VtY
eMyzjhaPbf0</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:getProductsForSearchText>
            <cpq:priceListId>a1a4P00000D3AVG</cpq:priceListId>
            <cpq:searchText>server</cpq:searchText>
        </cpq:getProductsForSearchText>
```

```
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:ProductSearchResultDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getProductsForSearchTextResponse>
            <result>
                <ProductSearchResultDO:HasProducts>true</ProductSearchResultDO:HasPro
ducts>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>BL460c Gen8 Server Blade</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIFjUAP</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
```

```
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>2939.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfJ0QAK</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfJ0QAK</Id>
                                <Family>Hardware</Family>
                                <Name>BL460c Gen8 Server Blade</Name>
                                <ProductCode>HW-BL002</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000351</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>2939.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-BL002</
ProductSearchResultDO:ProductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfJ0QAK</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>BL660c Gen8 Server Blade</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
```

```
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIGIUA5</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>7786.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfKsQAK</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfKsQAK</Id>
                                <Family>Hardware</Family>
                                <Name>BL660c Gen8 Server Blade</Name>
                                <ProductCode>HW-BL004</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000386</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>7786.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-BL004</
ProductSearchResultDO:ProductCode>
```

```
                    <ProductSearchResultDO:ProductId>01t4P000008OfKsQAK</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>Cloud Server Solution</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIFUUA5</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>45999.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfJDQA0</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
```

```
                                <Id>01t4P000008OfJDQA0</Id>
                                <Name>Cloud Server Solution</Name>
                                <ProductCode>CSS</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000324</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>45999.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>CSS</ProductSearchResultDO:Pro
ductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfJDQA0</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>DL320e Gen8 Hot Plug 4LFF CTO
Server</ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIGYUA5</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>1056.00000</
Apttus_Config2__ListPrice__c>
```

```
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfL7QAK</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfL7QAK</Id>
                                <Family>Hardware</Family>
                                <Name>DL320e Gen8 Hot Plug 4LFF CTO Server</Name>
                                <ProductCode>HW-DL006</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000402</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>1056.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-DL006</
ProductSearchResultDO:ProductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfL7QAK</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>DL360p Gen8 10-SFF CTO Server</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIGFUA5</Id>
```

```
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>4467.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfKrQAK</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfKrQAK</Id>
                                <Family>Hardware</Family>
                                <Name>DL360p Gen8 10-SFF CTO Server</Name>
                                <ProductCode>HW-DL008</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000383</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>4467.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-DL008</
ProductSearchResultDO:ProductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfKrQAK</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
```

```
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>DL380p Gen8 E-Mail or Messaging
Server Up to 1000 User Records</ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIG8UAP</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>8090.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfKlQAK</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfKlQAK</Id>
                                <Family>Hardware</Family>
                                <Name>DL380p Gen8 E-Mail or Messaging Server Up to
1000 User Records</Name>
                                <ProductCode>HW-DL011</ProductCode>
                            </Apttus_Config2__ProductId__r>
```

```
                            <Name>PI-0000000376</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>8090.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-DL011</
ProductSearchResultDO:ProductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfKlQAK</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>DL385 Gen8 File and Print Server
Recommended</ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIFmUAP</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>4631.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
```

```
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfKPQA0</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfKPQA0</Id>
                                <Family>Hardware</Family>
                                <Name>DL385 Gen8 File and Print Server Recommended</
Name>
                                <ProductCode>HW-DL012</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000354</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>4631.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-DL012</
ProductSearchResultDO:ProductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfKPQA0</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>ML350e Gen8 Hot Plug 6LFF CTO
Server</ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIGHUA5</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
```

```
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>2666.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfJ9QAK</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfJ9QAK</Id>
                                <Family>Hardware</Family>
                                <Name>ML350e Gen8 Hot Plug 6LFF CTO Server</Name>
                                <ProductCode>HW-ML001</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000385</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>2666.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-ML001</
ProductSearchResultDO:ProductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfJ9QAK</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
```

```
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>WS460c Gen8 Graphics Server Blade</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Z4P00000EDIGBUA5</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>4954.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                            <Apttus_Config2__ProductId__c>01t4P000008OfJ8QAK</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01t4P000008OfJ8QAK</Id>
                                <Family>Hardware</Family>
                                <Name>WS460c Gen8 Graphics Server Blade</Name>
                                <ProductCode>HW-BL003</ProductCode>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000000379</Name>
                        </ProductSearchResultDO:PriceItem>
```

```
                        <ProductSearchResultDO:Value>4954.00000</
ProductSearchResultDO:Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode>HW-BL003</
ProductSearchResultDO:ProductCode>
                    <ProductSearchResultDO:ProductId>01t4P000008OfJ8QAK</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
            </result>
        </getProductsForSearchTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Products and List Prices For a Price List Category and Search Text

This API is used to get the list of products that match search text criteria you used and belong to the specified category id in the price list. This API automatically takes into consideration product visibility rules and will enforce them if applicable. For more information, see Configuring Product Visibility.

API	Signature
getProductsForCategorySearchText	*webService static Apttus_CPQApi.CPQ.ProductSearchResultDO getProductsForCategorySearchText(Id priceListId, Id categoryId, String searchText)*

Parameters		
Name	**Type**	**Description**
priceListId	ID	The id of the price list.
categoryId	ID	The id of the category.

Parameters		
Name	**Type**	**Description**
searchText	String	The search terms that will be used to retrieve the products for the price list.

Response Data Object - Apttus_CPQApi.CPQ.ProductSearchResultDO		
Field	**Type**	**Description**
HasProducts	Boolean	This returns true if the list of product data objects is not empty.
Products	List<Apttus_CPQApi.CPQ.ProductDO>	The list of product data objects.

Data Object - Apttus_CPQApi.CPQ.ProductDO		
Field	**Type**	**Description**
ContentUrl	String	The product content location.
Description	String	The product description.
HasPrices	Boolean	Indicates if there are list prices for the product.
HasProperties	Boolean	Indicates if there are properties set for the product.
ImageUrl	String	The location of the image, if there is one, associated with the product.
Name	String	The product name.

Data Object - Apttus_CPQApi.CPQ.ProductDO		
Field	**Type**	**Description**
Prices	List<Apttus_CPQApi.CPQ.PriceDO>	The list of price data objects.
ProductCode	String	The product code.
ProductId	ID	The Id of the product.
Property	List<Apttus_CPQApi.CPQ.PropertyDO>	The list of property data objects.

Data Object - Apttus_CPQApi.CPQ.PriceDO		
Field	**Type**	**Description**
ChargeType	String	The charge type.
PriceItem	Apttus_Config2__PriceListItem__c object	The price list items for a price list.
Value	Decimal	The list price.

Data Object - Apttus_CPQApi.CPQ.PropertyDO		
Field	**Type**	**Description**
Name	String	The name of the property.
Value	Decimal	The value of a property.

Code Sample

The user can enter the name of price list to view all the categories belonging to that price list. The sample below defines the behavior when the user selects a category and searches a product from the categories displayed. Fetch and pass the priceListID, categoryID, and productID as parameters to the API and display the fields of CPQ.ProductDo object. For example, the user enters a valid price list name, all the subsequent categories for the pricelist are displayed. The user selects the category Hardware and searches for a product Laptop. All the products matching the search string laptop are displayed along with the fields such as price, name , id and so on. You can execute the count loop to specify the number of products to be displayed per page and to provide next and previous pages.

```
public void getProductsForCategorySearchText()
{
    //Search for single category multiple category using comma separator and fetch category id
    categoryId ='';

    for(CategoryWrapperClass wrap: lstwrap )
    {
        if(wrap.selected == true)
        {
            categoryId = categoryId + wrap.CategoryId +',';
        }
    }
    //If no category selected is selected by the user, show the following message
    if(categoryId.Trim()=='')
    {
        ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info, 'Please select at least one Category.'));
        lstProductwrap = New List<ProductWrapperClass>();
        lstProductwrapAll = New List<ProductWrapperClass>();
    }
    //If searched price list is valid, categories for that pricelist are displayed and if the user selects a category and enters search text by 'productName' parameter, pass the PricelistID, categoryID, and productID as parameters.
    else
    {
        Apttus_CPQApi.CPQ.ProductSearchResultDO productResult = Apttus_CPQApi.CPQWebService.getProductsForCategorySearchText(priceListId,categoryId.Substring(0,categoryId.length()-1),productName);
```

```
        productCount = 'Product Count: ' + productResult.Products.size();
        lstProductwrap = New List<ProductWrapperClass>();
        lstProductwrapAll = New List<ProductWrapperClass>();
        For(Apttus_CPQApi.CPQ.ProductDO catresult :
productResult.Products)
        {
            // Fetch and display the following components for the products
            ProductWrapperClass wrap = New ProductWrapperClass ();
            wrap.ProductId = catresult.ProductId;
            wrap.ProductCode= catresult.ProductCode;
            wrap.ProductName=catresult.Name;
            wrap.Description=catresult.Description;
            wrap.ImageUrl=catresult.ImageUrl;
            wrap.ContentUrl=catresult.ContentUrl;
            wrap.HasPrices=catresult.HasPrices;
            wrap.Quantity=1;
            wrap.Prices=catresult.Prices;
            lstProductwrapAll.add(wrap);
        }

    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00D4P000000z7dE!
AQQAQGL6XYy.QsjqQQ6RTBnh.1ApTbiqkGAdVz9BS70lxobcyXgHHplmGXAe7p_cf6ziWJ8tpQt_4Q4Bi2VtY
eMyzjhaPbf0</cpq:sessionId>
```

```
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:getProductsForCategorySearchText>
            <cpq:priceListId>a1a4P00000D3AVG</cpq:priceListId>
            <cpq:categoryId>a0nZ0000003x6pL</cpq:categoryId>
            <cpq:searchText>server</cpq:searchText>
        </cpq:getProductsForCategorySearchText>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:ProductSearchResultDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getProductsForCategorySearchTextResponse>
            <result>
                <ProductSearchResultDO:HasProducts>true</ProductSearchResultDO:HasPro
ducts>
                <ProductSearchResultDO:Products>
                    <ProductSearchResultDO:ContentUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Description xsi:nil="true"/>
                    <ProductSearchResultDO:HasPrices>true</ProductSearchResultDO:HasP
rices>
                    <ProductSearchResultDO:HasProperties>false</ProductSearchResultDO:
HasProperties>
                    <ProductSearchResultDO:ImageUrl xsi:nil="true"/>
                    <ProductSearchResultDO:Name>AutoQuoteStandalone1</
ProductSearchResultDO:Name>
                    <ProductSearchResultDO:Prices>
                        <ProductSearchResultDO:ChargeType>Standard Price</
ProductSearchResultDO:ChargeType>
                        <ProductSearchResultDO:PriceItem xsi:type="Apttus_Config2__Pr
iceListItem__c">
                            <Id>a1Ce0000003fQ0XEAU</Id>
                            <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
```

```
                            <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                            <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                            <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                            <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                            <Apttus_Config2__ListPrice__c>100.00000</
Apttus_Config2__ListPrice__c>
                            <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                            <Apttus_Config2__PriceListId__c>a1De0000001yPXQEA2</
Apttus_Config2__PriceListId__c>
                            <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                            <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                            <Apttus_Config2__ProductId__c>01te0000005NGoUAAW</
Apttus_Config2__ProductId__c>
                            <Apttus_Config2__ProductId__r xsi:type="Product2">
                                <Id>01te0000005NGoUAAW</Id>
                                <Name>AutoQuoteStandalone1</Name>
                            </Apttus_Config2__ProductId__r>
                            <Name>PI-0000538756</Name>
                        </ProductSearchResultDO:PriceItem>
                        <ProductSearchResultDO:Value>100.00000</ProductSearchResultDO:
Value>
                    </ProductSearchResultDO:Prices>
                    <ProductSearchResultDO:ProductCode xsi:nil="true"/>
                    <ProductSearchResultDO:ProductId>01te0000005NGoUAAW</
ProductSearchResultDO:ProductId>
                </ProductSearchResultDO:Products>
            </result>
        </getProductsForCategorySearchTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Option Groups, Options, and List Prices for a Price List Product

This retrieves option groups, options, and list prices for a price list product.

API	Signature
getOptionGroupsForPriceListProduct	*webService static Apttus_CPQApi.CPQ.ProductOptionGroupSearchResultDO getOptionGroupsForPriceListProduct(Id priceListId, Id productId)*

Parameters		
Name	**Type**	**Description**
pricelistId	ID	The Id of the price list.
productId	ID	The Id of the product bundle.

Response Data Object - Apttus_CPQApi.CPQ.ProductOptionGroupSearchResultDO		
Name	**Type**	**Description**
HasOptionGroups	Boolean	Indicates if the bundle has option groups.
OptionGroups	List<Apttus_CPQApi.CPQ.ProductOptionGroupDO>	The list of product option group data objects.

Data Object - Apttus_CPQApi.CPQ.ProductOptionGroupDO		
Field	**Type**	**Description**
ChildOptionGroups	List<Apttus_CPQApi.CPQ.ProductOptionGroupDO>	List of child option group data objects.
HasChildOptionGroups	Boolean	Indicates whether there are child option groups.
HasOptionComponents	Boolean	Indicates if the option group has product components.
Label	String	The option group label.
MaxOptions	Integer	Maximum number of options that can be selected from the group.
MinOptions	Integer	Minimum number of options that must be selected from the group.
Name	String	The option group name.
OptionComponents	List<Apttus_CPQApi.CPQ.ProductOptionComponentDO>	List of product component data objects.
ParentOptiongroupId	ID	Id of the parent option group.
ProductOptionGroupId	ID	Id of the product option group.

Data Object - Apttus_CPQApi.CPQ.ProductOptionComponentDO		
Field	**Type**	**Description**
ComponentId	ID	Id of the component.
ComponentProductId	ID	Id of the component product.

Data Object - Apttus_CPQApi.CPQ.ProductOptionComponentDO		
Field	**Type**	**Description**
Description	String	The product description.
HasPrices	Boolean	Indicates if there are list prices for the product.
ImageUrl	String	The location of the image, if there is one, associated with the product.
IsDefault	Boolean	Indicates if the option is included by default for the bundle.
IsRequired	Boolean	Indicates if the option is mandatory for the bundle.
Name	String	The product name.
Prices	List<Apttus_CPQApi.CPQ.PriceDO>	List of price data objects.
ProductCode	String	The product code.

Data Object - Apttus_CPQApi.CPQ.PriceDO		
Field	**Type**	**Description**
ChargeType	String	The charge type.
PriceItem	Apttus_Config2__PriceListItem__c object	The price list items for a price list.
Value	Decimal	The list price.

Code Sample

Using the sample below you enable the user to search and select products by a price list. You can then pass the IDs of the price list and product as arguments to the API. If in the result object HasOptionGroups returns true, fetch and display the option group components. If the option group components HasComponents=true display the option components such as quantity. For example if the user searches a product and clicks Search, invoke the API to fetch the option groups for that bundle.

```
public void getOptionGroupsForPriceListProduct()
{
    string productID='';
    //Fetch product ID of selected products
    for(ProductWrapperClass selProdWrap: lstProductwrap)
    {
        if(selProdWrap.selected)
        {
            productID=selProdWrap.ProductId;
            break;
        }
    }

    //If no product is selected, show an appropriate error message
    if(productID=='')
    {
        ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Please select at least one
Product.'));

        lstOptionGroup = new
 List<Apttus_CPQApi.CPQ.ProductOptionGroupDO>();
        lstProductOptionWrapper = new List<ProductOptionWrapperClass>();
    }
    //If product is selected, pass pricelistID and productID as parameters
to the API
    else
    {
        Apttus_CPQApi.CPQ.ProductOptionGroupSearchResultDO result =
Apttus_CPQApi.CPQWebService.getOptionGroupsForPriceListProduct(priceListId
, productID);
        lstOptionGroup = new
 List<Apttus_CPQApi.CPQ.ProductOptionGroupDO>();
        lstProductOptionWrapper = new List<ProductOptionWrapperClass>();
```

```
28
29          //If the product has option groups, i,e the flag HasOptionGroups
    returns true run the loop below fetching the components.
30          if(result.HasOptionGroups)
31          {
32              //For an option group fetch and display the following
    components
33              For(Apttus_CPQApi.CPQ.ProductOptionGroupDO optionResult :
    result.OptionGroups)
34              {
35                  ProductOptionWrapperClass objProductOptionWrapperClass=new
     ProductOptionWrapperClass();
36
    objProductOptionWrapperClass.ProductOptionGroupId=optionResult.ProductOpti
    onGroupId;
37                  objProductOptionWrapperClass.Name=optionResult.Name;
38                  objProductOptionWrapperClass.Label=optionResult.Label;
39
    objProductOptionWrapperClass.ParentOptiongroupId=optionResult.ParentOption
    groupId;
40
    objProductOptionWrapperClass.MinOptions=optionResult.MinOptions;
41
    objProductOptionWrapperClass.MaxOptions=optionResult.MaxOptions;
42
    objProductOptionWrapperClass.HasChildOptionGroups=optionResult.HasChildOpt
    ionGroups;
43
    objProductOptionWrapperClass.ChildOptionGroups=optionResult.ChildOptionGro
    ups;
44                  List<ProductQuantityOptionWrapperClass>
    listOptionComponent= new List<ProductQuantityOptionWrapperClass>();
45
46                  //For an option group with components such as min, max and
    quantity fetch and display the editable components to the user
47                  for(Apttus_CPQApi.CPQ.ProductOptionComponentDO
    optionComponent : optionResult.OptionComponents)
48                  {
49                      ProductQuantityOptionWrapperClass
    objProductQuantityOptionWrapperClass = new
     ProductQuantityOptionWrapperClass ();
50                      objProductQuantityOptionWrapperClass.OptionComponent =
    optionComponent;
51                      objProductQuantityOptionWrapperClass.Quantity = 1;
52
```

```
53                      //Fetch and display the list price of the option
components
54                      List<Apttus_CPQApi.CPQ.PriceDO> priceList = new
 List<Apttus_CPQApi.CPQ.PriceDO>();
55                      for(Apttus_CPQApi.CPQ.PriceDO price :
optionComponent.Prices)
56                      {
57                          priceList.Add(price);
58                      }
59                      objProductQuantityOptionWrapperClass.prices =
priceList ;
60
listOptionComponent.Add(objProductQuantityOptionWrapperClass);
61                  }
62                  objProductOptionWrapperClass.OptionComponents =
listOptionComponent;
63                  lstOptionGroup.Add(optionResult);
64                  lstProductOptionWrapper.Add(objProductOptionWrapperClass);
65              }
66
67              ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Option Count: ' +
lstProductOptionWrapper.size()));
68          }
69      }
70  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
```

```
         <cpq:sessionId>00D4P000000z7dE!
AQQAQLOWqt9rvkoE67lP.NXIamuJl0mjoOR6a_l0vkPoATAH.gKFFRWManncKXN0KGY92akqlxsMOzeOlRfEr
g4V2LtI03XQ</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:getOptionGroupsForPriceListProduct>
         <cpq:priceListId>a1a4P00000D3AVG</cpq:priceListId>
         <cpq:productId>01t4P000008OfJ8</cpq:productId>
      </cpq:getOptionGroupsForPriceListProduct>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:ProductOptionGroupSearchResultDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getOptionGroupsForPriceListProductResponse>
            <result>
                <ProductOptionGroupSearchResultDO:HasOptionGroups>true</
ProductOptionGroupSearchResultDO:HasOptionGroups>
                <ProductOptionGroupSearchResultDO:OptionGroups>
                    <ProductOptionGroupSearchResultDO:HasChildOptionGroups>false</
ProductOptionGroupSearchResultDO:HasChildOptionGroups>
                    <ProductOptionGroupSearchResultDO:HasOptionComponents>true</
ProductOptionGroupSearchResultDO:HasOptionComponents>
                    <ProductOptionGroupSearchResultDO:Label>Enclosure & Rack</
ProductOptionGroupSearchResultDO:Label>
                    <ProductOptionGroupSearchResultDO:MaxOptions>1</
ProductOptionGroupSearchResultDO:MaxOptions>
                    <ProductOptionGroupSearchResultDO:MinOptions>1</
ProductOptionGroupSearchResultDO:MinOptions>
                    <ProductOptionGroupSearchResultDO:Name>Enclosure & Rack</
ProductOptionGroupSearchResultDO:Name>
                    <ProductOptionGroupSearchResultDO:OptionComponents>
                        <ProductOptionGroupSearchResultDO:ComponentId>a1z4P000009c5Ra
QAI</ProductOptionGroupSearchResultDO:ComponentId>
```

```
                        <ProductOptionGroupSearchResultDO:ComponentProductId>01t4P000
008OfLWQA0</ProductOptionGroupSearchResultDO:ComponentProductId>
                        <ProductOptionGroupSearchResultDO:Description xsi:nil="true"/
>
                        <ProductOptionGroupSearchResultDO:HasPrices>true</
ProductOptionGroupSearchResultDO:HasPrices>
                        <ProductOptionGroupSearchResultDO:ImageUrl xsi:nil="true"/>
                        <ProductOptionGroupSearchResultDO:IsDefault>true</
ProductOptionGroupSearchResultDO:IsDefault>
                        <ProductOptionGroupSearchResultDO:IsRequired>false</
ProductOptionGroupSearchResultDO:IsRequired>
                        <ProductOptionGroupSearchResultDO:Name>c3000 Server
Enclosure</ProductOptionGroupSearchResultDO:Name>
                        <ProductOptionGroupSearchResultDO:Prices>
                            <ProductOptionGroupSearchResultDO:ChargeType>Standard
Price</ProductOptionGroupSearchResultDO:ChargeType>
                            <ProductOptionGroupSearchResultDO:PriceItem xsi:type="Apt
tus_Config2__PriceListItem__c">
                                <Id>a1Z4P00000EDIH4UAP</Id>
                                <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                                <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                                <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                                <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
                                <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                                <Apttus_Config2__ListPrice__c>199.00000</
Apttus_Config2__ListPrice__c>
                                <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                                <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                                <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                                <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                                <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                                <Apttus_Config2__ProductId__c>01t4P000008OfLWQA0</
Apttus_Config2__ProductId__c>
```

```
                                <Apttus_Config2__ProductId__r xsi:type="Product2">
                                    <Id>01t4P000008OfLWQA0</Id>
                                    <Family>Hardware</Family>
                                    <Name>c3000 Server Enclosure</Name>
                                    <ProductCode>HW-ER001</ProductCode>
                                </Apttus_Config2__ProductId__r>
                                <Name>PI-0000000435</Name>
                            </ProductOptionGroupSearchResultDO:PriceItem>
                            <ProductOptionGroupSearchResultDO:Value>199.00000</
ProductOptionGroupSearchResultDO:Value>
                        </ProductOptionGroupSearchResultDO:Prices>
                        <ProductOptionGroupSearchResultDO:ProductCode>HW-ER001</
ProductOptionGroupSearchResultDO:ProductCode>
                    </ProductOptionGroupSearchResultDO:OptionComponents>
                    <ProductOptionGroupSearchResultDO:OptionComponents>
                        <ProductOptionGroupSearchResultDO:ComponentId>a1z4P000009c5Rb
QAI</ProductOptionGroupSearchResultDO:ComponentId>
                        <ProductOptionGroupSearchResultDO:ComponentProductId>01t4P000
008OfKTQA0</ProductOptionGroupSearchResultDO:ComponentProductId>
                        <ProductOptionGroupSearchResultDO:Description xsi:nil="true"/
>
                        <ProductOptionGroupSearchResultDO:HasPrices>true</
ProductOptionGroupSearchResultDO:HasPrices>
                        <ProductOptionGroupSearchResultDO:ImageUrl xsi:nil="true"/>
                        <ProductOptionGroupSearchResultDO:IsDefault>false</
ProductOptionGroupSearchResultDO:IsDefault>
                        <ProductOptionGroupSearchResultDO:IsRequired>false</
ProductOptionGroupSearchResultDO:IsRequired>
                        <ProductOptionGroupSearchResultDO:Name>c7000 Server
Enclosure</ProductOptionGroupSearchResultDO:Name>
                        <ProductOptionGroupSearchResultDO:Prices>
                            <ProductOptionGroupSearchResultDO:ChargeType>Standard
Price</ProductOptionGroupSearchResultDO:ChargeType>
                            <ProductOptionGroupSearchResultDO:PriceItem xsi:type="Apt
tus_Config2__PriceListItem__c">
                                <Id>a1Z4P00000EDIFqUAP</Id>
                                <Apttus_Config2__Active__c>true</
Apttus_Config2__Active__c>
                                <Apttus_Config2__AllocateGroupAdjustment__c>true</
Apttus_Config2__AllocateGroupAdjustment__c>
                                <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                                <Apttus_Config2__AllowProration__c>false</
Apttus_Config2__AllowProration__c>
```

```
                                <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                                <Apttus_Config2__ListPrice__c>299.00000</
Apttus_Config2__ListPrice__c>
                                <Apttus_Config2__PriceIncludedInBundle__c>false</
Apttus_Config2__PriceIncludedInBundle__c>
                                <Apttus_Config2__PriceListId__c>a1a4P00000D3AVGQA3</
Apttus_Config2__PriceListId__c>
                                <Apttus_Config2__PriceMethod__c>Per Unit</
Apttus_Config2__PriceMethod__c>
                                <Apttus_Config2__PriceType__c>One Time</
Apttus_Config2__PriceType__c>
                                <Apttus_Config2__PriceUom__c>Each</
Apttus_Config2__PriceUom__c>
                                <Apttus_Config2__ProductId__c>01t4P000008OfKTQA0</
Apttus_Config2__ProductId__c>
                                <Apttus_Config2__ProductId__r xsi:type="Product2">
                                    <Id>01t4P000008OfKTQA0</Id>
                                    <Family>Hardware</Family>
                                    <Name>c7000 Server Enclosure</Name>
                                    <ProductCode>HW-ER002</ProductCode>
                                </Apttus_Config2__ProductId__r>
                                <Name>PI-0000000358</Name>
                            </ProductOptionGroupSearchResultDO:PriceItem>
                            <ProductOptionGroupSearchResultDO:Value>299.00000</
ProductOptionGroupSearchResultDO:Value>
                        </ProductOptionGroupSearchResultDO:Prices>
                        <ProductOptionGroupSearchResultDO:ProductCode>HW-ER002</
ProductOptionGroupSearchResultDO:ProductCode>
                    </ProductOptionGroupSearchResultDO:OptionComponents>
                    <ProductOptionGroupSearchResultDO:ParentOptionGroupId xsi:nil="tr
ue"/>
                    <ProductOptionGroupSearchResultDO:ProductOptionGroupId>a154P00000
5liDzQAI</ProductOptionGroupSearchResultDO:ProductOptionGroupId>
                </ProductOptionGroupSearchResultDO:OptionGroups>
            </result>
        </getOptionGroupsForPriceListProductResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Adding Products to a Cart

This API adds one or more products (with default options) to the cart along with quantity, term, start date, and end date.

API	Signature
addMultiProducts	*webService static Apttus_CPQApi.CPQ.AddMultiProductResponseDO addMultiProducts(Apttus_CPQApi.CPQ.AddMultiProductRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.AddMultiProductRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.AddMultiProductRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.
SelectedProducts	List <Apttus_CPQApi.CPQ.SelectedProductDO>	The list of selected product data objects.

Data Object - Apttus_CPQApi.CPQ.SelectedProductDO		
Field	**Type**	**Description**
AttributeValues	List	List of attributes values.

Data Object - Apttus_CPQApi.CPQ.SelectedProductDO		
Field	**Type**	**Description**
Comments	String	Comments associated with the record.
CopyBundleConfigurationFromSource	Boolean	You can use this to copy the configuration of the bundle from the source
CustomData	Apttus_Config2__LineItem__c	This can be used to include the list of custom fields you have added to the product.
CustomFields	List<String> CustomFields	List of custom fields created for your product.
EndDate	Date	The end date.
ProductId	ID	Id of the product bundle.
Quantity	Decimal	The bundle quantity.
RelatedLineItems	List	List of related line items for bundle
SellingTerm	Decimal	The bundle selling term.
SourceFields	List	List of the fields in the source bundle that you want to copy.
SourceId	ID	ID of the source bundle.
StartDate	Date	The start date. You should ensure you use the correct date format.

Response Data Object - Apttus_CPQApi.CPQ.AddMultiProductResponseDO		
Field	**Type**	**Description**
LineNumbers	List<Decimal>	The list of line numbers added to the cart.

Code Sample

The sample below enables you to add multiple selected products with a specific product ID and its associated quantity, validity and selling term to a specific cart with a specific cartID. For example, if the user has selected software and hardware products, the products are added to the cart, on click of Add to Cart. The user is navigated to the cart page where they can view and change the quantity, selling term and other editable aspects of the product.

```
/**
 * The below method demonstrates how to add multiple products to an existing cart (every quote has a cart)
 * Lets assume the Quote's Cart is blank and the standalone/bundle products are Laptop, Monitor, Wifi Router
 * The input of this method is Quote Number and the Ids of the Laptop bundle product, Monitor and Wifi Router standalone products
 */
public static void addMultipleProducts(String quoteNumber, List<ID> productIds)
{

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Id FROM Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name = :quoteNumber LIMIT 1];

    if(!cart.isEmpty() && productIds != null && !productIds.isEmpty())
    {

        // Assume the quantity and selling term for the product is 1
        Integer quantity = 1;
        Integer sellingTerm = 1;

        // Create the request object
```

```
19          Apttus_CPQApi.CPQ.AddMultiProductRequestDO request = new
 Apttus_CPQApi.CPQ.AddMultiProductRequestDO();
20          request.CartId = cart.get(0).Id;
21
22          // Add the products to the request
23          for(ID productId : productIds)
24          {
25              Apttus_CPQApi.CPQ.SelectedProductDO selectedProduct = new
 Apttus_CPQApi.CPQ.SelectedProductDO();
26              selectedProduct.ProductId = productId;
27              selectedProduct.Quantity = quantity;
28              selectedProduct.SellingTerm = sellingTerm;
29              request.SelectedProducts.add(selectedProduct);
30          }
31
32          // Execute the addMultiProducts routine
33          Apttus_CPQApi.CPQ.AddMultiProductResponseDO response =
Apttus_CPQApi.CPQWebService.addMultiProducts(request);
34
35          System.debug('Line Numbers of added products = ' +
response.LineNumbers);
36      }
37  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote

Request/Response XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
```

```
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQHmIRqgn4R9Oi1yQjWTIVk4UZmsDe_.eK0Z9z6qLij7Tu.L_Yo8dRA_p8OmhKMeRs4uzCZTadIgQ9fEbD
KciXEQYRyaA</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:addMultiProducts>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YtUs</cpq1:CartId>
                <cpq1:SelectedProducts>
                    <cpq1:Comments>Some comment</cpq1:Comments>
                    <cpq1:ProductId>01tZ0000004lUXU</cpq1:ProductId>
                    <cpq1:EndDate>2020-06-30</cpq1:EndDate>
                    <cpq1:Quantity>5</cpq1:Quantity>
                    <cpq1:SellingTerm>1</cpq1:SellingTerm>
                    <cpq1:StartDate>2020-05-28</cpq1:StartDate>
                </cpq1:SelectedProducts>
                <cpq1:SelectedProducts>
                    <cpq1:Comments>some comment</cpq1:Comments>
                    <cpq1:ProductId>01tZ0000004lUXt</cpq1:ProductId>
                    <cpq1:EndDate>2020-06-30</cpq1:EndDate>
                    <cpq1:Quantity>1</cpq1:Quantity>
                    <cpq1:SellingTerm>1</cpq1:SellingTerm>
                    <cpq1:StartDate>2020-05-28</cpq1:StartDate>
                </cpq1:SelectedProducts>
                <cpq1:SelectedProducts>
                    <cpq1:Comments>Some comment</cpq1:Comments>
                    <cpq1:ProductId>01tZ0000004lUYI</cpq1:ProductId>
                    <cpq1:EndDate>2020-06-30</cpq1:EndDate>
                    <cpq1:Quantity>2</cpq1:Quantity>
                    <cpq1:SellingTerm>5</cpq1:SellingTerm>
                    <cpq1:StartDate>2020-05-28</cpq1:StartDate>
                </cpq1:SelectedProducts>
            </cpq:request>
        </cpq:addMultiProducts>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:AddMultiProductResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <addMultiProductsResponse>
            <result>
                <AddMultiProductResponseDO:LineNumbers>1</AddMultiProductResponseDO:L
ineNumbers>
                <AddMultiProductResponseDO:LineNumbers>2</AddMultiProductResponseDO:L
ineNumbers>
                <AddMultiProductResponseDO:LineNumbers>3</AddMultiProductResponseDO:L
ineNumbers>
            </result>
        </addMultiProductsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Adding a Bundle to a Cart

This adds a bundle along with selected products, options, and updated quantity to the cart.

API	Signature
addBundle	*webService static Apttus_CPQApi.CPQ.AddBundleResponseDO addBundle(Apttus_CPQApi.CPQ.AddBundleRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.AddBundleRequestDO	The id of the agreement

Request Data Object - Apttus_CPQApi.CPQ.AddBundleRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.
SelectedBundle	Apttus_CPQApi.CPQ.SelectedBundleDO	The bundle to be added to the cart.

Data Object - Apttus_CPQApi.CPQ.SelectedBundleDO		
Field	**Type**	**Description**
SelectedOptions	List<Apttus_CPQApi.CPQ.SelectedOptionDO>	The list of selected option data objects.
SelectedProduct	Apttus_CPQApi.CPQ.SelectedProductDO	The selected product data object.

Data Object - Apttus_CPQApi.CPQ.SelectedProductDO		
Field	**Type**	**Description**
AttributeValues	List	List of attributes values.
Comments	String	Comments associated with the record.
CopyBundleConfigurationFromSource	Boolean	You can use this to copy the configuration of the bundle from the source

Data Object - Apttus_CPQApi.CPQ.SelectedProductDO		
Field	**Type**	**Description**
CustomData	Apttus_Config2__LineItem__c	This can be used to include the list of custom fields you have added to the product.
CustomFields	List<String> CustomFields	List of custom fields created for your product.
EndDate	Date	The end date.
ProductId	ID	Id of the product bundle.
Quantity	Decimal	The bundle quantity.
RelatedLineItems	List	List of related line items for bundle
SellingTerm	Decimal	The bundle selling term.
SourceFields	List	List of the fields in the source bundle that you want to copy.
SourceId	ID	ID of the source bundle.
StartDate	Date	The start date. You should ensure you use the correct date format.

Data Object - Apttus_CPQApi.CPQ.SelectedOptionDO		
Field	**Type**	**Description**
AttributeValues	List	List of attributes values.
Comments	String	Comments associated with the record.

Data Object - Apttus_CPQApi.CPQ.SelectedOptionDO		
Field	**Type**	**Description**
ComponentId	ID	Id of the component.
ComponentProductId	ID	Id of the component product.
CustomData	Apttus_Config2__LineItem__c	The values to be set for the custom fields in the CustomFields List
CustomFields	List<String>	List of Custom Field's API Name
EndDate	Date	The end date.
Quantity	Decimal	The option quantity.
SellingTerm	Decimal	The option selling term.
SourceId	ID	ID of the source option.
StartDate	Date	The start date. You should ensure you use the correct date format.

Response Data Object - Apttus_CPQApi.CPQ.AddBundleResponseDO		
Field	**Type**	**Description**
LineNumber	Decimal	The bundle line number.

Code Sample

The sample below enables you to add a bundled product with a specific product ID and its associated quantity to a specific cart with a specific cartID. For example, a user selects a bundled product-Laptop+Mouse and clicks Add to Cart, the request to add bundle to the cart is invoked and the bundle with the specific product ID is added to a cart with the cart ID.

```
/**
 * The below method demonstrates how to add a bundle product to an existing cart (every quote has a cart)
 * Lets assume the Quote's Cart is blank and Laptop is a bundle product and its two options are Keyboard and Mouse
 * The input of this method is Quote Number and the Id of the Laptop bundle product
 * Inside the method we will add the Laptop bundle product and also its two options Keyboard and Mouse to the cart
 */
public static void addBundle(String quoteNumber, ID bundleProductId)
{

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Apttus_Config2__PriceListId__c,Id FROM Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name = :quoteNumber LIMIT 1];

    if(!cart.isEmpty() && bundleProductId != null)
    {

        // Assume the quantity and selling term for the bundle product and its options is 1
        Integer quantity = 1;
        Integer sellingTerm = 1;

        // Create the request object
        Apttus_CPQApi.CPQ.AddBundleRequestDO request = new Apttus_CPQApi.CPQ.AddBundleRequestDO();
        request.CartId = cart.get(0).Id;

        // Add the bundle product to the request
        request.SelectedBundle = new Apttus_CPQApi.CPQ.SelectedBundleDO();
        request.SelectedBundle.SelectedProduct = new Apttus_CPQApi.CPQ.SelectedProductDO();
        request.SelectedBundle.SelectedProduct.ProductId = bundleProductId;
        request.SelectedBundle.SelectedProduct.Quantity = quantity;
        request.SelectedBundle.SelectedProduct.SellingTerm = sellingTerm;

```

```
30          // Get all the options of the bundle product
31          Apttus_CPQApi.CPQ.ProductOptionGroupSearchResultDO
productOptionGroupResult =
Apttus_CPQApi.CPQWebService.getOptionGroupsForPriceListProduct(cart.get(0
).Apttus_Config2__PriceListId__c, bundleProductId);
32          if(productOptionGroupResult.HasOptionGroups)
33          {
34              // Add the option products to the request
35              request.SelectedBundle.SelectedOptions = new
 List<Apttus_CPQApi.CPQ.SelectedOptionDO>();
36              for(Apttus_CPQApi.CPQ.ProductOptionGroupDO
productOptionGroup : productOptionGroupResult.OptionGroups)
37              {
38                  if(productOptionGroup.HasOptionComponents)
39                  {
40                      for(Apttus_CPQApi.CPQ.ProductOptionComponentDO
productOptionComponent : productOptionGroup.OptionComponents)
41                      {
42                          Apttus_CPQApi.CPQ.SelectedOptionDO
selectedOptionDO = new Apttus_CPQApi.CPQ.SelectedOptionDO();
43                          selectedOptionDO.ComponentId =
productOptionComponent.ComponentId;
44                          selectedOptionDO.ComponentProductId =
productOptionComponent.ComponentProductId;
45                          selectedOptionDO.Quantity = quantity;
46                          selectedOptionDO.SellingTerm = sellingTerm;
47
request.SelectedBundle.SelectedOptions.add(selectedOptionDO);
48                      }
49                  }
50              }
51          }
52
53          // Execute the addBundle routine
54          Apttus_CPQApi.CPQ.AddBundleResponseDO response =
Apttus_CPQApi.CPQWebService.addBundle(request);
55
56          System.debug('Line Number of added bundle = ' +
response.LineNumber);
57      }
58  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQFRdZcUFQ8LrHGF_qYNN9deP0ee.07JRqgeS3IF8IIlzILBdAOPySGSkif1VBtALP6pryVNOXfhn0faHO
pGsc9GFVLl0</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:addBundle>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YdIH</cpq1:CartId>
                <cpq1:SelectedBundle>
                    <cpq1:SelectedProduct>
                        <cpq1:ProductId>01tZ0000004lUXt</cpq1:ProductId>
                        <cpq1:Quantity>1</cpq1:Quantity>
                        <cpq1:SellingTerm>1</cpq1:SellingTerm>
                        <cpq1:StartDate>2020-05-14</cpq1:StartDate>
                        <cpq1:EndDate>2020-05-15</cpq1:EndDate>
                        <cpq1:Comments>None</cpq1:Comments>
                    </cpq1:SelectedProduct>
                    <cpq1:SelectedOptions>
                        <cpq1:ComponentId>a1YZ0000002jPuS</cpq1:ComponentId>
```

```
                        <cpq1:ComponentProductId>01tZ0000004u2io</cpq1:ComponentProdu
ctId>
                        <cpq1:Quantity>1</cpq1:Quantity>
                        <cpq1:SellingTerm>1</cpq1:SellingTerm>
                        <cpq1:StartDate>2020-05-14</cpq1:StartDate>
                        <cpq1:EndDate>2020-05-15</cpq1:EndDate>
                        <cpq1:Comments>none</cpq1:Comments>
                    </cpq1:SelectedOptions>
                </cpq1:SelectedBundle>
            </cpq:request>
        </cpq:addBundle>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:AddBundleResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
    <soapenv:Body>
        <addBundleResponse>
            <result>
                <AddBundleResponseDO:LineNumber>8</AddBundleResponseDO:LineNumber>
            </result>
        </addBundleResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Adding Custom Bundles

This API adds custom bundles to the cart. You can define fields like **Quantity**, **Selling Term**, **Start Date**, and **End Date** for the bundle. You can add more than one bundle. The sequencing of the products added to the cart is based on the order in which you add the line items into the request DataObject list. If you have more than one option associated with the bundle, the option sequence would happen based on the sequence we add them to the DataObject list.

API	Signature
`addCustomBundle`	*webService static Apttus_CPQApi.CPQ.AddCustomBundleResponseDO addCustomBundle(Apttus_CPQApi.CPQ.AddCustomBundleRequestDO request)*

Request Parameter		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.AddCustomBundleRequestDO	The request data object

Request Data Object - Apttus_CPQApi.CPQ.AddCustomBundleRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.
LineItemColl	Apttus_CPQApi.CPQ.LineItemCollDO	The Line Item data object

Data Object - Apttus_CPQApi.CPQ.LineItemCollDO		
Field	**Type**	**Description**
LineItems	List<Apttus_Config2__LineItem__c>	The list of the Line Items.

Response Data Object - Apttus_CPQApi.CPQ.AddCustomBundleResponseDO		
Field	**Type**	**Description**
LineNumber	Decimal	The bundle line number.

Code Sample

The sample code below enables you to add custom bundles with a list of bundle and standalone line items to an existing cart by running the code in Execute Anonymous window from developer console

.

```
/**
 * The below method demonstrates how to add custom bundle products to an existing cart (every quote has a cart)
 * Lets assume the Quote's Cart is already configured and the standalone/bundle products are Laptop, Monitor, Wifi Router
 * The input of this method is Quote Number and the line items, Monitor and Wifi Router standalone products
 */

   List<Apttus_Config2__LineItem__c> lineItems = new List<Apttus_Config2__LineItem__c>();
        Apttus_CPQApi.CPQ.LineItemCollDO itemCollDO = new Apttus_CPQApi.CPQ.LineItemCollDO();
        Product2 bundleSO = [SELECT Id FROM Product2 WHERE Name = Laptop' LIMIT 1];
        Product2 optionSO = [SELECT Id FROM Product2 WHERE Name = Monitor' LIMIT 1];
        Product2 optionSO1 = [SELECT Id FROM Product2 WHERE Name = ‘Wifi Router’ LIMIT 1];
        Product2 standaloneSO = [SELECT Id FROM Product2 WHERE Name = 'Quoting Standalone 4' LIMIT 1];

        // bundle products
        itemCollDO.LineItems.add(new Apttus_Config2__LineItem__c(Apttus_Config2__Quantity__c = 1,

Apttus_Config2__SellingTerm__c = 1,
```

```
17
    Apttus_Config2__Comments__c = 'Test bundle',
18
    Apttus_Config2__ProductId__c = bundleSO.Id,
19
    Apttus_Config2__HasOptions__c = true,
20
    Apttus_Config2__LineType__c = 'Product/Service',
21
    Apttus_Config2__IsPrimaryLine__c = true,
22
    Apttus_Config2__ChargeType__c = 'License Fee',
23
    Apttus_Config2__PriceType__c = 'One Time',
24
    Apttus_Config2__PriceMethod__c = 'Per Unit',
25
    Apttus_Config2__ListPrice__c = 5000,
26
    Apttus_Config2__IsCustomPricing__c = true,
27
    Apttus_Config2__AllowManualAdjustment__c = true));
28
29
30          // option products
31          itemCollDO.LineItems.add(new
     Apttus_Config2__LineItem__c(Apttus_Config2__Quantity__c = 1,
32
    Apttus_Config2__SellingTerm__c = 1,
33
    Apttus_Config2__Comments__c = 'Test option 1',
34
    Apttus_Config2__ProductId__c = bundleSO.Id,
35
    Apttus_Config2__OptionId__c = optionSO.Id,
36
    Apttus_Config2__LineType__c = 'Option',
37
    Apttus_Config2__IsQuantityModifiable__c = true,
38
    Apttus_Config2__IsPrimaryLine__c = true,
39
    Apttus_Config2__ChargeType__c = 'License Fee',
```

```
40
Apttus_Config2__PriceType__c = 'One Time',
41
Apttus_Config2__PriceMethod__c = 'Per Unit',
42
Apttus_Config2__ListPrice__c = 750,
43
Apttus_Config2__IsCustomPricing__c = true,
44
Apttus_Config2__AllowManualAdjustment__c = true));
45
46          itemCollDO.LineItems.add(new
 Apttus_Config2__LineItem__c(Apttus_Config2__Quantity__c = 1,
47
Apttus_Config2__SellingTerm__c = 1,
48
Apttus_Config2__Comments__c = 'Test option 2',
49
Apttus_Config2__ProductId__c = bundleSO.Id,
50
Apttus_Config2__OptionId__c = optionSO1.Id,
51
Apttus_Config2__LineType__c = 'Option',
52
Apttus_Config2__IsQuantityModifiable__c = true,
53
Apttus_Config2__IsPrimaryLine__c = false,
54
Apttus_Config2__ChargeType__c = 'Installation Fee',
55
Apttus_Config2__PriceType__c = 'One Time',
56
Apttus_Config2__PriceMethod__c = 'Per Unit',
57
Apttus_Config2__ListPrice__c = 10,
58
Apttus_Config2__IsCustomPricing__c = true,
59
Apttus_Config2__AllowManualAdjustment__c = true));
60
61  itemCollDO.LineItems.add(new
 Apttus_Config2__LineItem__c(Apttus_Config2__Quantity__c = 1,
62
Apttus_Config2__SellingTerm__c = 1,
```

```
63
    Apttus_Config2__Comments__c = 'Test bundle',
64
    Apttus_Config2__ProductId__c = standaloneSO.Id,
65
    Apttus_Config2__HasOptions__c = false,
66
    Apttus_Config2__LineType__c = 'Product/Service',
67
    Apttus_Config2__IsPrimaryLine__c = true,
68
    Apttus_Config2__ChargeType__c = 'License Fee',
69
    Apttus_Config2__PriceType__c = 'One Time',
70
    Apttus_Config2__PriceMethod__c = 'Per Unit',
71
    Apttus_Config2__ListPrice__c = 5000,
72
    Apttus_Config2__IsCustomPricing__c = true,
73
    Apttus_Config2__AllowManualAdjustment__c = true));
74          Apttus_CPQApi.CPQ.AddCustomBundleRequestDO request = new
     Apttus_CPQApi.CPQ.AddCustomBundleRequestDO();
75          request.CartId = 'a1I6C00000116mn'; //cart Id
76          request.LineItemColl = itemCollDO;
77          Apttus_CPQApi.CPQWebService.addCustomBundle(request);
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
```

```
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQJnmdThijKiZBe7nFMnVQReN90.MMbcgJD7OLgwd3vGvODaYV9beTHNKgBG0W1ecM4askcVGqf9Qzk2nW
zcvoOm58gGD</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:addCustomBundle>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002Yqya</cpq1:CartId>
                <cpq1:LineItemColl>
                    <cpq1:LineItems>
                        <cpq:Apttus_Config2__Quantity__c>1</cpq:Apttus_Config2__Quant
ity__c>
                        <cpq:Apttus_Config2__SellingTerm__c>1</cpq:Apttus_Config2__Se
llingTerm__c>
                        <cpq:Apttus_Config2__Comments__c>None</cpq:Apttus_Config2__Co
mments__c>
                        <cpq:Apttus_Config2__ProductId__c>01te0000005NGpD</cpq:Apttus
_Config2__ProductId__c>
                        <cpq:Apttus_Config2__ClassificationId__c>a0ne0000007G1mR</cpq:
Apttus_Config2__ClassificationId__c>
                        <cpq:Apttus_Config2__HasOptions__c>false</cpq:Apttus_Config2_
_HasOptions__c>
                        <cpq:Apttus_Config2__LineType__c>Product/Service</cpq:Apttus_
Config2__LineType__c>
                        <cpq:Apttus_Config2__IsQuantityModifiable__c>true</cpq:Apttus
_Config2__IsQuantityModifiable__c>
                        <cpq:Apttus_Config2__IsPrimaryLine__c >true</cpq:Apttus_Confi
g2__IsPrimaryLine__c >
                        <cpq:Apttus_Config2__ChargeType__c>Standard Price</cpq:Apttus
_Config2__ChargeType__c>
                        <cpq:Apttus_Config2__PriceType__c >One Time</cpq:Apttus_Confi
g2__PriceType__c >
                        <cpq:Apttus_Config2__PriceMethod__c>Per Unit</cpq:Apttus_Conf
ig2__PriceMethod__c>
                        <cpq:Apttus_Config2__ListPrice__c>100</cpq:Apttus_Config2__Li
stPrice__c>
                        <cpq:Apttus_Config2__IsCustomPricing__c>true</cpq:Apttus_Conf
ig2__IsCustomPricing__c>
                        <cpq:Apttus_Config2__AllowManualAdjustment__c>true</cpq:Apttu
s_Config2__AllowManualAdjustment__c>
```

```
                    </cpq1:LineItems>
                    <cpq1:LineItems>
                        <cpq:Apttus_Config2__Quantity__c>1</cpq:Apttus_Config2__Quant
ity__c>
                        <cpq:Apttus_Config2__SellingTerm__c>1</cpq:Apttus_Config2__Se
llingTerm__c>
                        <cpq:Apttus_Config2__Comments__c>Some comment</cpq:Apttus_Con
fig2__Comments__c>
                        <cpq:Apttus_Config2__ProductId__c>01te0000005NGpX</cpq:Apttus
_Config2__ProductId__c>
                        <cpq:Apttus_Config2__HasOptions__c>false</cpq:Apttus_Config2_
_HasOptions__c>
                        <cpq:Apttus_Config2__LineType__c>Product/Service</cpq:Apttus_
Config2__LineType__c>
                        <cpq:Apttus_Config2__IsQuantityModifiable__c>true</cpq:Apttus
_Config2__IsQuantityModifiable__c>
                        <cpq:Apttus_Config2__IsPrimaryLine__c >true</cpq:Apttus_Confi
g2__IsPrimaryLine__c >
                        <cpq:Apttus_Config2__ChargeType__c>Standard Price</cpq:Apttus
_Config2__ChargeType__c>
                        <cpq:Apttus_Config2__PriceType__c >One Time</cpq:Apttus_Confi
g2__PriceType__c >
                        <cpq:Apttus_Config2__PriceMethod__c>Per Unit</cpq:Apttus_Conf
ig2__PriceMethod__c>
                        <cpq:Apttus_Config2__ListPrice__c>100</cpq:Apttus_Config2__Li
stPrice__c>
                        <cpq:Apttus_Config2__IsCustomPricing__c>true</cpq:Apttus_Conf
ig2__IsCustomPricing__c>
                        <cpq:Apttus_Config2__AllowManualAdjustment__c>false</cpq:Aptt
us_Config2__AllowManualAdjustment__c>
                    </cpq1:LineItems>
                    <cpq1:LineItems>
                        <cpq:Apttus_Config2__Quantity__c>2</cpq:Apttus_Config2__Quant
ity__c>
                        <cpq:Apttus_Config2__SellingTerm__c>1</cpq:Apttus_Config2__Se
llingTerm__c>
                        <cpq:Apttus_Config2__OptionId__c>01te0000005NGpm</cpq:Apttus_
Config2__OptionId__c>
                        <cpq:Apttus_Config2__LineType__c>Option</cpq:Apttus_Config2__
LineType__c>
                        <cpq:Apttus_Config2__IsQuantityModifiable__c>true</cpq:Apttus
_Config2__IsQuantityModifiable__c>
```

```
                        <cpq:Apttus_Config2__IsPrimaryLine__c >false</cpq:Apttus_Conf
ig2__IsPrimaryLine__c >
                        <cpq:Apttus_Config2__ChargeType__c>Standard Price</cpq:Apttus
_Config2__ChargeType__c>
                        <cpq:Apttus_Config2__PriceType__c >One Time</cpq:Apttus_Confi
g2__PriceType__c >
                        <cpq:Apttus_Config2__PriceMethod__c>Per Unit</cpq:Apttus_Conf
ig2__PriceMethod__c>
                        <cpq:Apttus_Config2__ListPrice__c>200</cpq:Apttus_Config2__Li
stPrice__c>
                        <cpq:Apttus_Config2__IsCustomPricing__c>false</cpq:Apttus_Con
fig2__IsCustomPricing__c>
                        <cpq:Apttus_Config2__AllowManualAdjustment__c>false</cpq:Aptt
us_Config2__AllowManualAdjustment__c>
                    </cpq1:LineItems>
                </cpq1:LineItemColl>
            </cpq:request>
        </cpq:addCustomBundle>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:AddCustomBundleResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <addCustomBundleResponse>
            <result>
                <AddCustomBundleResponseDO:LineNumber>6</AddCustomBundleResponseDO:Li
neNumber>
            </result>
        </addCustomBundleResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Adding Options to a Bundle

This adds one or more Options Products to a Bundle product.

API	Signature
addOptions	*webService static Apttus_CPQApi.CPQ.AddOptionsResponseDO addOptions(Id cartId, Integer lineNumber, List selectedOptions)*

Request Parameters

Name	Type	Description
CartID	ID	The cart ID
lineNumber	Integer	Line Number of the Bundle Product
selectedOptions	List<Apttus_CPQApi.CPQ.SelectedOptionDO>	List of Options

Data Object - Apttus_CPQApi.CPQ.SelectedOptionDO

Field	Type	Description
AttributeValues	List	List of attributes values.
Comments	String	Comments associated with the record.
ComponentId	ID	Id of the component.
ComponentProductId	ID	Id of the component product.
CustomData	Apttus_Config2__LineItem__c	The values to be set for the custom fields in the CustomFields List
CustomFields	List<String>	List of Custom Field's API Name
EndDate	Date	The end date.
Quantity	Decimal	The option quantity.

Data Object - Apttus_CPQApi.CPQ.SelectedOptionDO		
Field	**Type**	**Description**
SellingTerm	Decimal	The option selling term.
SourceId	ID	ID of the source option.
StartDate	Date	The start date. You should ensure you use the correct date format.

Code Sample

Using the sample below you can dynamically add options to a bundled product that the user selects in the cart line item. For example, if the user selects a bundled product, Laptop, in the cart line item you can fetch the configured options for that selected product using the *addoptions* API. In the sample below, for a selected product you initially fetch the productID. Using the retrieve option groups API *getOptionGroupsForPriceListProduct*, you can check whether the selected product has option groups associated. If an option group is associated, fetch the option components and create a list comprising the CPQ.SelectedOptionDO. Invoke the *addoptions* API and pass the cartID, productID, and the list CPQ.SelectedOptionDO.

```
public void addOptions()
{
    string productName='';
    Integer lineNumber = 0;
    for(LineItemWrapperClass selProdWrap: lstWrapItems)
    {
        //For a selected bundled product from a cart line item execute the
loop below
        if(selProdWrap.selected)
        {
            productName = selProdWrap.ProductName;
            lineNumber = Integer.valueOf(selProdWrap.LineNumber);
            break;
        }
    }
    //If no product is selected, show an appropriate error message
    if(String.isBlank(productName))
```

```
17      {
18          ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Please select at least one
Product.'));
19      }
20      else
21      {
22          List<Apttus_CPQApi.CPQ.SelectedOptionDO> selectedOptDOList = new
 List<Apttus_CPQApi.CPQ.SelectedOptionDO>();
23
24          //For the selected product fetch the product ID
25          List<Product2> productId = [SELECT Id FROM Product2 WHERE Name
= :productName LIMIT 1];
26
27          //Execute the getoptionGroups API to fetch the options with option
groups
28          Apttus_CPQApi.CPQ.ProductOptionGroupSearchResultDO result =
Apttus_CPQApi.CPQWebService.getOptionGroupsForPriceListProduct(priceListId
, productId[0].Id);
29
30          //If the selected product has option groups execute the loop below
31          if(result.HasOptionGroups)
32          {
33              List<Apttus_CPQApi.CPQ.ProductOptionGroupDO> prodOptGrpDOList
= result.OptionGroups;
34              for(Apttus_CPQApi.CPQ.ProductOptionGroupDO prodOptGrpDO :
prodOptGrpDOList)
35              {
36                  //For an option group if a product has option components
such as quantity execute the loop below
37                  if(prodOptGrpDO.HasOptionComponents)
38                  {
39                      List<Apttus_CPQApi.CPQ.ProductOptionComponentDO>
prodOptCompDOList = new
 List<Apttus_CPQApi.CPQ.ProductOptionComponentDO>();
40                      prodOptCompDOList = prodOptGrpDO.OptionComponents;
41
42                      //Fetch all the option components for a particular
option group.
43                      for(Apttus_CPQApi.CPQ.ProductOptionComponentDO
prodOptCompDO :prodOptCompDOList )
44                      {
```

```
45                          Apttus_CPQApi.CPQ.SelectedOptionDO selectedOptDO =
    new Apttus_CPQApi.CPQ.SelectedOptionDO();
46                          selectedOptDO.ComponentId =
    prodOptCompDO.ComponentId;
47                          selectedOptDO.ComponentProductId =
    prodOptCompDO.ComponentProductId;
48                          selectedOptDO.Quantity = 1;
49                          selectedOptDOList.add(selectedOptDO);
50                      }
51                  }
52              }
53          }
54          //Invoke the addOptions API and pass the cartID, selected product
    line number, and the Option List retrieved from the getoptiongroups API
55          Apttus_CPQApi.CPQ.AddOptionsResponseDO addOptRespDO =
    Apttus_CPQApi.CPQWebService.addOptions(cartId, lineNumber,
    selectedOptDOList);
56
57      }
58  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQHmIRqgn4R9Oi1yQjWTIVk4UZmsDe_.eK0Z9z6qLij7Tu.L_Yo8dRA_p8OmhKMeRs4uzCZTadIgQ9fEbD
KciXEQYRyaA</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
```

```
        <cpq:addOptions>
            <cpq:cartId>a1OZ0000002ZQVZ</cpq:cartId>
            <cpq:lineNumber>1</cpq:lineNumber>
            <cpq:selectedOptions>
                <cpq1:Comments>Some comment</cpq1:Comments>
                <cpq1:ComponentId>a1YZ0000003pIGM</cpq1:ComponentId>
                <cpq1:ComponentProductId>01tZ0000004lUYI</cpq1:ComponentProductId>
                <cpq1:EndDate>2020-06-30</cpq1:EndDate>
                <cpq1:Quantity>5</cpq1:Quantity>
                <cpq1:SellingTerm>1</cpq1:SellingTerm>
                <cpq1:StartDate>2020-05-28</cpq1:StartDate>
            </cpq:selectedOptions>
        </cpq:addOptions>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:AddOptionsResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <addOptionsResponse>
            <result>
                <AddOptionsResponseDO:addedOptionLineItems xsi:type="Apttus_Config2__
LineItem__c">
                    <Id>a19Z0000006sPeBIAU</Id>
                    <Apttus_Config2__AddedBy__c>User</Apttus_Config2__AddedBy__c>
                    <Apttus_Config2__AllowManualAdjustment__c>true</
Apttus_Config2__AllowManualAdjustment__c>
                    <Apttus_Config2__Comments__c>Some comment</
Apttus_Config2__Comments__c>
                    <Apttus_Config2__ConfigStatus__c>NA</
Apttus_Config2__ConfigStatus__c>
                    <Apttus_Config2__ConfigurationId__c>a1OZ0000002ZQVZMA4</
Apttus_Config2__ConfigurationId__c>
                    <Apttus_Config2__ConstraintCheckStatus__c>NA</
Apttus_Config2__ConstraintCheckStatus__c>
```

```
                    <Apttus_Config2__Customizable__c>false</
Apttus_Config2__Customizable__c>
                    <Apttus_Config2__Description__c>Auto_API_OptionProduct1</
Apttus_Config2__Description__c>
                    <Apttus_Config2__EndDate__c>2020-06-30</
Apttus_Config2__EndDate__c>
                    <Apttus_Config2__HasAttributes__c>false</
Apttus_Config2__HasAttributes__c>
                    <Apttus_Config2__HasOptions__c>false</
Apttus_Config2__HasOptions__c>
                    <Apttus_Config2__IsPrimaryLine__c>true</
Apttus_Config2__IsPrimaryLine__c>
                    <Apttus_Config2__IsQuantityModifiable__c>false</
Apttus_Config2__IsQuantityModifiable__c>
                    <Apttus_Config2__ItemSequence__c>2</
Apttus_Config2__ItemSequence__c>
                    <Apttus_Config2__LineNumber__c>1</Apttus_Config2__LineNumber__c>
                    <Apttus_Config2__LineType__c>Option</Apttus_Config2__LineType__c>
                    <Apttus_Config2__OptionId__c>01tZ0000004lUYIIA2</
Apttus_Config2__OptionId__c>
                    <Apttus_Config2__ParentBundleNumber__c>1</
Apttus_Config2__ParentBundleNumber__c>
                    <Apttus_Config2__PriceListId__c>a1DZ0000002mg5nMAA</
Apttus_Config2__PriceListId__c>
                    <Apttus_Config2__PricingStatus__c>Pending</
Apttus_Config2__PricingStatus__c>
                    <Apttus_Config2__PrimaryLineNumber__c>4</
Apttus_Config2__PrimaryLineNumber__c>
                    <Apttus_Config2__ProductId__c>01tZ0000004lUXUIA2</
Apttus_Config2__ProductId__c>
                    <Apttus_Config2__ProductOptionId__c>a1YZ0000003pIGMMA2</
Apttus_Config2__ProductOptionId__c>
                    <Apttus_Config2__ProductVersion__c>1.00</
Apttus_Config2__ProductVersion__c>
                    <Apttus_Config2__Quantity__c>5</Apttus_Config2__Quantity__c>
                    <Apttus_Config2__SellingTerm__c>1</Apttus_Config2__SellingTerm__c>
                    <Apttus_Config2__StartDate__c>2020-05-28</
Apttus_Config2__StartDate__c>
                    <Apttus_Config2__Uom__c>Each</Apttus_Config2__Uom__c>
                </AddOptionsResponseDO:addedOptionLineItems>
            </result>
        </addOptionsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Cloning Bundle Line Items on the Cart

This method allows you to clone the primary bundle line items along with option line items and child line items (if applicable) on the Cart. This API is invoked when you click **Clone** action icon next to a bundle product on the Cart.

API	Signature
cloneBundleLineItems	*webService static Apttus_CPQApi.CPQ.CloneLineItemsResponseDO cloneBundleLineItems(Apttus_CPQApi.CPQ.CloneLineItemsRequestDO request)*

Request Parameter

Name	Type	Description
Request	Apttus_CPQApi.CPQ.CloneLineItemsRequestDO	This is the request data object.

Request Data Object – Apttus_CPQApi.CPQ.CloneLineItemsRequestDO

Name	Type	Description
Cart ID	ID	The Id of the cart.
Primary LineNumber	Integer	The line numbers of the primary line items which have to be cloned using this API. ⓘ You must pass the value *Apttus_Config2__LineNumber_c* field instead of *Apttus_Config2__PrimaryLineNumber_c* to clone bundles.

Response Data Object – Apttus_CPQApi.CPQ.CloneLineItemsResponseDO

Name	Type	Description
OriginalToCloneMap	List<Apttus_CPQApi.CPQ.IntegerMapDO>	The list of primary line item numbers.

Response Data Object - Apttus_CPQApi.CPQ.IntegerMapDO		
Name	**Type**	**Description**
mapItem.Key	Integer	The line item numbers of the source primary line item from which new line items c API.
mapItem.Value	Integer	The line item numbers of the cloned line items.

Code Sample

Using the below sample, you can clone the primary bundle line items after you configure your products, add the attributes, configure your options and arrive on the cart. This API accepts the Cart ID and the primary bundle and option line item numbers. In the form of response, this API provides the original line items in the `mapItem.Key` parameter and the newly cloned line items in the `mapItem.Value` parameter.

For example, once you configure your cart by adding products using `addBundle` or `addMultiProducts` API, you can invoke this API in order to clone the bundle and option line items on the cart at any of the following:

- After the product is added on the Cart.
- Before making changes to pricing of the products.
- After updating the cart pricing using `updatePriceForCart` API.

```
/**
 * The below method demonstrates how to clone bundle product in an existing cart
(every quote has a cart)
 * Lets assume the Quote's Cart has a 3 products,
 * Laptop is a bundle product (line number 1) and Monitor and Wifi Router are
standalone products (line number 2 and 3 respectively)
 * The input of this method is Quote Number and the line number of the Laptop bundle
product
 */
public static void cloneBundleLineItems(String quoteNumber, List<Integer>
primaryLineNumbers)
{

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Id FROM
Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name
= :quoteNumber LIMIT 1];
```

```
    if(!cart.isEmpty() && primaryLineNumbers != null && !
primaryLineNumbers.isEmpty())
    {

        // Create the request object
        Apttus_CPQApi.CPQ.CloneLineItemsRequestDO request = new
 Apttus_CPQApi.CPQ.CloneLineItemsRequestDO();
        request.CartId = cart.get(0).Id;
        request.PrimaryLineNumbers = primaryLineNumbers;

        // Execute the cloneBundleLineItems routine
        Apttus_CPQApi.CPQ.CloneLineItemsResponseDO response =
Apttus_CPQApi.CPQWebService.cloneBundleLineItems(request);

        for(Apttus_CPQApi.CPQ.IntegerMapDO intMapDO : response.OriginalToCloneMap)
        {
            System.debug('Source Bundle Line Number = ' + intMapDO.Key + ', Cloned
Bundle Line Number = ' + intMapDO.Value);
        }
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding Products to a Cart OR Adding a Bundle to a Cart
- Updating Price For A Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQGrz6Z.wmD7Foe_ZhEj44N.Sh34lJRyn0uiVnJa1hQAZgWx3lQ2kM._9AWarlZiTn5kpbVtSuMZEgdXN_
Mlze.kGwLVg</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:cloneBundleLineItems>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YouHMAS</cpq1:CartId>
                <cpq1:PrimaryLineNumbers>3</cpq1:PrimaryLineNumbers>
                <cpq1:PrimaryLineNumbers>4</cpq1:PrimaryLineNumbers>
            </cpq:request>
        </cpq:cloneBundleLineItems>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:CloneLineItemsResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <cloneBundleLineItemsResponse>
            <result>
                <CloneLineItemsResponseDO:OriginalToCloneMap>
                    <CloneLineItemsResponseDO:Key>3</CloneLineItemsResponseDO:Key>
                    <CloneLineItemsResponseDO:Value>5</CloneLineItemsResponseDO:Value
>
                </CloneLineItemsResponseDO:OriginalToCloneMap>
                <CloneLineItemsResponseDO:OriginalToCloneMap>
                    <CloneLineItemsResponseDO:Key>4</CloneLineItemsResponseDO:Key>
                    <CloneLineItemsResponseDO:Value>6</CloneLineItemsResponseDO:Value
>
                </CloneLineItemsResponseDO:OriginalToCloneMap>
            </result>
        </cloneBundleLineItemsResponse>
    </soapenv:Body>
```

```
</soapenv:Envelope>
```

Cloning Line Items on the Cart

This method enables you to clone the primary line items on the Cart. This API is invoked when you click **Clone** action icon next to a standalone product on the Cart.

API	Signature
cloneLineItems	*webService static Apttus_CPQApi.CPQ.CloneLineItemsResponseDO cloneLineItems(Apttus_CPQApi.CPQ.CloneLineItemsRequestDO request)*

Request Parameter		
Name	**Type**	**Description**
Request	Apttus_CPQApi.CPQ.CloneLineItemsRequestDO	This is the request data object.

Request Data Object – Apttus_CPQApi.CPQ.CloneLineItemsRequestDO		
Name	**Type**	**Description**
Cart ID	ID	The Id of the cart.
PrimaryLineNumber	Integer	The line item numbers of the primary line items which have to be cloned using this API.

Response Data Object – Apttus_CPQApi.CPQ.CloneLineItemsResponseDO		
Name	**Type**	**Description**
OriginalToCloneMap	List<Apttus_CPQApi.CPQ.IntegerMapDO>	The list of primary line item numbers.

Response Data Object – Apttus_CPQApi.CPQ.IntegerMapDO		
Name	**Type**	**Description**
mapItem.Key	Integer	The line item numbers of the source primary line item from which new line items are cloned using this API.
mapItem.Value	Integer	The line item numbers of the cloned line items.

Code Sample

Using the following sample, you can clone the primary line items after you configure your products, add the attributes and arrive on the cart. This API accepts the Cart ID and the primary line item numbers, and clones these line items. In the form of the response, this API provides the original line items in the `mapItem.Key` parameter and the newly cloned line items in the `mapItem.Value` parameter.

For example, once you configure your cart using *addMultiProducts* API, you can invoke this API in order to clone the line items on the cart at any of the following:

- After the product is added on the Cart.
- Before making changes to pricing of the products.
- After updating the cart pricing using *updatePriceForCart* API.

.

```
/**
 * The below method demonstrates how to clone standalone product in an existing cart
(every quote has a cart)
 * Lets assume the Quote's Cart has a 3 products,
 * Laptop is a bundle product (line number 1) and Monitor and Wifi Router are
standalone products (line number 2 and 3 respectively)
 * The input of this method is Quote Number and the line numbers of the Monitor and
Wifi Router are standalone products
 */
public static void cloneLineItems(String quoteNumber, List<Integer>
primaryLineNumbers)
{

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Id FROM
Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name
= :quoteNumber LIMIT 1];
```

```
    if(!cart.isEmpty() && primaryLineNumbers != null && !
primaryLineNumbers.isEmpty())
    {

        // Create the request object
        Apttus_CPQApi.CPQ.CloneLineItemsRequestDO request = new
 Apttus_CPQApi.CPQ.CloneLineItemsRequestDO();
        request.CartId = cart.get(0).Id;
        request.PrimaryLineNumbers = primaryLineNumbers;

        // Execute the cloneLineItems routine
        Apttus_CPQApi.CPQ.CloneLineItemsResponseDO response =
Apttus_CPQApi.CPQWebService.cloneLineItems(request);

        for(Apttus_CPQApi.CPQ.IntegerMapDO intMapDO : response.OriginalToCloneMap)
        {
            System.debug('Source Bundle Line Number = ' + intMapDO.Key + ', Cloned
Bundle Line Number = ' + intMapDO.Value);
        }
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding Products to a Cart
- Updating Price For A Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQGrz6Z.wmD7Foe_ZhEj44N.Sh34lJRyn0uiVnJa1hQAZgWx3lQ2kM._9AWarlZiTn5kpbVtSuMZEgdXN_
Mlze.kGwLVg</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:cloneLineItems>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YotOMAS</cpq1:CartId>
                <cpq1:PrimaryLineNumbers>1</cpq1:PrimaryLineNumbers>
                <cpq1:PrimaryLineNumbers>2</cpq1:PrimaryLineNumbers>
                <cpq1:PrimaryLineNumbers>5</cpq1:PrimaryLineNumbers>
            </cpq:request>
        </cpq:cloneLineItems>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:CloneLineItemsResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <cloneLineItemsResponse>
            <result>
                <CloneLineItemsResponseDO:OriginalToCloneMap>
                    <CloneLineItemsResponseDO:Key>1</CloneLineItemsResponseDO:Key>
                    <CloneLineItemsResponseDO:Value>6</CloneLineItemsResponseDO:Value
>
                </CloneLineItemsResponseDO:OriginalToCloneMap>
                <CloneLineItemsResponseDO:OriginalToCloneMap>
                    <CloneLineItemsResponseDO:Key>2</CloneLineItemsResponseDO:Key>
                    <CloneLineItemsResponseDO:Value>7</CloneLineItemsResponseDO:Value
>
                </CloneLineItemsResponseDO:OriginalToCloneMap>
                <CloneLineItemsResponseDO:OriginalToCloneMap>
                    <CloneLineItemsResponseDO:Key>5</CloneLineItemsResponseDO:Key>
```

```
                    <CloneLineItemsResponseDO:Value>8</CloneLineItemsResponseDO:Value
>
                </CloneLineItemsResponseDO:OriginalToCloneMap>
            </result>
        </cloneLineItemsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Cloning Option Line Items on the Cart

This method allows you to clone the option line items and child line items (if applicable) on the Cart. This API is invoked when you click **Clone** action icon next to an option product on the Cart.

API	Signature
cloneOptionLineItems	*webService static Apttus_CPQApi.CPQ.CloneLineItemsResponseDO cloneOptionLineItems(Apttus_CPQApi.CPQ.CloneLineItemsRequestDO request)*

Request Parameter		
Name	**Type**	**Description**
Request	Apttus_CPQApi.CPQ.CloneLineItemsRequestDO	This is the request data object.

Request Data Object – Apttus_CPQApi.CPQ.CloneLineItemsRequestDO		
Name	**Type**	**Description**
Cart ID	ID	The Id of the cart.
PrimaryLineNumber	Integer	The line numbers of the primary line items which have to be cloned using this API.

Response Data Object - Apttus_CPQApi.CPQ.CloneLineItemsResponseDO		
Name	**Type**	**Description**
OriginalToCloneMap	List<Apttus_CPQApi.CPQ.IntegerMapDO>	The list of primary line item numbers.

Response Data Object - Apttus_CPQApi.CPQ.IntegerMapDO		
Name	**Type**	**Description**
mapItem.Key	Integer	The line item numbers of the source primary line item from which new line items are cloned using this API.
mapItem.Value	Integer	The line item numbers of the cloned line items.

Code Sample

Using the below code sample, you can clone the option line items after you configure your bundle products, add the attributes and arrive on the cart. This API accepts the Cart ID and the option line item numbers, and clones these line items. In the form of the response, this API provides the original line items in the `mapItem.Key` parameter and the newly cloned line items in the `mapItem.Value` parameter.

For example, once you configure your cart by adding option products using *addoptions* API, you can invoke this API in order to clone the option line items on the cart at any of the following:

- After the product is added on the Cart.
- Before making changes to pricing of the products.
- After updating the cart pricing using *updatePriceForCart* API.

```
/**
 * The below method demonstrates how to clone option line items in an existing cart
(every quote has a cart)
 * Lets assume the Quote's Cart has a 3 products,
 * Laptop is a bundle product (line number 1) and Monitor and Wifi Router are
standalone products (line number 2 and 3 respectively)
```

```
 * The input of this method is Quote Number and the line number of the Laptop bundle
product
 */
public static void cloneOptionLineItems(String quoteNumber, List<Integer>
primaryLineNumbers)
{

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Id FROM
Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name
= :quoteNumber LIMIT 1];

    if(!cart.isEmpty() && primaryLineNumbers != null && !
primaryLineNumbers.isEmpty())
    {

        // Create the request object
        Apttus_CPQApi.CPQ.CloneLineItemsRequestDO request = new
 Apttus_CPQApi.CPQ.CloneLineItemsRequestDO();
        request.CartId = cart.get(0).Id;
        request.PrimaryLineNumbers = primaryLineNumbers;

        // Execute the cloneOptionLineItems routine
        Apttus_CPQApi.CPQ.CloneLineItemsResponseDO response =
Apttus_CPQApi.CPQWebService.cloneOptionLineItems(request);

        for(Apttus_CPQApi.CPQ.IntegerMapDO intMapDO : response.OriginalToCloneMap)
        {
            System.debug('Original Primary Line Number = ' + intMapDO.Key + ', Cloned
Primary Line Number = ' + intMapDO.Value);
        }
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding a Bundle to a Cart

- Updating Price For A Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQGyIEG86MxZJB2SAYAO.SoCSmGG7G3SzeD6x2lnQ_zQCcBu.SpLUCrnOBk0PhOOtnf_OAAnE0nfpZfkaS
9mA_1G_1kmP</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:cloneOptionLineItems>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YqyaMAC</cpq1:CartId>
                <cpq1:PrimaryLineNumbers>1</cpq1:PrimaryLineNumbers>
                <cpq1:PrimaryLineNumbers>3</cpq1:PrimaryLineNumbers>
            </cpq:request>
        </cpq:cloneOptionLineItems>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:CloneLineItemsResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <cloneOptionLineItemsResponse>
            <result>
                <CloneLineItemsResponseDO:OriginalToCloneMap>
                    <CloneLineItemsResponseDO:Key>1</CloneLineItemsResponseDO:Key>
```

```
                    <CloneLineItemsResponseDO:Value>9</CloneLineItemsResponseDO:Value
>
                </CloneLineItemsResponseDO:OriginalToCloneMap>
                <CloneLineItemsResponseDO:OriginalToCloneMap>
                    <CloneLineItemsResponseDO:Key>3</CloneLineItemsResponseDO:Key>
                    <CloneLineItemsResponseDO:Value>12</CloneLineItemsResponseDO:Valu
e>
                </CloneLineItemsResponseDO:OriginalToCloneMap>
            </result>
        </cloneOptionLineItemsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Comparing Products

This is used to retrieve products, based on product IDs, and compare them side by side. This is typically done from the Product Catalog, to decide between products before adding them to the cart.

Prerequisite: You must set up feature sets and associate them with the products you want to compare. There is no API for this process—you must configure this in your org. For more information, see Configuring Product Comparison.

API	Signature
compareProducts	*webService static List compareProducts(Apttus_CPQApi.CPQ.FeatureInfoRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.FeatureInfoRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.FeatureInfoRequestDO		
Field	**Type**	**Description**
productIds	List<ID>	The list of Ids of the product you want to compare.

Response Data Object - Apttus_CPQApi.CPQ.FeatureInfoResponseDO		
Field	**Type**	**Description**
FeatureInfos	List<Apttus_CPQApi.FeatureSupport.FeatureInfo>	This returns the list of features for each of the products included in the comparison.
ProductId	ID	ID of the products, which have a feature set to be compared.

Data Object - Apttus_CPQApi.FeatureSupport.FeatureInfo		
Field	**Type**	**Description**
ProductFeatureSet	FeatureSet	Associated Product Feature Set
ProductFeatureSOId	Id	Id of associated ProductFeature__c Salesforce Object

Data Object - Apttus_CPQApi. FeatureSupport.FeatureSet		
Field	**Type**	**Description**
Description	String	Description of associated product feature set
Features	List<Apttus_CPQApi. FeatureSupport.ProductFeatureValue>	List of associated feature values
FeatureSetSOId	Id	Id of associated Apttus_Config2__FeatureSet__cSalesforce Object

Data Object - Apttus_CPQApi. FeatureSupport.FeatureSet		
Field	**Type**	**Description**
Name	String	Name of the associated Product Feature Set
Sequence	Decimal	Order of the feature set

Data Object - Apttus_CPQApi.FeatureSupport.ProductFeatureValue		
Field	**Type**	**Description**
FeatureSOId	Id	ID of the associated feature set.
IsIncluded	Boolean	Whether it is included
Name	String	Name of the feature
ProdFeatureValueSOId	Id	Product Feature Value Sold
ProdId	Id	ID of the product.
Sequence	Decimal	The feature display sequence
Value	String	Product Feature Value

Code Sample

The sample below enables you to fetch the IDs of the selected products. Two or more products should be selected to enable comparison. After the user selects two or more products such as a Laptop from different vendors, invoke this API.

```
public List<CPQ.FeatureInfoResponseDO> compareProducts(List<ID>productIds)
{

    //Prompt appropriate error message validation message when the productIds
parameter has less than two products.
```

```
    if(productIds.size()< 2)
    {
        System.debug('Please select atleast two products to compare.');
        return null;
    }

    //If two or more products are selected execute the code below

    else{
        Apttus_CPQApi.CPQ.FeatureInfoRequestDO request = new
 Apttus_CPQApi.CPQ.FeatureInfoRequestDO();
        request.ProductIds = productIds;
        //Invoke the compare products API
        List<Apttus_CPQApi.CPQ.FeatureInfoResponseDO> response =
Apttus_CPQApi.CPQWebService.compareProducts(request);
        return  response;
        }
}
```

Code sample

The sample code below allows you to create a table in a Visualforce page to display the compared products:

```
'compareFeatureSets' in Visualforce code is the getter variable which hold
the
response from compareProducts apex method.


Visualforce code


<!-- Compare Products Panel -->

<apex:outputPanel layout="block" Id="compareProductsPanel" styleClass="gri
d-2-13 clearfix compareProductsPanel">
<table>
    <apex:repeat var="featureSetName" value="{!compareFeatureSets}">
        <tr> <th data-priority="1" colspan="{!colSpan}"><h4>{!
featureSetName}</h4></th></tr>
```

```
14      <apex:repeat var="featureValue" value="{!compareFeatureSets
        [featureSetName][0].productFeatureSet.features}">
15      <apex:outputPanel layout="none"rendered="{!
        hasIncludedProduct[compareFeatureSets[featureSetName]
        [0].productFeatureSet.featureSetSOId]}">
16          <tr><th class="label">{!featureValue.Name}</th>
17              <apex:repeat var="product" value="{!productIds}">
18              <apex:repeat var="prodFeatureInfo" value="{!
        productFeatureInfosResponse[product]}">
19              <apex:outputPanel layout="none" rendered="{!
        prodFeatureInfo.productFeatureSet.Name == featureSetName}">
20              <apex:repeat var="prodFeatureValue" value="{!
        prodFeatureInfo.productFeatureSet.features}">
21              <apex:outputPanel layout="none" rendered="{!
        prodFeatureValue.featureSOId == featureValue.featureSOId &&
        prodFeatureValue.isIncluded}">
22                  <td style="border:1px #CCCCCC solid;">{!
        prodFeatureValue.Value}</td>
23              </apex:outputPanel>
24              <apex:outputPanel layout="none" rendered="{!
        prodFeatureValue.featureSOId == featureValue.featureSOId && NOT
        (prodFeatureValue.isIncluded)}">
25                  <td style="border:1px #CCCCCC solid;"></td>
26              </apex:outputPanel>
27              </apex:repeat>
28              </apex:outputPanel>
29              </apex:repeat>
30              </apex:repeat>
31          </tr>
32      </apex:outputPanel>
33      </apex:repeat>
34      </apex:repeat>
35  </table>
36  </apex:outputPanel>
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

None.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQHmIRqgn4R9Oi1yQjWTIVk4UZmsDe_.eK0Z9z6qLij7Tu.L_Yo8dRA_p8OmhKMeRs4uzCZTadIgQ9fEbD
KciXEQYRyaA</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:compareProducts>
            <cpq:request>
                <cpq1:ProductIds>01ti0000006iMl3</cpq1:ProductIds>
                <cpq1:ProductIds>01ti0000006iMl8</cpq1:ProductIds>
            </cpq:request>
        </cpq:compareProducts>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:FeatureInfoResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ"
    xmlns:FeatureInfo="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
FeatureSupport"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
```

```
        <compareProductsResponse>
            <result>
                <FeatureInfoResponseDO:FeatureInfos>
                    <FeatureInfo:ProductFeatureSet>
                        <FeatureInfo:Description xsi:nil="true"/>
                        <FeatureInfo:Features>
                            <FeatureInfo:FeatureSOId>a1wi0000000jcV3AAI</FeatureInfo:
FeatureSOId>
                            <FeatureInfo:IsIncluded>true</FeatureInfo:IsIncluded>
                            <FeatureInfo:Name>AutoCPQ951Feature1</FeatureInfo:Name>
                            <FeatureInfo:ProdFeatureValueSOId>a1zi0000000WsYeAAK</
FeatureInfo:ProdFeatureValueSOId>
                            <FeatureInfo:ProdId>01ti0000006iMl3AAE</FeatureInfo:ProdI
d>
                            <FeatureInfo:Sequence>1</FeatureInfo:Sequence>
                            <FeatureInfo:Value>No</FeatureInfo:Value>
                        </FeatureInfo:Features>
                        <FeatureInfo:Features>
                            <FeatureInfo:FeatureSOId>a1wi0000000jcV4AAI</FeatureInfo:
FeatureSOId>
                            <FeatureInfo:IsIncluded>true</FeatureInfo:IsIncluded>
                            <FeatureInfo:Name>AutoCPQ951Feature1</FeatureInfo:Name>
                            <FeatureInfo:ProdFeatureValueSOId>a1zi0000000WsYfAAK</
FeatureInfo:ProdFeatureValueSOId>
                            <FeatureInfo:ProdId>01ti0000006iMl3AAE</FeatureInfo:ProdI
d>
                            <FeatureInfo:Sequence>2</FeatureInfo:Sequence>
                            <FeatureInfo:Value>Yes</FeatureInfo:Value>
                        </FeatureInfo:Features>
                        <FeatureInfo:FeatureSetSOId>a1vi0000000psEiAAI</FeatureInfo:F
eatureSetSOId>
                        <FeatureInfo:Name>AutoCPQ951FeatureSet</FeatureInfo:Name>
                        <FeatureInfo:Sequence>1</FeatureInfo:Sequence>
                    </FeatureInfo:ProductFeatureSet>
                    <FeatureInfo:ProductFeatureSOId xsi:nil="true"/>
                </FeatureInfoResponseDO:FeatureInfos>
                <FeatureInfoResponseDO:ProductId>01ti0000006iMl3AAE</
FeatureInfoResponseDO:ProductId>
            </result>
            <result>
                <FeatureInfoResponseDO:FeatureInfos>
                    <FeatureInfo:ProductFeatureSet>
                        <FeatureInfo:Description xsi:nil="true"/>
```

```
                        <FeatureInfo:Features>
                            <FeatureInfo:FeatureSOId>a1wi0000000jcV3AAI</FeatureInfo:
FeatureSOId>
                            <FeatureInfo:IsIncluded>true</FeatureInfo:IsIncluded>
                            <FeatureInfo:Name>AutoCPQ951Feature1</FeatureInfo:Name>
                            <FeatureInfo:ProdFeatureValueSOId>a1zi0000000WsYjAAK</
FeatureInfo:ProdFeatureValueSOId>
                            <FeatureInfo:ProdId>01ti0000006iMl8AAE</FeatureInfo:ProdI
d>
                            <FeatureInfo:Sequence>1</FeatureInfo:Sequence>
                            <FeatureInfo:Value>Yes</FeatureInfo:Value>
                        </FeatureInfo:Features>
                        <FeatureInfo:Features>
                            <FeatureInfo:FeatureSOId>a1wi0000000jcV4AAI</FeatureInfo:
FeatureSOId>
                            <FeatureInfo:IsIncluded>true</FeatureInfo:IsIncluded>
                            <FeatureInfo:Name>AutoCPQ951Feature1</FeatureInfo:Name>
                            <FeatureInfo:ProdFeatureValueSOId>a1zi0000000WsYkAAK</
FeatureInfo:ProdFeatureValueSOId>
                            <FeatureInfo:ProdId>01ti0000006iMl8AAE</FeatureInfo:ProdI
d>
                            <FeatureInfo:Sequence>2</FeatureInfo:Sequence>
                            <FeatureInfo:Value>No</FeatureInfo:Value>
                        </FeatureInfo:Features>
                        <FeatureInfo:FeatureSetSOId>a1vi0000000psEiAAI</FeatureInfo:F
eatureSetSOId>
                        <FeatureInfo:Name>AutoCPQ951FeatureSet</FeatureInfo:Name>
                        <FeatureInfo:Sequence>1</FeatureInfo:Sequence>
                    </FeatureInfo:ProductFeatureSet>
                    <FeatureInfo:ProductFeatureSOId xsi:nil="true"/>
                </FeatureInfoResponseDO:FeatureInfos>
                <FeatureInfoResponseDO:ProductId>01ti0000006iMl8AAE</
FeatureInfoResponseDO:ProductId>
            </result>
        </compareProductsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Associating Constraint Rules to a Cart

This associates constraints rules to a cart.

API	Signature
associateConstraintRules	*webService static void associateConstraintRules(Id cartId, List existingPrimaryNumbers)*

Parameters		
Name	**Type**	**Description**
cartId	ID	The id of the cart.
existingPrimaryNumbers	List<Integer>	This is a collection of primary numbers for which rules are already attached. When existing primary numbers are supplied, rules are associated with the new lines; whereas when null or empty collection is supplied, rules are attached to all the line items.

Code Sample

The sample below enables you to associate a constraint rule to a product added as a line item in a cart. The sample below fetches the line numbers of the products in the cart and associates the constraint rule to the new line numbers of the cart. If the list object for primary line items is null, then the constraint rules are associated with all the products in the line items of the cart. For example if the delivery and packaging charge constraint rule is already applied to the existing products in the cart, you will provide a list of *collection of primary line items comprising selected products* to which the rules are already applied, thus enabling the delivery and packaging constraint rule to be associated with the new line items added to the cart.

```
1  public void associateConstraintRules()
2  {
3     List<Apttus_Config2__LineItem__c> liSOList = [select
   Apttus_Config2__ProductID__c,
4     Apttus_Config2__LineNumber__c from Apttus_Config2__LineItem__c where
5     Apttus_Config2__ConfigurationId__c = :cartID];
```

```
6
7       acrList = new List<Apttus_Config2__LineItem__c>();
8
9       List<Integer> primaryLines = new List<Integer>();
10      for(Apttus_Config2__LineItem__c liSO: liSOList)
11      {
12          for(Apttus_Config2__LineItem__c acrSO: acrList)
13          {
14               if(acrSO.Id == liSO.Id)
15               {
16
primaryLines.Add(liSO.Apttus_Config2__LineNumber__c.intValue());
17               }
18          }
19      }
20
21      for(Apttus_Config2__LineItem__c liSO: liSOList)
22      {
23          for(LineItemWrapperClass objLineItemWrapperClass : lstWrapItems)
24          {
25              if(objLineItemWrapperClass.Selected)
26              {
27                  acrList.add(liSO);
28                  break;
29              }
30          }
31      }
32
33      Apttus_CPQApi.CPQWebService.associateConstraintRules(cartId,
primaryLines);
34  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQGL6XYy.QsjqQQ6RTBnh.1ApTbiqkGAdVz9BS70lxobcyXgHHplmGXAe7p_cf6ziWJ8tpQt_4Q4Bi2VtY
eMyzjhaPbf0</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:associateConstraintRules>
         <cpq:cartId>a1l4P00000Bg6eAQAR</cpq:cartId>
         <!--Zero or more repetitions:-->
         <cpq:existingPrimaryNumbers>1</cpq:existingPrimaryNumbers>
      </cpq:associateConstraintRules>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Body>
      <associateConstraintRulesResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

Applying Constraint Rules to a Cart

This processes and applies all the constraint rules.

Best Practice

It is recommended that you use the associateConstraintRules API before using this API. Directly using this API may not trigger the constraint rule.

API	Signature
applyConstraintRules	*webService static void applyConstraintRules(Id cartId, Boolean finalCheck)*

Parameters		
Name	**Type**	**Description**
cartId	ID	The id of the cart.
finalCheck	Boolean	If this is set to true, runs rules that are marked as check on finalization.

Code Sample

The sample below enables you to apply constraint rules to a cart with a specific cart ID. If you set the finalCheck flag as true, the constraint rules are run when you finalize the cart. If the flag is set as false, the rules are run before cart finalization. For example, if you want to add a Shipping Costs constraint rule only after the cart is finalized, set the finalCheck as true and the shipping costs constraint rule is applied once the user finalized the cart. If you want to apply constraint rules before the cart is finalized, set finalCheck as false. For example, if you set the finalCheck flag as false, installation charges are added along with a product selected before the cart is finalized.

```
public void applyConstraintRules()
{
    // For rules that are not marked as Check on Finalization
    Apttus_CPQApi.CPQWebService.applyConstraintRules(cartID, false);

    // For rules that are marked as Check on Finalization
    //Apttus_CPQApi.CPQWebService.applyConstraintRules(cartID, true);
```

```
8 }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Associating Constraint Rules to a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQFudWwmQHKaOg8qt7T4KJ9MmTK0J055OXmfoUk9bUEL_idltBYg5muQuM4Pm0HVjinAgttLfi55uyxVSv
F5yrkoH.rH4</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:applyConstraintRules>
         <cpq:cartId>a1l4P00000Bg7CkQAJ</cpq:cartId>
         <cpq:finalCheck>false</cpq:finalCheck>
      </cpq:applyConstraintRules>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Body>
      <applyConstraintRulesResponse/>
```

```
    </soapenv:Body>
</soapenv:Envelope>
```

Applying Constraint Rules to Deleted Products

This runs the rules that are related to the deleted products.

API	Signature
applyConstraintRulesOnDelete	*webService static void applyConstraintRulesOnDelete(Id cartId, List deletedProductIds)*

Parameters		
Name	**Type**	**Description**
cartId	ID	The id of the cart.
deletedProductsIds	List<ID>	This is a list of product ids of the line item deleted from the cart.

Code Sample

The sample below enables you to delete any constraint rules applied to a product that is deleted by a customer. For example, for all products you have added an inclusion constraint rule which states whenever you add a product, an installation charge is included along with the product. When the user deletes the product from the cart, this API enables you to delete the associated installation constraint rule applied to the product. Using the Product_ID parameter you can specify the product from which the constraint rule is to be disassociated from.

```
1  Apttus_CPQApi.CPQWebService.applyConstraintRulesOnDelete(CART_ID, new
    String[] {PRODUCT_IDS});
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQAHIR1W9.1as89Auz.CvNYxqyg56gLLWgUtP5VZxidvTsb1DpQZpmyDuqZOiF4VctBp3jhhJIxG9oRQ4A
4F9h98N0inT</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:applyConstraintRulesOnDelete>
         <cpq:cartId>a1l4P00000Bg7D9QAJ</cpq:cartId>
         <cpq:deletedProductIds>01t4P000008OfLL</cpq:deletedProductIds>
      </cpq:applyConstraintRulesOnDelete>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

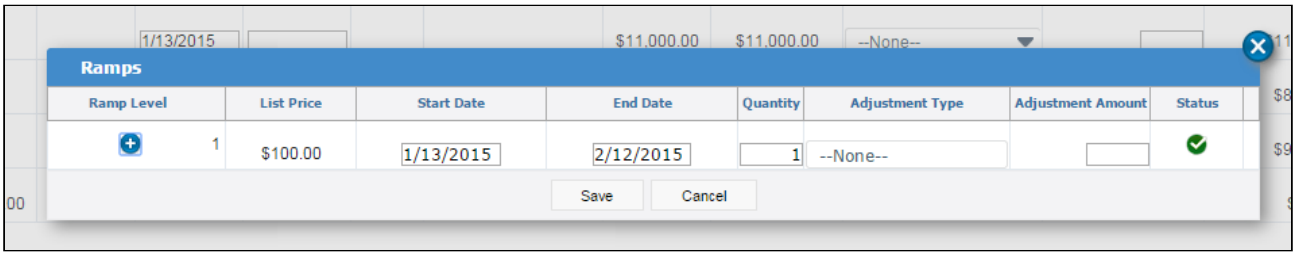
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Body>
      <applyConstraintRulesOnDeleteResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

Adding Price Ramps to a Cart (CPQ Web Service)

After you add a line item to a cart, this API enables you to add primary and secondary ramp line items for the line item. Once the ramp line items are created, you can also update the ramp line item details or delete ramp line item details using standard SOQL queries.

When you use CPQ out of the box, invoke the ramp using the red icon to the left of the primary line item.

Once you click the ramp icon, the ramp dialog appears:



The ramp dialog allows you to add, edit dates and quantity, make adjustments, save the changes, and cancel the changes.

Once the customer adds a ramp to a cart line item, they can do the following:

- Edit the start date, end date, quantity, adjustment type and adjustment amount based on the custom setting.
- The start date of a ramp line item defaults to the end date+1 of the previous line item.
- The end date of a ramp line item defaults to a date such that the difference between the start date and end date is the same as that of the previous line item.
- The user can add more ramp line items after or in between the ramp line items.
- The user can remove the new line before saving by clicking on the icon in the right most column.

Use the addMultiProducts API to add products to the cart.

This API adds one or more products (with default options) to the cart along with quantity, term, start date, and end date.

API	Signature
addMultiProducts	*webService static Apttus_CPQApi.CPQ.AddMultiProductResponseDO addMultiProducts(Apttus_CPQApi.CPQ.AddMultiProductRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_CPQApi.CPQ.AddMultiProductRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.AddMultiProductRequestDO

Field	Type	Description
CartId	ID	The Id of the cart.
SelectedProducts	List <Apttus_CPQApi.CPQ.SelectedProductDO>	The list of selected product data objects.

Data Object - Apttus_CPQApi.CPQ.SelectedProductDO

Field	Type	Description
AttributeValues	List	List of attributes values.
Comments	String	Comments associated with the record.
CopyBundleConfigurationFromSource	Boolean	You can use this to copy the configuration of the bundle from the source
CustomData	Apttus_Config2__LineItem__c	This can be used to include the list of custom fields you have added to the product.

Data Object - Apttus_CPQApi.CPQ.SelectedProductDO		
Field	**Type**	**Description**
CustomFields	List<String> CustomFields	List of custom fields created for your product.
EndDate	Date	The end date.
ProductId	ID	Id of the product bundle.
Quantity	Decimal	The bundle quantity.
RelatedLineItems	List	List of related line items for bundle
SellingTerm	Decimal	The bundle selling term.
SourceFields	List	List of the fields in the source bundle that you want to copy.
SourceId	ID	ID of the source bundle.
StartDate	Date	The start date. You should ensure you use the correct date format.

Response Data Object - Apttus_CPQApi.CPQ.AddMultiProductResponseDO		
Field	**Type**	**Description**
LineNumbers	List<Decimal>	The list of line numbers added to the cart.

Code Sample

The sample below enables you to add ramp line items after you have:

- Added Products to the cart using the AddMultiProducts APIs,
- Updated the Price for the added products using the updatePriceforCart API.
- Selected the products for which you want to add a ramp for.

Using the sample below you fetch the list of selected products for which you want to add a ramp. You also fetch the parameters for each of the selected products. For all the ramps

you create, set PriceGroup as Price Ramp and PricingStatus as Pending. For a primary line item, set IsPrimaryLine__c = true, IsPrimaryRampLine__c = true, and PrimaryLineNumber__c = 1.

```
public void createRampLineItems()
{
    List<String> rampLineItems = new List<String>();

    if(lstWrapItems.size() > 0)
    {
        // Create a list of selected products for which you want to create a ramp for
        for(LineItemWrapperClass objLineItemWrapperClass : lstWrapItems)
        {
            if(objLineItemWrapperClass.Selected)
            {
                rampLineItems.add(objLineItemWrapperClass.Name);
            }
        }

        // Sort Ramp Line Items by name
        rampLineItems.sort();

        Integer rampLineItemIndex = 1;

        for(String rampLineItemName : rampLineItems)
        {
            // Get Line Item parameters for the selected products
            Apttus_Config2__LineItem__c lineItem = [SELECT Apttus_Config2__ItemSequence__c, Apttus_Config2__PricingStatus__c, Apttus_Config2__PriceGroup__c,
            Apttus_Config2__IsPrimaryRampLine__c, Apttus_Config2__IsPrimaryLine__c, Apttus_Config2__LineNumber__c, Apttus_Config2__PrimaryLineNumber__c from Apttus_Config2__LineItem__c WHERE
            Name=:rampLineItemName];

            //Set the parameters for each of the line items
            lineItem.Apttus_Config2__PriceGroup__c = 'Price Ramp';
            lineItem.Apttus_Config2__PricingStatus__c = 'Pending';
            lineItem.Apttus_Config2__LineNumber__c = 1;
```

```
            lineItem.Apttus_Config2__PrimaryLineNumber__c = 1;
            lineItem.Apttus_Config2__ItemSequence__c = rampLineItemIndex;

            //For a primary line item set the following
        `   if(rampLineItemIndex == 1)
            {
                lineItem.Apttus_Config2__IsPrimaryLine__c = true;
                lineItem.Apttus_Config2__IsPrimaryRampLine__c = true;
            }

            //For all secondary line items set the following parameters
            else
            {
                lineItem.Apttus_Config2__IsPrimaryLine__c = false;
                lineItem.Apttus_Config2__IsPrimaryRampLine__c = false;
            }

            // Update Line Items
            update lineItem;

            rampLineItemIndex++;
        }
    }
        else
        {
            ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'No line items available.'));
        }
}
```

Computing the Net Price for a Bundle

This API is used to calculate the price for the individual and summary line items of the cart. It runs the pricing rules, calculate bundle, line item, option level net prices and update the cart line with calculated prices. After the pricing is calculated for the cart line items, the resulting information is stored in the *Line Item* and *Summary Group* objects.

This API will also apply all the relevant rules and calculate the pricing including the following among others:

- Line Item level discount
- Total Price based on the specified quantity
- Applied Tiered Pricing
- Apply Ramp Pricing

- Apply Contractual Pricing
- Apply Related Pricing Rules

API	Signature
computeNetPriceForBundle	*webService static Apttus_CPQApi.CPQ.ComputeNetPriceResponseDO computeNetPriceForBundle(Apttus_CPQApi.CPQ.ComputeNetPriceRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.ComputeNetPriceRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.ComputeNetPriceRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.
LineNumber	Decimal	The bundle line number.

Response Data Object - Apttus_CPQApi.CPQ.ComputeNetPriceResponseDO		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates whether getting the total net price was successful.

Code Sample

The sample below prompts the user with the appropriate messages if a cart does not exist or when no products exist in the cart. If the user clicks calculate net price and products exist in the cart, the API computes the net price for all the line numbers of products in the cart with a specific cartID. For example, if the user wants to check the aggregate price of all the products in the cart, you can create a **Total Net Price** button on your cart page and invoke the computeNetPriceForBundle() API on click of the button. The API will calculate the total price taking into account any discount or pricing rules associated with the line items in the cart.

```
public void computeNetPriceForBundle()
{
    //If a cart does not exist, there is no cartID. Show the following message to the user.
    if(cartId == null)
    {
        ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info, 'Please create cart.'));
    }

    //If no products are added to the cart, the lineNumber of the cart is null. Prompt the use to add products to the cart.
    else if(lineNumber == null)
    {
        ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info, 'Please add a bundle to the cart.'));
    }
    else
    {
        //Selected products are added to the cart. The request object comprises the id of the cart and the line number in the cart
        for(Decimal line: lineNumber)
        {
            Apttus_CPQApi.CPQ.ComputeNetPriceRequestDO request = new Apttus_CPQApi.CPQ.ComputeNetPriceRequestDO();
            request.CartId = cartId;
            request.LineNumber = line;

            Apttus_CPQApi.CPQ.ComputeNetPriceResponseDO priceResponse = Apttus_CPQApi.CPQWebService.computeNetPriceForBundle(request);
```

```
24              ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Success: ' +
priceResponse.IsSuccess));
25          }
26      }
27  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding Products to a Cart
- Adding a Bundle to a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00D4P000000z7dE!
AQQAQAHIR1W9.1as89Auz.CvNYxqyg56gLLWgUtP5VZxidvTsb1DpQZpmyDuqZOiF4VctBp3jhhJIxG9oRQ4A
4F9h98N0inT</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:computeNetPriceForBundle>
            <cpq:request>
                <cpq1:CartId>a1l4P00000Bg7D9QAJ</cpq1:CartId>
                <cpq1:LineNumber>5</cpq1:LineNumber>
            </cpq:request>
        </cpq:computeNetPriceForBundle>
```

```
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
xmlns:ComputeNetPriceResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
   <soapenv:Body>
      <computeNetPriceForBundleResponse>
         <result>
            <ComputeNetPriceResponseDO:IsSuccess>true</ComputeNetPriceResponseDO:IsSu
ccess>
         </result>
      </computeNetPriceForBundleResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Updating Price For A Cart

This API enables you to update the price for items in a given cart. Only line items in pending status are updated.

API	Signature
updatePriceForCart	*webService static Apttus_CPQApi.CPQ.UpdatePriceResponseDO updatePriceForCart(Apttus_CPQApi.CPQ.UpdatePriceRequestDO request)*

Parameters		
Name	**Type**	**Description**
Request	Apttus_CPQApi.CPQ.UpdatePriceRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.UpdatePriceRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart to update the price for.

Response Data Object - Apttus_CPQApi.CPQ.UpdatePriceResponseDO		
Field	**Type**	**Description**
CompletedLineNumbers	List<Integer>	The list of line numbers representing line items that were updated in the current update price invocation.
IsPostPricePending	Boolean	Indicates whether the execution of post-price logic is pending.
IsPrePricePending	Boolean	Indicates whether the execution of pre-price logic is pending.
IsPricePending	Boolean	Indicates whether there are more items to update in future invocations. The caller needs to invoke the update price method in a separate transaction. This should be repeated until there are no more items pending.
IsTotalPricePending	Boolean	Indicates whether the calculation of the total price is pending.
PendingLineNumbers	List<Integer>	The list of line numbers in pending status after the method completes.

Code Sample

The sample below enables you to update the net price in the cart for pending line items. In the sample below, with a single call of updatePriceForCart API, the API computes the net price for a single line item with status pending. *updatePriceForCart* API is invoked using <apex:actionPoller> command until compute net price operation is completed for all the line items in the cart. For example, there are 10 line items in the cart and you invoke *updatePriceForCart* API 10 times at a regular interval of 5 seconds using <apex:actionPoller> command to complete the computing net price operation for all 10 line items.

```
 <apex:actionPoller action="{!doUpdatePrice}" rerender="MessageID"
 interval="5" rendered="{!actionPollerActive}" enabled="{!
actionPollerActive}"/> .
```

In the sample below the action poller is executed as soon as the page is loaded. The *hasPendingitems* flag returns false since no pending items have been added to the cart yet. If a valid cartID has been passed as an argument to the API and *isPricePending* returns true, run the action poller and invoke the *updatePriceforCart* API. The action poller does not run when *hasPendingitems*=false. You can invoke this API when the user adds discounts to the cart and clicks Reprice.

```
public void updatePriceforCart()
{
    hasPendingItems = false;
    actionPollerActive = true;

    if(String.isBlank(cartId))
    {
        ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Please create cart.'));

        objUpdatePriceRequestDO = new
 Apttus_CpqApi.CPQ.UpdatePriceRequestDO();
    }
    else
    {
        // create the update price request
        objUpdatePriceRequestDO = new
 Apttus_CpqApi.CPQ.UpdatePriceRequestDO();

        // add request parameters
```

```
        objUpdatePriceRequestDO.CartId = cartId;

        // update price for the cart
        Apttus_CpqApi.CPQ.UpdatePriceResponseDO result =
Apttus_CpqApi.CPQWebService.updatePriceForCart(objUpdatePriceRequestDO);

        hasPendingItems = result.IsPricePending;

        // Start the action poller. If isPricePending=true returns true,
run the action poller and the updatePrice API is invoked again
        if (hasPendingItems)
        {
            actionPollerActive = true;
            flagUpdatePrice = true;
        }
    }
}

// invoke the following in a new transaction using action poller
public void doUpdatePrice()
{

    // update price for the cart

    if(String.isNotBlank(cartId) && flagUpdatePrice == true)
    {
        // create the update price request
        objUpdatePriceRequestDO = new
 Apttus_CpqApi.CPQ.UpdatePriceRequestDO();

        // add request parameters
        objUpdatePriceRequestDO.CartId = cartId;

        Apttus_CpqApi.CPQ.UpdatePriceResponseDO result =
Apttus_CpqApi.CPQWebService.updatePriceForCart(objUpdatePriceRequestDO);

        hasPendingItems = result.IsPricePending;
        List<Integer> pendingLineNumbers = result.PendingLineNumbers;
        List<Integer> completedLineNumbers = result.CompletedLineNumbers;

        if( hasPendingItems == false && completedLineNumbers.size() != 0 )
        {
            // stop the action poller
```

```
57              actionPollerActive = false;
58          }
59          else if( hasPendingItems == true && pendingLineNumbers.size() > 0
    )
60          {
61              actionPollerActive = true;
62          }
63      }
64  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding Products to a Cart OR Adding a Bundle to a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQLC3rla0VIl_gk7v0A9K1n7dLEcCAANQHwNfXAPFKZEH292fAdeal5Nps8X2klNu98fcp6usnVFmzKARw
E99xq_dE8U6</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:updatePriceForCart>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002ZQVZ</cpq1:CartId>
            </cpq:request>
        </cpq:updatePriceForCart>
```

```
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:UpdatePriceResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
    <soapenv:Body>
        <updatePriceForCartResponse>
            <result>
                <UpdatePriceResponseDO:CompletedLineNumbers>1</UpdatePriceResponseDO:
CompletedLineNumbers>
                <UpdatePriceResponseDO:IsPostPricePending>false</
UpdatePriceResponseDO:IsPostPricePending>
            </result>
        </updatePriceForCartResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Price Breakup for a Cart or Specific Line Item

This method can be used to retrieve the price breakup for a cart or specific line item.

API	Signature
getPriceBreakup	*webService static Apttus_CPQApi.CPQ.GetPriceBreakupResponseDO getPriceBreakup(Apttus_CPQApi.CPQ.GetPriceBreakupRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.GetPriceBreakupRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.GetPriceBreakupRequestDO			
Name	**Type**	**Required**	**Description**
CartId	ID	Yes	The ID of the new cart object. Mandatory parameter.
LineItemId	ID	No	The ID of the line item. This can be null. If it is null the price breakup for all applicable line items in the cart is retrieved. If a value is provided, only the price breakup for that specific line item is retrieved.

Response Data Object - Apttus_CPQApi.CPQ.GetPriceBreakupResponseDO		
Name	**Type**	**Description**
HasPriceBreakups	Boolean	Indicates whether there are price breakups for the cart.
PriceBreakups	List <Apttus_CPQApi.CPQ.PriceBreakupCollDO>	The list of price breakup collection objects

Data Object - Apttus_CPQApi.CPQ.PriceBreakupCollDO		
Name	**Type**	**Description**
BreakupItems	List<Apttus_Config2__PriceBreakup__c>	The list of price breakup sobjects associated with the line item.
LineItemId	ID	The ID of the line item associated with the price break sobjects.

Code Sample

Using the sample below you can enable the end-user to view the price breakup for a selected product. Suppose the user has selected a product Laptop, using the Price Breakup API, you can show the end-user the pricing breakup. For example, if the user has bought 25 laptops and for 20 quantities of the laptop a 5% discount is provided on the net price, and for the other 5 quantities, a discount of 10% is given. Using the *getPriceBreakup* API you can show the user the tiered pricing pattern applied to a product.

```
1  public void getPriceBreakup()
2  {
3      lstPriceBreakup = new List<PriceBreakUpWrapperClass>();
4      //For a selected product in a cart execute this loop.
5      for(LineItemWrapperClass objLineItemWrapperClass: lstWrapItems)
6      {
7          if(objLineItemWrapperClass.selected)
8          {
9              Apttus_CPQApi.CPQ.GetPriceBreakupRequestDO priceBreakUpRequest
   = new Apttus_CPQApi.CPQ.GetPriceBreakupRequestDO();
10             priceBreakUpRequest.CartId = cartId;
11             priceBreakUpRequest.LineItemId = objLineItemWrapperClass.Id;
12
13             //Pass the parameters to the API
14             Apttus_CPQApi.CPQ.GetPriceBreakupResponseDO
   priceBreakUpResponse =
   Apttus_CPQApi.CPQWebService.getPriceBreakup(priceBreakUpRequest);
15
16             //If the response returns true i.e. HasProducts=true execute
   the loop below and fetch the priceBreakup list.
```

```
17              if(priceBreakUpResponse.HasPriceBreakups)
18              {
19                  lstPriceBreakUpCalls = new
 List<Apttus_CPQApi.CPQ.PriceBreakupCollDO>();
20                  lstPriceBreakUpCalls = priceBreakUpResponse.PriceBreakups;
21
22                  for(Apttus_CPQApi.CPQ.PriceBreakupCollDO
objPriceBreakUpCall : lstPriceBreakUpCalls )
23                  {
24                      //priceBreakUpList = new
List<Apttus_Config2__PriceBreakup__c>();
25                      //priceBreakUpList = objPriceBreakUpCall.BreakupItems;
26
27                      String lineItemID = objPriceBreakUpCall.LineItemId;
28
29                      List<Apttus_Config2__PriceBreakup__c>
liSOpriceBreakUpList = [select id, Apttus_Config2__Sequence__c,
Apttus_Config2__BreakupType__c,
30                      Apttus_Config2__TierStartValue__c,
Apttus_Config2__TierEndValue__c, Apttus_Config2__TierQuantity__c,
Apttus_Config2__TierBasePrice__c, Apttus_Config2__TierExtendedPrice__c
from Apttus_Config2__PriceBreakup__c where Apttus_Config2__LineItemId__c
= :lineItemID];
31
32                      for(Apttus_Config2__PriceBreakup__c
lipriceBreakUpList : liSOpriceBreakUpList)
33                      {
34                          PriceBreakUpWrapperClass
objPriceBreakUpWrapperClass = new PriceBreakUpWrapperClass();
35                          objPriceBreakUpWrapperClass.BreakUpID =
lipriceBreakUpList.Id;
36                          objPriceBreakUpWrapperClass.Sequence =
lipriceBreakUpList.Apttus_Config2__Sequence__c;
37                          objPriceBreakUpWrapperClass.BreakupType =
lipriceBreakUpList.Apttus_Config2__BreakupType__c;
38                          objPriceBreakUpWrapperClass.TierStartValue =
lipriceBreakUpList.Apttus_Config2__TierStartValue__c;
39                          objPriceBreakUpWrapperClass.TierEndValue =
lipriceBreakUpList.Apttus_Config2__TierEndValue__c;
40                          objPriceBreakUpWrapperClass.TierQty =
lipriceBreakUpList.Apttus_Config2__TierQuantity__c;
41                          objPriceBreakUpWrapperClass.TierUnitPrice =
lipriceBreakUpList.Apttus_Config2__TierBasePrice__c;
```

```
42                          objPriceBreakUpWrapperClass.TierUnitPrice =
objPriceBreakUpWrapperClass.TierUnitPrice.setScale(2,
System.RoundingMode.HALF_UP);
43                          objPriceBreakUpWrapperClass.TierExtendedPrice =
lipriceBreakUpList.Apttus_Config2__TierExtendedPrice__c;
44                          objPriceBreakUpWrapperClass.TierExtendedPrice =
objPriceBreakUpWrapperClass.TierExtendedPrice.setScale(2,
System.RoundingMode.HALF_UP);
45
46                          lstPriceBreakup.add(objPriceBreakUpWrapperClass);
47                      }
48
49                  }
50              }
51              //If the product selected has no price breakups show an
appropriate message.
52              else
53              {
54                  ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'No Price Breakup Available.')
);
55              }
56          }
57      }
58  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService" xmlns:cpq1="http://
soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQAHIR1W9.1as89Auz.CvNYxqyg56gLLWgUtP5VZxidvTsb1DpQZpmyDuqZOiF4VctBp3jhhJIxG9oRQ4A
4F9h98N0inT</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:getPriceBreakup>
         <cpq:request>
            <cpq1:CartId>a1l4P00000Bg7DTQAZ</cpq1:CartId>
            <cpq1:LineItemId>a1V4P0000053bKq</cpq1:LineItemId>
         </cpq:request>
      </cpq:getPriceBreakup>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
xmlns:GetPriceBreakupResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <getPriceBreakupResponse>
         <result>
            <GetPriceBreakupResponseDO:HasPriceBreakups>true</
GetPriceBreakupResponseDO:HasPriceBreakups>
            <GetPriceBreakupResponseDO:PriceBreakups>
               <GetPriceBreakupResponseDO:BreakupItems xsi:type="Apttus_Config2__Pric
eBreakup__c">
                  <Id>a1W4P00000FIAW0UAP</Id>
```

```
                    <Apttus_Config2__BreakupType__c>Tier Price</
Apttus_Config2__BreakupType__c>
                    <Apttus_Config2__LineItemId__c>a1V4P0000053bKqUAI</
Apttus_Config2__LineItemId__c>
                    <Apttus_Config2__Sequence__c>1</Apttus_Config2__Sequence__c>
                    <Apttus_Config2__TierBasePrice__c>2204.25000</
Apttus_Config2__TierBasePrice__c>
                    <Apttus_Config2__TierEndValue__c>10.00000</
Apttus_Config2__TierEndValue__c>
                    <Apttus_Config2__TierQuantity__c>10.00000</
Apttus_Config2__TierQuantity__c>
                    <Apttus_Config2__TierStartValue__c>0.00000</
Apttus_Config2__TierStartValue__c>
                    <Name>PB-00000084</Name>
                </GetPriceBreakupResponseDO:BreakupItems>
                <GetPriceBreakupResponseDO:BreakupItems xsi:type="Apttus_Config2__Pric
eBreakup__c">
                    <Id>a1W4P00000FIAW1UAP</Id>
                    <Apttus_Config2__BreakupType__c>Tier Price</
Apttus_Config2__BreakupType__c>
                    <Apttus_Config2__LineItemId__c>a1V4P0000053bKqUAI</
Apttus_Config2__LineItemId__c>
                    <Apttus_Config2__Sequence__c>2</Apttus_Config2__Sequence__c>
                    <Apttus_Config2__TierBasePrice__c>1469.50000</
Apttus_Config2__TierBasePrice__c>
                    <Apttus_Config2__TierEndValue__c>25.00000</
Apttus_Config2__TierEndValue__c>
                    <Apttus_Config2__TierQuantity__c>15.00000</
Apttus_Config2__TierQuantity__c>
                    <Apttus_Config2__TierStartValue__c>10.00000</
Apttus_Config2__TierStartValue__c>
                    <Name>PB-00000085</Name>
                </GetPriceBreakupResponseDO:BreakupItems>
                <GetPriceBreakupResponseDO:BreakupItems xsi:type="Apttus_Config2__Pric
eBreakup__c">
                    <Id>a1W4P00000FIAW2UAP</Id>
                    <Apttus_Config2__BreakupType__c>Tier Price</
Apttus_Config2__BreakupType__c>
                    <Apttus_Config2__LineItemId__c>a1V4P0000053bKqUAI</
Apttus_Config2__LineItemId__c>
                    <Apttus_Config2__Sequence__c>3</Apttus_Config2__Sequence__c>
                    <Apttus_Config2__TierBasePrice__c>2939.00000</
Apttus_Config2__TierBasePrice__c>
```

```
                    <Apttus_Config2__TierEndValue__c>40.00000</
Apttus_Config2__TierEndValue__c>
                    <Apttus_Config2__TierQuantity__c>15.00000</
Apttus_Config2__TierQuantity__c>
                    <Apttus_Config2__TierStartValue__c>25.00000</
Apttus_Config2__TierStartValue__c>
                    <Name>PB-00000086</Name>
                </GetPriceBreakupResponseDO:BreakupItems>
                <GetPriceBreakupResponseDO:BreakupItems xsi:type="Apttus_Config2__Pric
eBreakup__c">
                    <Id>a1W4P00000FIAW3UAP</Id>
                    <Apttus_Config2__BreakupType__c>Total</
Apttus_Config2__BreakupType__c>
                    <Apttus_Config2__LineItemId__c>a1V4P0000053bKqUAI</
Apttus_Config2__LineItemId__c>
                    <Apttus_Config2__NetUnitPrice__c>2204.25000</
Apttus_Config2__NetUnitPrice__c>
                    <Apttus_Config2__Sequence__c>4</Apttus_Config2__Sequence__c>
                    <Apttus_Config2__TierBasePrice__c>2204.25000</
Apttus_Config2__TierBasePrice__c>
                    <Apttus_Config2__TierQuantity__c>40.00000</
Apttus_Config2__TierQuantity__c>
                    <Name>PB-00000087</Name>
                </GetPriceBreakupResponseDO:BreakupItems>
                <GetPriceBreakupResponseDO:LineItemId>a1V4P0000053bKqUAI</
GetPriceBreakupResponseDO:LineItemId>
            </GetPriceBreakupResponseDO:PriceBreakups>
         </result>
      </getPriceBreakupResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Removing a Bundle from a Cart

This removes a product bundle and related line items and options from the cart.

API	Signature
removeBundle	*webService static Apttus_CPQApi.CPQ.RemoveBundleResponseDO removeBundle(Apttus_CPQApi.CPQ.RemoveBundleRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.RemoveBundleRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.RemoveBundleRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.
LineNumber	Decimal	The bundle line number.

Response Data Object - Apttus_CPQApi.CPQ.RemoveBundleResponseDO		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates the success of the operation.

Code Sample

The sample below enables you remove a selected bundle from the cart using the line number and a cart id. Invoke the API, when the user selects a bundle from the cart and click Remove or Delete. Based on the line number and the cartID the asset is removed from the cart. Update the new line items using the getLineItems function.

```
1  /**
```

```
 * The below method demonstrates how to remove a bundle product from an existing cart (every quote has a cart)
 * Lets assume the Quote's Cart has a bundle product called Laptop and its two options are Keyboard and Mouse
 * The input of this method is Quote Number and the line number of the Laptop bundle product
 */
public static void removeBundle(String quoteNumber, Integer lineNumber)
{

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Id FROM Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name = :quoteNumber LIMIT 1];

    if(!cart.isEmpty() && lineNumber != null)
    {

        // Create the request object
        Apttus_CPQApi.CPQ.RemoveBundleRequestDO request = new Apttus_CPQApi.CPQ.RemoveBundleRequestDO();
        request.CartId = cart.get(0).Id;
        request.LineNumber = lineNumber;

        // Execute the removeBundle routine
        Apttus_CPQApi.CPQ.RemoveBundleResponseDO response = Apttus_CPQApi.CPQWebService.removeBundle(request);

        System.debug('Remove bundle from cart response status = ' + response.IsSuccess);
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding a Bundle to a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService" xmlns:cpq1="http://
soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQFudWwmQHKaOg8qt7T4KJ9MmTK0J055OXmfoUk9bUEL_idltBYg5muQuM4Pm0HVjinAgttLfi55uyxVSv
F5yrkoH.rH4</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:removeBundle>
         <cpq:request>
            <cpq1:CartId>a1l4P00000Bg7D9QAJ</cpq1:CartId>
            <cpq1:LineNumber>1</cpq1:LineNumber>
         </cpq:request>
      </cpq:removeBundle>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
xmlns:RemoveBundleResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
   <soapenv:Body>
      <removeBundleResponse>
         <result>
            <RemoveBundleResponseDO:IsSuccess>true</RemoveBundleResponseDO:IsSuccess>
         </result>
      </removeBundleResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Removing Multiple Bundles from a Cart

This removes one or more products (with default options) or bundles from the cart along with quantity, term, start date, and end date.

API	Signature
removeMultiBundles	*webService static Apttus_CPQApi.CPQ.RemoveMultiBundlesResponseDO removeMultiBundles(Apttus_CPQApi.CPQ.RemoveMultiBundlesRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.RemoveMultiBundlesRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.RemoveMultiBundlesRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.
LineNumber	List<Decimal>	The List of LineNumbers that need to be removed from the cart.

Response Data Object - Apttus_CPQApi.CPQ.RemoveMultiBundlesResponseDO		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates the success of the operation.

Code Sample

Use the sample below to remove one or more products of the cart. When the user selects multiple products on the cart and clicks Delete or Remove, invoke this API. The multiple products are deleted from the cart. Based on the line number and the cartID the selected assets are removed from the cart. Update the new line items using the getLineItems function.

```
/**
 * The below method demonstrates how to remove multiple products from an existing cart (every quote has a cart)
 * Lets assume the Quote's Cart has a 3 products,
 * Laptop is a bundle product (line number 1) and Monitor and Wifi Router are standalone products (line number 2 and 3 respectively)
 * The input of this method is Quote Number and the line numbers of the Laptop bundle product and Monitor and Wifi Router are standalone products
 */
public static void removeMultiBundles(String quoteNumber, List<Integer> lineNumbers) {

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Id FROM Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name = :quoteNumber LIMIT 1];

    if(!cart.isEmpty() && lineNumbers != null && !lineNumbers.isEmpty()) {

        // Create the request object
        Apttus_CPQApi.CPQ.RemoveMultiBundlesRequestDO request = new Apttus_CPQApi.CPQ.RemoveMultiBundlesRequestDO();
        request.CartId = cart.get(0).Id;
        request.LineNumbers = lineNumbers;

        // Execute the removeMultiBundles routine
        Apttus_CPQApi.CPQ.RemoveMultiBundlesResponseDO response = Apttus_CPQApi.CPQWebService.removeMultiBundles(request);

        System.debug('Remove bundle from cart response status = ' + response.IsSuccess);
    }
```

```
23  }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService" xmlns:cpq1="http://
soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQAHIR1W9.1as89Auz.CvNYxqyg56gLLWgUtP5VZxidvTsb1DpQZpmyDuqZOiF4VctBp3jhhJIxG9oRQ4A
4F9h98N0inT</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:removeMultiBundles>
         <cpq:request>
            <cpq1:CartId>a1l4P00000Bg7D9QAJ</cpq1:CartId>
            <cpq1:LineNumbers>1</cpq1:LineNumbers>
            <cpq1:LineNumbers>3</cpq1:LineNumbers>
         </cpq:request>
      </cpq:removeMultiBundles>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
xmlns:RemoveMultiBundlesResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
   <soapenv:Body>
      <removeMultiBundlesResponse>
```

```
         <result>
            <RemoveMultiBundlesResponseDO:IsSuccess>true</
RemoveMultiBundlesResponseDO:IsSuccess>
         </result>
      </removeMultiBundlesResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Removing Line Items from the Cart

This API enables you to remove specific line items from the Cart. This API is invoked when you click **Remove** action icon next to any product on the Cart.

API	Signature
removeLineItems	*webService static Apttus_CPQApi.CPQ.RemoveLineItemsResponseDO removeLineItems(Apttus_CPQApi.CPQ.RemoveLineItemsRequestDO request)*

Request Parameter

Name	Type	Description
Request	Apttus_CPQApi.CPQ.RemoveLineItemsRequestDO	This is the request data object.

Request Data Object – Apttus_CPQApi.CPQ.RemoveLineItemsRequestDO

Name	Type	Description
Cart ID	ID	The Id of the cart.
PrimaryLineNumbers	Integer	The line item numbers of the primary line items which have to be removed using this API.

Response Data Object - Apttus_CPQApi.CPQ.RemoveLineItemsResponseDO		
Name	**Type**	**Description**
IsSuccess	Boolean	Indicates whether the operation of removing the line items was a success. True indicates success, false indicates that the line items were not removed from the Cart.

Code sample

You can use this API after you realize that the end-user has added some products (bundle, standalone, or option) and you want to remove those products from the Cart at once. Along with the primary line items, this API also removes the child line items under these primary line items (if applicable).

```
/**
 * The below method demonstrates how to remove multiple line items from an existing
cart (every quote has a cart)
 * Lets assume the Quote's Cart has 3 products,
 * a bundle product called Laptop (line number 1) and its two options are Keyboard
and Mouse
 * and Monitor and Wifi Router are standalone products (line number 2 and 3
respectively)
 * The input of this method is Quote Number and the line numbers of the Laptop bundle
product and Monitor standalone product
 */
public static void removeLineItems(String quoteNumber, List<Integer>
primaryLineNumbers)
{

    List<Apttus_Config2__ProductConfiguration__c> cart = [SELECT Id FROM
Apttus_Config2__ProductConfiguration__c WHERE Apttus_QPConfig__Proposald__r.Name
= :quoteNumber LIMIT 1];

    if(!cart.isEmpty() && primaryLineNumbers != null && !
primaryLineNumbers.isEmpty())
    {

        // Create the request object
```

```
        Apttus_CPQApi.CPQ.RemoveLineItemsRequestDO request = new
 Apttus_CPQApi.CPQ.RemoveLineItemsRequestDO();
        request.CartId = cart.get(0).Id;
        request.PrimaryLineNumbers = primaryLineNumbers;

        // Execute the removeLineItems routine
        Apttus_CPQApi.CPQ.RemoveLineItemsResponseDO response =
Apttus_CPQApi.CPQWebService.removeLineItems(request);

        System.debug('Remove line items from cart response status = ' +
response.IsSuccess);
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding Products to a Cart OR Adding a Bundle to a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQGyIEG86MxZJB2SAYAO.SoCSmGG7G3SzeD6x2lnQ_zQCcBu.SpLUCrnOBk0PhOOtnf_OAAnE0nfpZfkaS
9mA_1G_1kmP</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
```

```
        <cpq:removeLineItems>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YqyaMAC</cpq1:CartId>
                <cpq1:PrimaryLineNumbers>2</cpq1:PrimaryLineNumbers>
                <cpq1:PrimaryLineNumbers>7</cpq1:PrimaryLineNumbers>
            </cpq:request>
        </cpq:removeLineItems>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:RemoveLineItemsResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <removeLineItemsResponse>
            <result>
                <RemoveLineItemsResponseDO:IsSuccess>true</RemoveLineItemsResponseDO:
IsSuccess>
            </result>
        </removeLineItemsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Removing Options from a Bundle

This API removes options from the bundle.

API	Signature
removeOptions	*webService static Apttus_CPQApi.CPQ.RemoveOptionsResponseDO removeOptions(Id cartId, Integer lineNumber, List componentIds)*

Request Parameter		
Name	**Type**	**Description**
cartID	ID	The ID of the cart the bundle belongs to.
lineNumber	Integer	The line number of the bundle line items.
componentIds	List<ID>	The list of component IDs of the relevant option line items

Response Data Object - Apttus_CPQApi.*CPQ.RemoveOptionsResponseDO*		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicated whether the removal was successful. The values returned are *true* and *false*.
RemoveIds	List<ID>	The IDs of removed option line items.

Code Sample

The sample code below enables you to remove options from the bundle and get the removal status and list of Ids of the removed options.

```
/**
 * The below method demonstrates how to remove options from bundle
 */
Public void deleteOptions(Id cartDd, Integer lineNumber, List<Id>
componentIds) {
    Apttus_CPQApi.CPQ.RemoveOptionsResponseDO response =
Apttus_CPQApi.CPQ.RemoveOptionsResponseDO();
    Response = Apttus_CPQApi.CPQWebService.removeOptions(cartId,
lineNumber, componentIds);
    System.debug('Has removed? ' + Response. IsSuccess);
```

```
    System.debug('Options removed  ' + Response. RemovedIds);
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Query Product Option Component Id

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQHmIRqgn4R9Oi1yQjWTIVk4UZmsDe_.eK0Z9z6qLij7Tu.L_Yo8dRA_p8OmhKMeRs4uzCZTadIgQ9fEbD
KciXEQYRyaA</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:removeOptions>
            <cpq:cartId>a1OZ0000002ZQVZ</cpq:cartId>
            <cpq:lineNumber>1</cpq:lineNumber>
            <cpq:componentIds>a1YZ0000003pIGMMA2</cpq:componentIds>
        </cpq:removeOptions>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
```

```
    xmlns:RemoveOptionsResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <removeOptionsResponse>
            <result>
                <RemoveOptionsResponseDO:IsSuccess>true</RemoveOptionsResponseDO:IsSu
ccess>
                <RemoveOptionsResponseDO:RemovedIds>a19Z0000006sPeBIAU</
RemoveOptionsResponseDO:RemovedIds>
            </result>
        </removeOptionsResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Constraint Rules Results

This API retrieves the constraints rules applied to a cart.

API	Signature
getConstraintRuleResult	*webService static Apttus_CPQApi.CPQ.ConstraintResultDO getConstraintRuleResult(Id cartId)*

Parameters		
Name	**Type**	**Description**
CartId	ID	The cart ID

Response Data Object - Apttus_CPQApi.CPQ.ConstraintResultDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.

Response Data Object - Apttus_CPQApi.CPQ.ConstraintResultDO		
Field	**Type**	**Description**
ConstraintRuleActions	List<Apttus_CPQApi.CPQ.AppliedActionDO>	The list of constraint rule action applied to the cart.
HasPendingError	Boolean	Indicates whether there are pending rule actions that are error type.
HasPendingWarning	Boolean	Indicates whether there are pending rule actions that are warning type.
NeedMoreProcessing	Boolean	Indicates whether the rule required more processing.

Response Data Object - Apttus_CPQApi.CPQ.AppliedActionDO		
Field	**Type**	**Description**
ActionIntent	String	The action intent of the applied constraint rule.
ActionType	String	The action type of the applied constraint rule.
AffectedPrimaryNumbers	List	The primary line number of the line item the constraint rule is applied on.
AffectedProductIds	List	IDs of the line item the constraint rule is applied on.
AppliedActionId	ID	The ID of the constraint rule action applied on the cart.
AppliedRuleInfoId	ID	The Applied Rule Info ID of the constraint rule action
CartId	ID	The ID of the cart.

Response Data Object - Apttus_CPQApi.CPQ.AppliedActionDO		
Field	**Type**	**Description**
ConstraintRuleActionId	ID	The ID of the constraint rule action.
CriteriaFields	List	The criteria fields in the constraint rule action.
IsAutoExecuted	Boolean	Indicates whether the auto-execution is enabled for the constraint rule.
IsHideMessage	Boolean	Indicates whether the messages are hidden.
IsIgnoredByUser	Boolean	Indicates whether ignoring the constraint rule is allowed.
IsPending	Boolean	Indicates whether any action is pending from the user to execute the constraint rule.
IsShowPrompt	Boolean	Indicates whether prompt is required.
IsTargetOptionProduct	Boolean	Indicates whether the target include or exclude products should be present in the shopping cart as options.
IsTargetPrimaryProduct	Boolean	Indicates whether the include or exclude products should be present in the shopping cart as primary lines.
MatchCountRule	String	
Message	String	The message added to the constraint rule action.
MessageType	String	The type of the message defined in the constraint rule action.
SuggestedProductIds	List	The IDs of the product suggested by rule action definition to be included in the cart.

Response Data Object - Apttus_CPQApi.CPQ.AppliedActionDO		
Field	**Type**	**Description**
TargetBundleNumber	Integer	The parent bundle number of the included or excluded products. For the products that are not option, the number is set as zero which represents the cart as the parent.
TriggeringPrimaryNumbers	List	The primary numbers of the line items that have triggered the rule action.
TriggeringProductIds	List	The list of ids representing the products that caused the rule to trigger

Code Sample

The sample below enables you to fetch the constraint rules applied to a cart with a specific Cart ID. Use this API when you want to view the constraint rules applied to a cart. For example, to generate a report about the constraint rules applied to a cart, you can fetch the constraint rules using this API.

```
public void getConstraintRuleResult()
{
    Integer numErrors = 0;

    Apttus_CPQApi.CPQ.ConstraintResultDO constraintResult =
Apttus_CPQApi.CPQWebService.getConstraintRuleResult(cartID);

    List<Apttus_CPQApi.CPQ.AppliedActionDO> appliedActionDOList =
constraintResult.ConstraintRuleActions;

    for(Apttus_CPQApi.CPQ.AppliedActionDO
appliedActDO:appliedActionDOList)
    {
        ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, appliedActDO.Message));
        if(appliedActDO.MessageType.equals('Error') &&
appliedActDO.IsPending)
        {
            numErrors++;
```

```
        }
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQAHIR1W9.1as89Auz.CvNYxqyg56gLLWgUtP5VZxidvTsb1DpQZpmyDuqZOiF4VctBp3jhhJIxG9oRQ4A
4F9h98N0inT</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:getConstraintRuleResult>
         <cpq:cartId>a1l4P00000Bg7D9QAJ</cpq:cartId>
      </cpq:getConstraintRuleResult>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
xmlns:ConstraintResultDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <getConstraintRuleResultResponse>
         <result>
            <ConstraintResultDO:CartId xsi:nil="true"/>
```

```
            <ConstraintResultDO:ConstraintRuleActions>
               <ConstraintResultDO:ActionIntent>Show Message</ConstraintResultDO:Acti
onIntent>
               <ConstraintResultDO:ActionType>Exclusion</ConstraintResultDO:ActionTyp
e>
               <ConstraintResultDO:AffectedPrimaryNumbers>13</ConstraintResultDO:Affe
ctedPrimaryNumbers>
               <ConstraintResultDO:AffectedProductIds>01t4P000008OfIrQAK</
ConstraintResultDO:AffectedProductIds>
               <ConstraintResultDO:AppliedActionId>a0y4P00000GHs7nQAD</
ConstraintResultDO:AppliedActionId>
               <ConstraintResultDO:AppliedRuleInfoId>a0z4P00000JyUToQAN</
ConstraintResultDO:AppliedRuleInfoId>
               <ConstraintResultDO:CartId>a1l4P00000Bg7D9QAJ</ConstraintResultDO:Cart
Id>
               <ConstraintResultDO:ConstraintRuleActionId>a1H4P000005JdfVUAS</
ConstraintResultDO:ConstraintRuleActionId>
               <ConstraintResultDO:IsAutoExecuted>false</ConstraintResultDO:IsAutoExe
cuted>
               <ConstraintResultDO:IsHideMessage>false</ConstraintResultDO:IsHideMess
age>
               <ConstraintResultDO:IsIgnoredByUser>false</ConstraintResultDO:IsIgnore
dByUser>
               <ConstraintResultDO:IsPending>true</ConstraintResultDO:IsPending>
               <ConstraintResultDO:IsShowPrompt>false</ConstraintResultDO:IsShowPromp
t>
               <ConstraintResultDO:IsTargetOptionProduct>true</ConstraintResultDO:IsT
argetOptionProduct>
               <ConstraintResultDO:IsTargetPrimaryProduct>true</ConstraintResultDO:Is
TargetPrimaryProduct>
               <ConstraintResultDO:MatchCountRule xsi:nil="true"/>
               <ConstraintResultDO:Message>The MSM410 Access Point is not compatible
with the selected blade server</ConstraintResultDO:Message>
               <ConstraintResultDO:MessageType>Error</ConstraintResultDO:MessageType>
               <ConstraintResultDO:SuggestedProductIds>01t4P000008OfIrQAK</
ConstraintResultDO:SuggestedProductIds>
               <ConstraintResultDO:TargetBundleNumber>0</ConstraintResultDO:TargetBun
dleNumber>
               <ConstraintResultDO:TriggeringPrimaryNumbers>14</ConstraintResultDO:Tr
iggeringPrimaryNumbers>
               <ConstraintResultDO:TriggeringProductIds>01t4P000008OfIzQAK</
ConstraintResultDO:TriggeringProductIds>
            </ConstraintResultDO:ConstraintRuleActions>
```

```
            <ConstraintResultDO:HasPendingError>true</ConstraintResultDO:HasPendingEr
ror>
            <ConstraintResultDO:HasPendingWarning>false</ConstraintResultDO:HasPendin
gWarning>
            <ConstraintResultDO:NeedMoreProcessing>false</ConstraintResultDO:NeedMore
Processing>
         </result>
      </getConstraintRuleResultResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Incentives on the Cart

You can use this API to retrieve the incentives that your users want to apply automatically or manually to the products in their shopping cart or a particular bundle.

API	Signature
getIncentives ForCart	*webService static Apttus_CPQApi.CPQ.GetIncentivesForCartResponseDO getIncentivesForCart(Apttus_CPQApi.CPQ.GetIncentivesForCartRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.GetIncentivesForCartRequestDO	The request data object.
Request Data Object - Apttus_CPQApi.CPQ.GetIncentivesForCartRequestDO		
BundleLineItemId	Id	The Id of the bundle on which promotions are to be applied.
CartID	ID	The Id of the cart, containing products on which the promotions are to be applied.

Request Data Object - Apttus_CPQApi.CPQ.GetIncentivesForCartRequestDO		
InclCoupons	Boolean	Indicates whether the coupons are included in the response.

Response Data Object - Apttus_CPQApi.CPQ.GetIncentivesForCartResponseDO		
Field	**Type**	**Description**
AppliedIncentives	List <AppliedIncentives>	A list of incentives applied on the given bundle line item.
IncentiveCoupons	List <IncentiveCoupons>	A list of coupons available on the bundle line item.
IncentiveItems	List <IncentiveItems>	A list of incentives by line item.
Incentives	List<Incentives>	A list of incentives available for the line item.
getAppliedIncentivesByItem()	Map	A list of incentives applied on a line item.
getCouponsByIncentive()	Map	A list of coupons in an incentive.
getIncentivesByItem()	Map	A list of incentives available for a line item.

Data Object - Apttus_Config2__Incentive__c		
Field	**Type**	**Description**
AccountScopeOper	Picklist	Consists of options such as Include and Exclude. These allow you to define a promotion by choosing to include or exclude an account.

Data Object - Apttus_Config2__Incentive__c		
Field	**Type**	**Description**
AccountScope	Look up	Enables you to define promotions for a particular Account. While defining a promotion, you can choose to include or exclude the Account based on which the promotion is applied.
AccountTypeScopeOper	Picklist	Consists of options such as Include and Exclude. These allow you to define a promotion by choosing to include or exclude an account type.
AccountTypeScope	Look up	Enables you to define promotions for a particular Account type. While defining a promotion, you can choose to include or exclude the Account Type based on which the promotion is applied.
Active	Boolean	Can be set to *Yes* or *No*. Set it to *Yes* when you want a promotion to be applied and active for the criteria you specify. The **Active** flag on the promotions will be set using a workflow linked to approvals similar to what you would do for any other custom object approval. For example, you can write a workflow rule to set the active status to true after the promotion is Approved (if approval is setup). Approvals set up would be similar to custom object approvals for any object.

Data Object - Apttus_Config2__Incentive__c		
Field	**Type**	**Description**
Active	Boolean	Can be set to *Yes* or *No*. Set it to *Yes* when you want a promotion to be applied and active for the criteria you specify. The **Active** flag on the promotions will be set using a workflow linked to approvals similar to what you would do for any other custom object approval. For example, you can write a workflow rule to set the active status to true after the promotion is Approved (if approval is setup). Approvals set up would be similar to custom object approvals for any object.
AutoApply	Boolean	Can be set to *Yes* or *No*. Set it to *Yes* when you want a promotion to be automatically applied when a promotion criteria is fulfilled. Set it to *No* when you want the user to manually enter an Incentive Code to apply promotions.
CountryScopeOper	Picklist	Consists of options such as Include and Exclude. These allow you to control if you want to include or exclude a country for a particular promotion.
CountryScope	Look up	Enables you to define promotions for a particular country. While defining a promotion you can choose to include or exclude a country. The country picklist must be defined for the Account object.
EffectiveDate	Date	Start Date from which the promotion is applicable.
ExpirationDate	Date	Date on which the promotion expires. It can be blank. It needs to be greater than the Start Date.

Data Object - Apttus_Config2__Incentive__c		
Field	**Type**	**Description**
IncentiveCode	String	This is the code that is used for applying the incentive. This code is captured on the order or proposal line items when an incentive is applied to the line item. By default, this value is set to Incentive Number, but is editable. The marketing team can also use this for marketing campaigns. When a promotion is not auto-applied, (that is, when the **Auto Apply ?** check box is deselected), the user can use the incentive code to avail a promotional offer.
IncentiveNumber	Auto Number	System generated unique number for the Incentive. This differs from the Salesforce record id. When an active incentive is changed over time, the Incentive number remains same.
Name	String	User-specific name for the Incentive. This promotion name is displayed to the user wherever the Incentive information is displayed. For example, whenever a user looks-up the promotion information applicable to a specific product, the promotion name you provide here is displayed.
PriceListScopeOper	Picklist	Based on the selected Price List, you can control whether to include or exclude that particular price list and its products.
PriceListScope	Look up	Consists of the price list for which you have defined promotions.
ProductFamilyScope	Look up	Consists of a product family for which you have defined promotions.
ProductFamilyScopeOper	Picklist	Based on the selected Product Family, you can control whether to include or exclude that particular product family.

Data Object - Apttus_Config2__Incentive__c		
Field	**Type**	**Description**
ProductScopeOper	Picklist	Based on the selected Product, you can control whether to include or exclude that product.
ProductScope	Look up	Consists of the products for which you have defined promotions.
RegionScopeOper	Picklist	Consists of options such as Include and Exclude. These allow you to control if you want to include or exclude a region for a particular promotion.
RegionScope	Look up	Enables you to define promotions for a particular region. While defining a promotion, you can choose to include or exclude a region. You must define the region picklist for the Account object.
UseType	Picklist	This picklist indicates whether the incentive is of type Promotion or Rebate. In our case, select *Promotion*.
Sequence	Number	Defines the sequence in which incentives are applied, when there are more than one incentives.
Status	Picklist	Specifies the status of the Promotion, which is dependent on the promotion life cycle. After a promotion is approved, you can set the status of the approval as active. Approvals on Promotions: Admins can set up approval processes and other workflows on promotions. Marketing managers can send promotions for approval and approvers can view the promotion, approve or reject the promotions with or without comments, and activate it.

Data Object - Apttus_Config2__Incentive__c		
Field	**Type**	**Description**
StopProcessingMoreIncentives	Boolean	Indicates if CPQ need not process other incentives if there are multiple incentives applicable at the same time.

Code Sample

This code sample enables you to retrieve incentives and coupons available in cart or a line item. You can also retrieve incentives that are applied on the cart or a line item.

```
// get incentive for cart
Apttus_CPQApi.CPQ.GetIncentivesForCartRequestDO request = new
 Apttus_CPQApi.CPQ.GetIncentivesForCartRequestDO();
request.CartId = configSO.Id;
request.BundleLineItemId = bdlLineItemSO.Id;
request.InclCoupons = true;
Apttus_CPQApi.CPQ.GetIncentivesForCartResponseDO result2 =
Apttus_CPQApi.CPQWebService.getIncentivesForCart(request);

// incentives by item
Map<ID, List<Apttus_Config2_Incentive_c>> incentivesByItem =
result2.getIncentivesByItem();
// applied incentives by item

Map<ID, List<Apttus_Config2_AdjustmentLineItem_c>> appliedIncentivesByItem =
result2.getAppliedIncentivesByItem();
// incentive coupons by incentive

Map<ID, List<Apttus_Config2_IncentiveCoupon_c>> couponsByIncentive =
result2.getCouponsByIncentive();
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService" xmlns:cpq1="http://
soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00DZ000000NAEIA!
ASAAQFRdZcUFQ8LrHGF_qYNN9deP0ee.07JRqgeS3IF8IIlzILBdAOPySGSkif1VBtALP6pryVNOXfhn0faHO
pGsc9GFVLl0</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:getIncentivesForCart>
         <cpq:request>
            <cpq1:CartId>a1OZ0000002YdG6</cpq1:CartId>
            <cpq1:InclCoupons>true</cpq1:InclCoupons>
         </cpq:request>
      </cpq:getIncentivesForCart>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:GetIncentivesForCartResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getIncentivesForCartResponse>
            <result>
                <GetIncentivesForCartResponseDO:Incentives xsi:type="Apttus_Config2__
Incentive__c">
                    <Id>a4O0v0000005FtMEAU</Id>
                    <Apttus_Config2__AccountScopeOper__c>in</
Apttus_Config2__AccountScopeOper__c>
```

```
                    <Apttus_Config2__AccountScope__c>All</
Apttus_Config2__AccountScope__c>
                    <Apttus_Config2__AccountTypeScopeOper__c>in</
Apttus_Config2__AccountTypeScopeOper__c>
                    <Apttus_Config2__AccountTypeScope__c>All</
Apttus_Config2__AccountTypeScope__c>
                    <Apttus_Config2__Active__c>true</Apttus_Config2__Active__c>
                    <Apttus_Config2__ApplicationMethod__c>Buy X Get X</
Apttus_Config2__ApplicationMethod__c>
                    <Apttus_Config2__AutoApply__c>false</Apttus_Config2__AutoApply__c>
                    <Apttus_Config2__ContextType__c>Line Item</
Apttus_Config2__ContextType__c>
                    <Apttus_Config2__CountryScopeOper__c>in</
Apttus_Config2__CountryScopeOper__c>
                    <Apttus_Config2__CountryScope__c>All</
Apttus_Config2__CountryScope__c>
                    <Apttus_Config2__EffectiveDate__c>2018-10-15</
Apttus_Config2__EffectiveDate__c>
                    <Apttus_Config2__ExpirationDate__c>2025-10-31</
Apttus_Config2__ExpirationDate__c>
                    <Apttus_Config2__IncentiveCode__c>ICT-002510</
Apttus_Config2__IncentiveCode__c>
                    <Apttus_Config2__IncentiveNumber__c>ICT-002510.0</
Apttus_Config2__IncentiveNumber__c>
                    <Apttus_Config2__PriceListScopeOper__c>in</
Apttus_Config2__PriceListScopeOper__c>
                    <Apttus_Config2__PriceListScope__c>All</
Apttus_Config2__PriceListScope__c>
                    <Apttus_Config2__ProductFamilyScopeOper__c>in</
Apttus_Config2__ProductFamilyScopeOper__c>
                    <Apttus_Config2__ProductFamilyScope__c>All</
Apttus_Config2__ProductFamilyScope__c>
                    <Apttus_Config2__ProductScopeOper__c>in</
Apttus_Config2__ProductScopeOper__c>
                    <Apttus_Config2__ProductScope__c>All</
Apttus_Config2__ProductScope__c>
                    <Apttus_Config2__RegionScopeOper__c>in</
Apttus_Config2__RegionScopeOper__c>
                    <Apttus_Config2__RegionScope__c>All</
Apttus_Config2__RegionScope__c>
                    <Apttus_Config2__Sequence__c>1</Apttus_Config2__Sequence__c>
                    <Apttus_Config2__Status__c>Draft</Apttus_Config2__Status__c>
```

```
                    <Apttus_Config2__StopProcessingMoreIncentives__c>true</
Apttus_Config2__StopProcessingMoreIncentives__c>
                    <Apttus_Config2__SubUseType__c>Promotion</
Apttus_Config2__SubUseType__c>
                    <Apttus_Config2__UseType__c>Promotion</Apttus_Config2__UseType__c>
                    <Name>002I-Auto Single Rate Type Promotion</Name>
                </GetIncentivesForCartResponseDO:Incentives>
                <GetIncentivesForCartResponseDO:Incentives xsi:type="Apttus_Config2__
Incentive__c">
                    <Id>a4O0v0000005FtHEAU</Id>
                    <Apttus_Config2__AccountScopeOper__c>in</
Apttus_Config2__AccountScopeOper__c>
                    <Apttus_Config2__AccountScope__c>All</
Apttus_Config2__AccountScope__c>
                    <Apttus_Config2__AccountTypeScopeOper__c>in</
Apttus_Config2__AccountTypeScopeOper__c>
                    <Apttus_Config2__AccountTypeScope__c>All</
Apttus_Config2__AccountTypeScope__c>
                    <Apttus_Config2__Active__c>true</Apttus_Config2__Active__c>
                    <Apttus_Config2__ApplicationMethod__c>Buy X Get X</
Apttus_Config2__ApplicationMethod__c>
                    <Apttus_Config2__AutoApply__c>false</Apttus_Config2__AutoApply__c>
                    <Apttus_Config2__ContextType__c>Line Item</
Apttus_Config2__ContextType__c>
                    <Apttus_Config2__CountryScopeOper__c>in</
Apttus_Config2__CountryScopeOper__c>
                    <Apttus_Config2__CountryScope__c>All</
Apttus_Config2__CountryScope__c>
                    <Apttus_Config2__EffectiveDate__c>2018-10-14</
Apttus_Config2__EffectiveDate__c>
                    <Apttus_Config2__ExpirationDate__c>2024-10-18</
Apttus_Config2__ExpirationDate__c>
                    <Apttus_Config2__IncentiveCode__c>ICT-002509</
Apttus_Config2__IncentiveCode__c>
                    <Apttus_Config2__IncentiveNumber__c>ICT-002509.0</
Apttus_Config2__IncentiveNumber__c>
                    <Apttus_Config2__PriceListScopeOper__c>in</
Apttus_Config2__PriceListScopeOper__c>
                    <Apttus_Config2__PriceListScope__c>All</
Apttus_Config2__PriceListScope__c>
                    <Apttus_Config2__ProductFamilyScopeOper__c>in</
Apttus_Config2__ProductFamilyScopeOper__c>
                    <Apttus_Config2__ProductFamilyScope__c>All</
Apttus_Config2__ProductFamilyScope__c>
```

```
                    <Apttus_Config2__ProductScopeOper__c>in</
Apttus_Config2__ProductScopeOper__c>
                    <Apttus_Config2__ProductScope__c>All</
Apttus_Config2__ProductScope__c>
                    <Apttus_Config2__RegionScopeOper__c>in</
Apttus_Config2__RegionScopeOper__c>
                    <Apttus_Config2__RegionScope__c>All</
Apttus_Config2__RegionScope__c>
                    <Apttus_Config2__Sequence__c>1</Apttus_Config2__Sequence__c>
                    <Apttus_Config2__Status__c>Draft</Apttus_Config2__Status__c>
                    <Apttus_Config2__StopProcessingMoreIncentives__c>true</
Apttus_Config2__StopProcessingMoreIncentives__c>
                    <Apttus_Config2__SubUseType__c>Promotion</
Apttus_Config2__SubUseType__c>
                    <Apttus_Config2__UseType__c>Promotion</Apttus_Config2__UseType__c>
                    <Name>002H-Auto Single Rate Type Promotion</Name>
                </GetIncentivesForCartResponseDO:Incentives>
            </result>
        </getIncentivesForCartResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Retrieving Asset Line Items

Gets the list of asset line item objects matching the parameters used as the search criteria.

API	Signature
getAssetsForSearchText	*webService static Apttus_CPQApi.CPQ.AssetSearchResultDO getAssetsForSearchText(Id accountId, List locationIds, String searchText)*

You cannot use the parameters *accountId* and *locationIds* simultaneously.

Parameters		
Name	**Type**	**Description**
accountId	ID	The id of the account associated with the asset.
locationIds	List <ID>	The ids of the different locations associated with the assets.
searchText	String	The search text.

Response Data Object - Apttus_CPQApi.CPQ.AssetSearchResultDO		
Name	**Type**	**Description**
AssetItems	List < Apttus_Config2__ AssetLineItem__c >	This is a list of the asset line items. This returns the asset line item records and the values for the fields that belong to each record.
HasAssetItems	Boolean	Indicates whether there are asset line items, that match the parameters used for the search.

Code Sample

When you generate a quote for an existing customer, it is possible that the customer has some assets associated with the account. You can provide a search field on the cart page that enables the customer to search for existing assets and view them. When the user types the search criteria for fetching the assets and clicks Search, invoke the API. This sample should enable the user to search a product by product name, product code, product description, category name, and configuration type. The search for get assets is similar to the product search in the catalog page and installed products page.

```
1  public void searchAssets()
2  {
```

```
    Apttus_CPQApi.CPQ.AssetSearchResultDO assetResults =
Apttus_CPQApi.CPQWebService.getAssetsForSearchText(accountId, null,
assetSearchText);
    List<AssetWrapperClass> lstAssetWrapper = new
 List<AssetWrapperClass>();

    // if assets items that are searched by a keyword exist (i.e
hasassetitems=true)
    // and some value is returned, show the following as a part of the
list
    if (assetResults.HasAssetItems == true)
    {
        List<Apttus_Config2__AssetLineItem__c> aliSOList =
assetResults.AssetItems;
        for (Apttus_Config2__AssetLineItem__c aliSO : aliSOList)
        {
            AssetWrapperClass objAssetWrapperClass = new AssetWrapperClass
();
            objAssetWrapperClass.AssetId = aliSO.ID;
            objAssetWrapperClass.ProductId =
aliSO.Apttus_Config2__Description__c;
            objAssetWrapperClass.Quantity =
aliSO.Apttus_Config2__Quantity__c;
            objAssetWrapperClass.StartDate =
aliSO.Apttus_Config2__StartDate__c;
            objAssetWrapperClass.EndDate =
aliSO.Apttus_Config2__EndDate__c;
            lstAssetWrapper.add(objAssetWrapperClass);
        }
    }

    // if there are no assets by the search text that the user entered,
    // display an appropriate message.
    else
    {
        lstAssetWrapper.clear();
        ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'No result found. Please try
again.'));
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQGL6XYy.QsjqQQ6RTBnh.1ApTbiqkGAdVz9BS70lxobcyXgHHplmGXAe7p_cf6ziWJ8tpQt_4Q4Bi2VtY
eMyzjhaPbf0</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:getAssetsForSearchText>
         <cpq:accountId>0014P000028PFFS</cpq:accountId>
         <!--<cpq:locationIds></cpq:locationIds>-->
         <cpq:searchText></cpq:searchText>
      </cpq:getAssetsForSearchText>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
xmlns:AssetSearchResultDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <getAssetsForSearchTextResponse>
         <result>
            <AssetSearchResultDO:AssetItems xsi:type="Apttus_Config2__AssetLineItem__
c">
               <Id>a114P000004ie6cQAA</Id>
```

```
                <Apttus_Config2__AccountId__c>0014P000028PFFSQA4</
Apttus_Config2__AccountId__c>
                <Apttus_Config2__AssetStatus__c>Activated</
Apttus_Config2__AssetStatus__c>
                <Apttus_Config2__BasePriceMethod__c>Per Unit</
Apttus_Config2__BasePriceMethod__c>
                <Apttus_Config2__BasePrice__c>160.00000</Apttus_Config2__BasePrice__c>
                <Apttus_Config2__BusinessObjectId__c>a0c4P00000G3AikQAF</
Apttus_Config2__BusinessObjectId__c>
                <Apttus_Config2__ChargeType__c>Standard Price</
Apttus_Config2__ChargeType__c>
                <Apttus_Config2__Description__c>SecureCloud Enterprise</
Apttus_Config2__Description__c>
                <Apttus_Config2__EndDate__c>2017-06-19</Apttus_Config2__EndDate__c>
                <Apttus_Config2__Frequency__c>Monthly</Apttus_Config2__Frequency__c>
                <Apttus_Config2__IsPrimaryLine__c>true</
Apttus_Config2__IsPrimaryLine__c>
                <Apttus_Config2__IsPrimaryRampLine__c>false</
Apttus_Config2__IsPrimaryRampLine__c>
                <Apttus_Config2__IsReadOnly__c>false</Apttus_Config2__IsReadOnly__c>
                <Apttus_Config2__IsUsageTierModifiable__c>false</
Apttus_Config2__IsUsageTierModifiable__c>
                <Apttus_Config2__ItemSequence__c>1</Apttus_Config2__ItemSequence__c>
                <Apttus_Config2__LineNumber__c>1</Apttus_Config2__LineNumber__c>
                <Apttus_Config2__LineType__c>Product/Service</
Apttus_Config2__LineType__c>
                <Apttus_Config2__MustUpgrade__c>false</Apttus_Config2__MustUpgrade__c>
                <Apttus_Config2__NetPrice__c>729600.00000</Apttus_Config2__NetPrice__c>
                <Apttus_Config2__PriceType__c>Recurring</Apttus_Config2__PriceType__c>
                <Apttus_Config2__ProductId__c>01t4P00000884AHQAY</
Apttus_Config2__ProductId__c>
                <Apttus_Config2__ProductId__r xsi:type="Product2">
                   <Id>01t4P00000884AHQAY</Id>
                   <Apttus_Config2__Customizable__c>false</
Apttus_Config2__Customizable__c>
                   <Apttus_Config2__HasAttributes__c>true</
Apttus_Config2__HasAttributes__c>
                   <Apttus_Config2__HasDefaults__c>false</
Apttus_Config2__HasDefaults__c>
                   <Apttus_Config2__HasOptions__c>true</Apttus_Config2__HasOptions__c>
                   <Family>CSD Products</Family>
                </Apttus_Config2__ProductId__r>
                <Apttus_Config2__Quantity__c>400.00000</Apttus_Config2__Quantity__c>
```

```
                <Apttus_Config2__SellingFrequency__c>Monthly</
Apttus_Config2__SellingFrequency__c>
                <Apttus_Config2__SellingTerm__c>12.00000</
Apttus_Config2__SellingTerm__c>
                <Apttus_Config2__StartDate__c>2016-06-20</Apttus_Config2__StartDate__c>
                <Name>SecureCloud Enterprise</Name>
             </AssetSearchResultDO:AssetItems>
             <AssetSearchResultDO:AssetItems xsi:type="Apttus_Config2__AssetLineItem__
c">
                <Id>a114P000004ie6dQAA</Id>
                <Apttus_Config2__AccountId__c>0014P000028PFFSQA4</
Apttus_Config2__AccountId__c>
                <Apttus_Config2__AssetStatus__c>Activated</
Apttus_Config2__AssetStatus__c>
                <Apttus_Config2__BasePriceMethod__c>Per Unit</
Apttus_Config2__BasePriceMethod__c>
                <Apttus_Config2__BasePrice__c>131328.00000</
Apttus_Config2__BasePrice__c>
                <Apttus_Config2__BusinessObjectId__c>a0c4P00000G3AikQAF</
Apttus_Config2__BusinessObjectId__c>
                <Apttus_Config2__ChargeType__c>Subscription Fee</
Apttus_Config2__ChargeType__c>
                <Apttus_Config2__Description__c>SecureCloud Premier Support</
Apttus_Config2__Description__c>
                <Apttus_Config2__EndDate__c>2017-06-19</Apttus_Config2__EndDate__c>
                <Apttus_Config2__Frequency__c>Yearly</Apttus_Config2__Frequency__c>
                <Apttus_Config2__IsPrimaryLine__c>true</
Apttus_Config2__IsPrimaryLine__c>
                <Apttus_Config2__IsPrimaryRampLine__c>false</
Apttus_Config2__IsPrimaryRampLine__c>
                <Apttus_Config2__IsReadOnly__c>false</Apttus_Config2__IsReadOnly__c>
                <Apttus_Config2__IsUsageTierModifiable__c>false</
Apttus_Config2__IsUsageTierModifiable__c>
                <Apttus_Config2__ItemSequence__c>1</Apttus_Config2__ItemSequence__c>
                <Apttus_Config2__LineNumber__c>2</Apttus_Config2__LineNumber__c>
                <Apttus_Config2__LineType__c>Product/Service</
Apttus_Config2__LineType__c>
                <Apttus_Config2__MustUpgrade__c>false</Apttus_Config2__MustUpgrade__c>
                <Apttus_Config2__NetPrice__c>131328.00000</Apttus_Config2__NetPrice__c>
                <Apttus_Config2__PriceType__c>Recurring</Apttus_Config2__PriceType__c>
                <Apttus_Config2__ProductId__c>01t4P00000884tTQAQ</
Apttus_Config2__ProductId__c>
                <Apttus_Config2__ProductId__r xsi:type="Product2">
                   <Id>01t4P00000884tTQAQ</Id>
```

```
                <Apttus_Config2__Customizable__c>false</
Apttus_Config2__Customizable__c>
                <Apttus_Config2__HasAttributes__c>false</
Apttus_Config2__HasAttributes__c>
                <Apttus_Config2__HasDefaults__c>false</
Apttus_Config2__HasDefaults__c>
                <Apttus_Config2__HasOptions__c>false</Apttus_Config2__HasOptions__c>
                <Family>CSD Products</Family>
             </Apttus_Config2__ProductId__r>
             <Apttus_Config2__Quantity__c>1.00000</Apttus_Config2__Quantity__c>
             <Apttus_Config2__SellingFrequency__c>Yearly</
Apttus_Config2__SellingFrequency__c>
             <Apttus_Config2__SellingTerm__c>1.00000</
Apttus_Config2__SellingTerm__c>
             <Apttus_Config2__StartDate__c>2016-06-20</Apttus_Config2__StartDate__c>
             <Name>SecureCloud Premier Support</Name>
          </AssetSearchResultDO:AssetItems>
          <AssetSearchResultDO:AssetItems xsi:type="Apttus_Config2__AssetLineItem__
c">
             <Id>a114P000004ie6eQAA</Id>
             <Apttus_Config2__AccountId__c>0014P000028PFFSQA4</
Apttus_Config2__AccountId__c>
             <Apttus_Config2__AssetStatus__c>Activated</
Apttus_Config2__AssetStatus__c>
             <Apttus_Config2__BasePriceMethod__c>Per Unit</
Apttus_Config2__BasePriceMethod__c>
             <Apttus_Config2__BasePrice__c>500.00000</Apttus_Config2__BasePrice__c>
             <Apttus_Config2__BusinessObjectId__c>a0c4P00000G3AikQAF</
Apttus_Config2__BusinessObjectId__c>
             <Apttus_Config2__ChargeType__c>Subscription Fee</
Apttus_Config2__ChargeType__c>
             <Apttus_Config2__Description__c>SecureTraining Online</
Apttus_Config2__Description__c>
             <Apttus_Config2__EndDate__c>2020-06-19</Apttus_Config2__EndDate__c>
             <Apttus_Config2__Frequency__c>Yearly</Apttus_Config2__Frequency__c>
             <Apttus_Config2__IsPrimaryLine__c>true</
Apttus_Config2__IsPrimaryLine__c>
             <Apttus_Config2__IsPrimaryRampLine__c>false</
Apttus_Config2__IsPrimaryRampLine__c>
             <Apttus_Config2__IsReadOnly__c>false</Apttus_Config2__IsReadOnly__c>
             <Apttus_Config2__IsUsageTierModifiable__c>false</
Apttus_Config2__IsUsageTierModifiable__c>
             <Apttus_Config2__ItemSequence__c>1</Apttus_Config2__ItemSequence__c>
```

```
                <Apttus_Config2__LineNumber__c>3</Apttus_Config2__LineNumber__c>
                <Apttus_Config2__LineType__c>Product/Service</
Apttus_Config2__LineType__c>
                <Apttus_Config2__MustUpgrade__c>false</Apttus_Config2__MustUpgrade__c>
                <Apttus_Config2__NetPrice__c>5000.00000</Apttus_Config2__NetPrice__c>
                <Apttus_Config2__PriceType__c>Recurring</Apttus_Config2__PriceType__c>
                <Apttus_Config2__ProductId__c>01t4P00000884t9QAA</
Apttus_Config2__ProductId__c>
                <Apttus_Config2__ProductId__r xsi:type="Product2">
                   <Id>01t4P00000884t9QAA</Id>
                   <Apttus_Config2__Customizable__c>false</
Apttus_Config2__Customizable__c>
                   <Apttus_Config2__HasAttributes__c>false</
Apttus_Config2__HasAttributes__c>
                   <Apttus_Config2__HasDefaults__c>false</
Apttus_Config2__HasDefaults__c>
                   <Apttus_Config2__HasOptions__c>false</Apttus_Config2__HasOptions__c>
                   <Family>CSD Products</Family>
                </Apttus_Config2__ProductId__r>
                <Apttus_Config2__Quantity__c>10.00000</Apttus_Config2__Quantity__c>
                <Apttus_Config2__SellingFrequency__c>Yearly</
Apttus_Config2__SellingFrequency__c>
                <Apttus_Config2__SellingTerm__c>1.00000</
Apttus_Config2__SellingTerm__c>
                <Apttus_Config2__StartDate__c>2019-06-20</Apttus_Config2__StartDate__c>
                <Name>SecureTraining Online</Name>
             </AssetSearchResultDO:AssetItems>
             <AssetSearchResultDO:HasAssetItems>true</AssetSearchResultDO:HasAssetItem
s>
          </result>
       </getAssetsForSearchTextResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Setting Incentives for Cart

You can use this API to set the incentives for a cart.

API	Signature
setIncentivesForCart	*webService static Apttus_CPQApi.CPQ.SetIncentivesForCartResponseDO setIncentivesForCart(Apttus_CPQApi.CPQ.SetIncentivesForCartRequestDO request)*

Request Parameter		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.SetIncentivesForCartRequestDO	The Set Incentive request data object.

Request Data Object - Apttus_CPQApi.CPQ.SetIncentivesForCartRequestDO		
Name	**Type**	**Description**
CartId	Id	The ID of the cart for which you want to set the incentives.
IncentiveCodes	List	The list incentives code that you want to set for cart or line item.
LineItemId	Id	The ID of the line item for which you want to set the incentive.

Data Object - Apttus_CPQApi.CPQApi.CPQ.SetIncentivesForCartResponseDO		
Name	**Type**	**Description**
IsSuccess	Boolean	Indicates whether the incentives were successfully set on the cart or line item.

Code Sample

The sample code below enables you to set incentives for a cart or a line item.

```
// set incentives for cart
Apttus_CPQApi.CPQ.SetIncentivesForCartRequestDO request = new
 Apttus_CPQApi.CPQ.SetIncentivesForCartRequestDO();
request.CartId = configSO.Id;
request.IncentiveCodes.add(incentiveSO.Apttus_Config2__IncentiveCode__c);

Apttus_CPQApi.CPQ.SetIncentivesForCartResponseDO result =
Apttus_CPQApi.CPQWebService.setIncentivesForCart(request);

// set incentives for cart line item
Apttus_CPQApi.CPQ.SetIncentivesForCartRequestDO request4 = new
 Apttus_CPQApi.CPQ.SetIncentivesForCartRequestDO();
request4.LineItemId = lineItemSO.Id;
request4.IncentiveCodes.add(incentiveSO.Apttus_Config2__IncentiveCode__c);

Apttus_CPQApi.CPQ.SetIncentivesForCartResponseDO result4 =
Apttus_CPQApi.CPQWebService.setIncentivesForCart(request4);
```

Retrieving Incentives Applied on the Cart

You can use this API to fetch the incentives that are applied on a cart.

API	Signature
getAppliedIncentiveCodesForCart	*webService static Apttus_CPQApi.CPQ.GetAppliedIncentiveCodesForCartResponseDO getAppliedIncentiveCodesForCart(Apttus_CPQApi.CPQ.GetAppliedIncentiveCodesForCartRequestDO request)*

Request Parameter		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.GetAppliedIncentiveCodesForCartRequestDO	The incentive code request data object

Request Data Object - Apttus_CPQApi.CPQ.GetAppliedIncentiveCodesForCartRequestDO		
Name	**Type**	**Description**
CartId	Id	The ID of the cart from which you want to fetch the applied promotion

Data Object - Apttus_CPQApi.CPQ.SetIncentivesForCartResponseDO		
Name	**Type**	**Description**
IncentiveCodes	List <IncentiveCodes>	The list of the incentives codes that are applied on the cart

Code Sample

The sample code below enables you to fetch the incentives applied on the cart.

```
Apttus_CPQApi.CPQ.GetAppliedIncentiveCodesForCartRequestDO request3 = new
 Apttus_CPQApi.CPQ.GetAppliedIncentiveCodesForCartRequestDO();
request3.CartId = configSO.Id;

Apttus_CPQApi.CPQ.GetAppliedIncentiveCodesForCartResponseDO result3 =
Apttus_CPQApi.CPQWebService.getAppliedIncentiveCodesForCart(request3);
```

Abandoning a Cart

This deletes the selected cart in *Draft* status.

API	Signature
abandonCart	*webService static Apttus_CPQApi.CPQ.AbandonCartResponseDO abandonCart(Apttus_CPQApi.CPQ.AbandonCartRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.AbandonCartRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.AbandonCartRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.

Response Data Object - Apttus_CPQApi.CPQ.AbandonCartResponseDO		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates whether the cart is deleted successfully.

Code Sample

The sample below enables you to send an API request for abandon cart along with the cart ID. The cart whose id has been passed will be abandoned when the request is successful. The response message in the sample indicates whether the request was successful. When

the user clicks Abandon Cart the *abandonCart* API will be invoked and the cartID associated with the request is abandoned.

```
public void abandonCart()
{
    Apttus_CPQApi.CPQ.AbandonCartRequestDO request = new
 Apttus_CPQApi.CPQ.AbandonCartRequestDO();
    request.CartId = cartId;

    Apttus_CPQApi.CPQ.AbandonCartResponseDO response =
Apttus_CPQApi.CPQWebService.abandonCart(request);
    ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,
'Success: ' + response.IsSuccess));
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote

Request/Response XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQN7k3xhhvMe.j.8gpR.ijcGq77HJeF4MEDfbxbBAqZ8r4WWNTv3OVb6o1bjtHJLbq5mvHcuAH_ie6sTC3
DzrWgnLdUuD</cpq:sessionId>
        </cpq:SessionHeader>
```

```
    </soapenv:Header>
    <soapenv:Body>
        <cpq:abandonCart>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YqyaMAC</cpq1:CartId>
            </cpq:request>
        </cpq:abandonCart>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:AbandonCartResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
    <soapenv:Body>
        <abandonCartResponse>
            <result>
                <AbandonCartResponseDO:IsSuccess>true</AbandonCartResponseDO:IsSucces
s>
            </result>
        </abandonCartResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Finalizing a Cart

This finalizes the cart, synchronizing the cart line items with the quote/proposal.

API	Signature
finalizeCart	*webService static Apttus_CPQApi.CPQ.FinalizeCartResponseDO finalizeCart(Apttus_CPQApi.CPQ.FinalizeCartRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.FinalizeCartRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.FinalizeCartRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.

Response Data Object - Apttus_CPQApi.CPQ.FinalizeCartResponseDO		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates whether finalizing the cart items was successful.

Code Sample

The sample enables you to finalize a cart with a specific cartID and synchronizes the products (line items) added to the cart with the quote or proposal used to generate the cart. You can create a button finalize cart which the user will click once the products in the cart are adjusted and final. Invoke this API on click of the button.

```
1  public void finalizeCart()
2     {
3         // create the finalize cart request
4         Apttus_CpqApi.CPQ.FinalizeCartRequestDO request = new
Apttus_CpqApi.CPQ.FinalizeCartRequestDO();
```

```
5
6           // add request parameters
7           request.CartId = CartId;
8
9           // finalize the cart
10          Apttus_CpqApi.CPQ.FinalizeCartResponseDO response =
Apttus_CpqApi.CPQWebService.finalizeCart(request);
11      }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Adding Products to a Cart OR Adding a Bundle to a Cart
- Updating Price For A Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQN7k3xhhvMe.j.8gpR.ijcGq77HJeF4MEDfbxbBAqZ8r4WWNTv3OVb6o1bjtHJLbq5mvHcuAH_ie6sTC3
DzrWgnLdUuD</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:finalizeCart>
            <cpq:request>
                <cpq1:CartId>a1OZ0000002YqyaMAC</cpq1:CartId>
            </cpq:request>
        </cpq:finalizeCart>
```

```
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:FinalizeCartResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
    <soapenv:Body>
        <finalizeCartResponse>
            <result>
                <FinalizeCartResponseDO:IsSuccess>true</FinalizeCartResponseDO:IsSucc
ess>
            </result>
        </finalizeCartResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Synchronizing a Cart

This is used to sync the shopping cart with the product configuration on the quote record.

API	Signature
synchronizeCart	*webService static Apttus_CPQApi.CPQ.SynchronizeCartResponseDO synchronizeCart(Apttus_CPQApi.CPQ.SynchronizeCartRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.SynchronizeCartRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.SynchronizeCartRequestDO		
Name	**Type**	**Description**
CartId	ID	The ID of the cart (Product Configuration) you want to synchronize.

Response Data Object - Apttus_CPQApi,CPQ.SynchronizeCartResponseDO		
Name	**Type**	**Description**
IsSuccess	Boolean	Indicates whether synchronizing the cart items was successful.

Code Sample

The sample below enables you to synchronize the cart items with a quote. When the customer has finalized the cart, you can synchronize the products in the cart with the quote used to generate the cart. Invoke this API when the user has finalized the cart. The cartID with the updated products is provided as a parameter to the API.

```
1  public void synchronizeCart()
2  {
3      //Pass the cartID as a parameter to the API to synchronize your quote
   with the updated cart items.
4      Apttus_CPQApi.CPQ.SynchronizeCartRequestDO request = new
    Apttus_CPQApi.CPQ.SynchronizeCartRequestDO();
```

```
    request.CartId  = cartId;

    Apttus_CPQApi.CPQ.SynchronizeCartResponseDO response =
Apttus_CPQApi.CPQWebService.synchronizeCart(request);
    ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,
'Success: ' + response.IsSuccess));
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Finalizing a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00D4P000000z7dE!
AQQAQAHIR1W9.1as89Auz.CvNYxqyg56gLLWgUtP5VZxidvTsb1DpQZpmyDuqZOiF4VctBp3jhhJIxG9oRQ4A
4F9h98N0inT</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:synchronizeCart>
            <cpq:request>
                <cpq1:CartId>a1l4P00000Bg7D9</cpq1:CartId>
            </cpq:request>
        </cpq:synchronizeCart>
    </soapenv:Body>
```

```
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
xmlns:SynchronizeCartResponseDO="http://soap.sforce.com/schemas/class/Apttus_CPQApi/
CPQ">
   <soapenv:Body>
      <synchronizeCartResponse>
         <result>
            <SynchronizeCartResponseDO:IsSuccess>true</SynchronizeCartResponseDO:IsSu
ccess>
         </result>
      </synchronizeCartResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Updating Quote Terms

This method enables you to update the expected start date and expected end date of the quote, along with its selling term. This method does the following:

- Updates *Expected Start Date* and *Expected End Date* for the Quote/Proposal object.
- Retrieves the finalized cart associated with the quote.
- Creates a new cart by checking out the finalized cart.
- Updates Expected Start Date & Expected End Date in the cart object. Note this is for the new *Product Configuration* cart object.
- Updates Start Date, End Date, and Selling Term on line item terms that have a valid start date, i.e. when the start date is not null.
- Selling Term may be set to null to clear out the existing selling term which triggers a recalculation of the selling term using the start and end dates. Alternatively, if start date and selling term are provided, the end date is automatically calculated.
- Updates the net price and total price for the cart.
- The ID of the new cart is returned to the caller.
- It is the caller's responsibility to finalize the cart.

The field *QuoteLineItemColl*, which is part of the *Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO*, can accept a list of quote line item sObjects which can hold the data to override one or more line items. For this the line items need to have the following fields populated:

- Start Date
- End Date
- Selling Term
- Derived Form

Line items that match the *Derived From* value gets overridden values from the Quote Line items. The remaining fields will default to values provided in the request object. A null value in the *Start Date* field in the request will prevent the values from defaulting to the line items from the request object.

API	Signature
updateQuoteTerm	*webService static Apttus_CPQApi.CPQ.UpdateQuoteTermResponseDO updateQuoteTerm(Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO	This is the request call made by the method.

Request Data Object - Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO			
Field	**Type**	**Required**	**Description**
EndDate	Date	No	The expected end of the quote.
LineItemColl	Apttus_CPQApi.CPQ.LineItemCollDO	No	The list of line items

Request Data Object - Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO			
Field	**Type**	**Required**	**Description**
QuoteId	ID	Yes	The Id of the quote you want to change the dates and selling term for.
QuoteLineItemCollDO	Apttus_CPQApi.CPQ.QuoteLineItemCollDO	No	The collection of Line items to be updated.
ReCalcSellingTerm	Boolean	No	You can use this method enabled recalculating the selling term
SellingTerm	Decimal	No	The selling term of the quote is typically measured in years, months, or days.
StartDate	Date	Yes	The expected start date of the quote.

Data Object - Apttus_CPQApi.CPQ.LineItemCollDO		
Field	**Type**	**Description**
LineItems	List	The list of line items
Data Object - Apttus_CPQApi.CPQ.QuoteLineItemCollDO		
Field	**Type**	**Description**
LineItems	List	The list of line items

Response Data Object - Apttus_CPQApi.CPQ.UpdateQuoteTermResponseDO		
Field	**Type**	**Description**
CartId	ID	Id of the cart updated.

Code Sample

The sample below enables you update the end date of a cart after the date of acceptance of the proposal. The auto-roll up end date is the sum of the proposal acceptance start date and selling term. When the recipient of the proposal accepts the quote and updates the status, invoke this API to update the start date and selling term of the products associated to the quote. In the sample below, the user enters the quote name in a search field and clicks Search. If a valid quote exists by the name, the quote is fetched. If the user clicks the update Quote Terms button, the API is invoked and the end date is updated.

```
1  public void autoroll()
2  {
3      //Search a quote by its name and fetch its ID
4      List<Apttus_Proposal__Proposal__c> listQuote = [Select Id from
   Apttus_Proposal__Proposal__c where Name=:quoteNumber LIMIT 1];
5
6      //If a valid quote exists with the name execute the if loop
7        if(listQuote.size() > 0)
8      {
9          Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO updateQTRequest = new
    Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO();
10         updateQTRequest.QuoteId = listQuote[0].Id;
11
12         //Start date=current date entered in the from field of the
   proposal.
13         updateQTRequest.StartDate = fromdate;
14
15         //End date=startdate+selling term
16         updateQTRequest.EndDate =
   updateQTRequest.StartDate.addMonths(sellingTerm);
17         /**updateQTRequest.EndDate = endDate;**/
18
19         /*updateQTRequest.SellingTerm = 12;*/
20
```

```
        //Execute the API to update quote term
        Apttus_CPQApi.CPQ.UpdateQuoteTermResponseDO updateQTReponse =
Apttus_CPQApi.CPQWebService.updateQuoteTerm(updateQTRequest);

        ID cartID = updateQTReponse.cartID;

        //Invoke the finalize cart request to update end date with the
cart
        Apttus_CPQApi.CPQ.FinalizeCartRequestDO finalizeRequest = new
 Apttus_CPQApi.CPQ.FinalizeCartRequestDO();
        finalizeRequest.CartId = cartId;

        //Invoke the finalize cart API to update the cart details
        Apttus_CPQApi.CPQ.FinalizeCartResponseDO finalizeResponse =
Apttus_CPQApi.CPQWebService.finalizeCart(finalizeRequest);
    }
}
```

The sample below enables you to update the start date and end date of a cart after the date of acceptance of the proposal. When the recipient of the proposal accepts the quote and updates the status, invoke this API to update the start date, end date and selling term of the products associated to the quote. In the sample below, the user enters the quote name in a search field and clicks Search. If a valid quote exists by the name, the quote is fetched. If the user clicks the update Quote Terms button, the API is invoked and the start date is updated. The validity i.e the selling term has to be updated on a pro-rata basis.

```
public void prorata()
{
    //Search a quote by its name and fetch its ID
    List<Apttus_Proposal__Proposal__c> listQuote = [Select Id from
Apttus_Proposal__Proposal__c where Name=:quoteNumber LIMIT 1];



    //If a valid quote exists with the name execute the if loop
    if(listQuote.size() > 0)
    {

        Apttus_Proposal__Proposal__c propSO = [select id,
Apttus_Proposal__ExpectedStartDate__c, Apttus_Proposal__ExpectedEndDate__c
from Apttus_Proposal__Proposal__c      where id = :listQuote[0].Id limit 1]
;
        Date startDate = propSO.Apttus_Proposal__ExpectedStartDate__c;
```

```
        Date endDate = propSO.Apttus_Proposal__ExpectedEndDate__c;

        Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO updateQTRequest = new
 Apttus_CPQApi.CPQ.UpdateQuoteTermRequestDO();
        updateQTRequest.QuoteId = listQuote[0].Id;
        //Fetch the start date of the proposal
        updateQTRequest.StartDate = fromdate;

        /**updateQTRequest.EndDate =
updateQTRequest.StartDate.addMonths(12);**/
        //Fetch the end date of the proposal
        updateQTRequest.EndDate = endDate ;
        updateQTRequest.SellingTerm = null;

        //Selling Term to be recalculated based on the start date and end
date
        updateQTRequest.ReCalcSellingTerm = true;
        /*updateQTRequest.SellingTerm = 12;*/

        //Execute the API to update quote term
        Apttus_CPQApi.CPQ.UpdateQuoteTermResponseDO updateQTReponse =
Apttus_CPQApi.CPQWebService.updateQuoteTerm(updateQTRequest);
        ID cartID = updateQTReponse.cartID;

        //Invoke the finalize cart request to update selling term of
product with the cart
        Apttus_CPQApi.CPQ.FinalizeCartRequestDO finalizeRequest = new
 Apttus_CPQApi.CPQ.FinalizeCartRequestDO();
        finalizeRequest.CartId = cartId;
        Apttus_CPQApi.CPQ.FinalizeCartResponseDO finalizeResponse =
Apttus_CPQApi.CPQWebService.finalizeCart(finalizeRequest);
    }
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Finalizing a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:cpq1="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQ">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQLC3rla0VIl_gk7v0A9K1n7dLEcCAANQHwNfXAPFKZEH292fAdeal5Nps8X2klNu98fcp6usnVFmzKARw
E99xq_dE8U6</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:updateQuoteTerm>
            <cpq:request>
                <cpq1:EndDate>2020-12-30</cpq1:EndDate>
                <cpq1:QuoteId>a0eZ0000005pG5NIAU</cpq1:QuoteId>
                <cpq1:QuoteLineItemColl>
                    <cpq1:LineItems>
                        <cpq:Apttus_QPConfig__EndDate__c>2020-12-30</cpq:Apttus_QPCon
fig__EndDate__c>
                        <cpq:Apttus_QPConfig__SellingTerm__c>1</cpq:Apttus_QPConfig__
SellingTerm__c>
                        <cpq:Apttus_QPConfig__StartDate__c>2020-11-01</cpq:Apttus_QPC
onfig__StartDate__c>
                        <cpq:Apttus_QPConfig__DerivedFromId__c>a19Z0000006sPdw</cpq:A
pttus_QPConfig__DerivedFromId__c>
                    </cpq1:LineItems>
                </cpq1:QuoteLineItemColl>
                <cpq1:ReCalcSellingTerm>false</cpq1:ReCalcSellingTerm>
                <cpq1:SellingTerm>1</cpq1:SellingTerm>
                <cpq1:StartDate>2020-11-01</cpq1:StartDate>
            </cpq:request>
        </cpq:updateQuoteTerm>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService"
    xmlns:UpdateQuoteTermResponseDO="http://soap.sforce.com/schemas/class/
Apttus_CPQApi/CPQ">
    <soapenv:Body>
        <updateQuoteTermResponse>
            <result>
                <UpdateQuoteTermResponseDO:CartId>a1OZ0000002hsIXMAY</
UpdateQuoteTermResponseDO:CartId>
            </result>
        </updateQuoteTermResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Modifying Assets (Deprecated)

This method enables you to complete standard asset based ordering actions of renew, terminate, amend, upgrade and increment.

This method creates the cart line item and associates it to the corresponding asset id. Actions can be performed on a single line item or a selected set of asset line items. Actions can also be performed on standalone product or bundle product.

When doing a renewal, you can use any of the standard adjustment types, including *% Uplift* which is used to pass an uplift percentage that is applied to the *base price* of the renewed line item, resulting in an updated *net renewal* price.

Note

CartId, *AssetLineItemId__c*, and *AssetActions* are required parameters, the use of other parameters depends on your business case.

API	Signature
modifyAsset	*webService static Apttus_CPQApi.CPQ.ModifyAssetResponseDO modifyAsset(Apttus_CPQApi.CPQ.ModifyAssetRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.ModifyAssetRequestDO	This is the request call made by the method.

Request Data Object - Apttus_CPQApi.CPQ.ModifyAssetRequestDO		
Field	**Type**	**Description**
AssetActions	List<Apttus_CPQApi.CPQ.AssetActionDO>	The list of asset actions, which is either *Renew*, *Amend*, *Increment*, *Cancel*, *Upgrade*.
CartId	ID	The Id of cart.

Response Data Object - Apttus_CPQApi.CPQ.ModifyAssetResponseDO		
Field	**Type**	**Description**
AssetActions	List<Apttus_CPQApi.CPQ.AssetActionDO>	The list of asset actions, which is either *Renew*, *Amend*, *Increment*, *Cancel*, *Upgrade*.
LineNumber	List<Decimal>	The asset line number.

Data Object - Apttus_CPQApi.CPQ.AssetActionDO		
Field	**Type**	**Description**
AdjustmentAmount	Decimal	Enter the value (number) for the adjustment. How the value is interpreted is based on the Adjustment Type. For example, 10 with an Adjustment Type of *% Uplift* means the price will be cut in half, whereas that same value of 10 with an Adjustment Type of *Price Override* means the price will become 10.
AdjustmentType	String	This is the type of adjustment that will be used. You can select from types such as % Discount, Discount Amount, and Price Override. You specific implementation may have more or less types.
AssetAction	String	This is the specific asset action for the asset line item, referenced by AssetLineItemId.
AssetLineItemId	ID	This is the ID for the asset line item that you are going to renew, amend, increment, or cancel. This is mandatory.
Comments	String	Text string containing comments.
CustomData	Apttus_Config2__LineItem__c	This can be used to include custom fields in line item object.
CustomFields	List	This can be used to include custom fields you have added to the asset object.
EndDate	Date	This is the end date for the asset action. If you are terminating an asset you only need an end date.
Message	String	Text string containing comments.
Pending	Boolean	This indicates whether an asset action is pending.

Data Object - Apttus_CPQApi.CPQ.AssetActionDO		
Field	**Type**	**Description**
Quantity	Decimal	The amount of assets.
SellingTerm	Decimal	This is the selling term of the asset line item that you want to renew, amend, or increment.
StartDate	Date	This is the start date for the asset action.

Code Sample

You can amend a list of standalone or bundle products. You can change the dates, quantity, or some options for a product using the Amend API. You can also cancel or change options in an Option Group using Amend.

The Sample below enables you to change amend the start and end date for selected assets of an account. The response returns the lineitem number of the asset to be updated. The assetAction status, the status of the asset is modified and updated. When the customer clicks Amend, you can invoke the modifyAssets API and choose to amend the date, quantity.

```
1  public void Amend()
2  {
3  //Create an assetAction variable with value as amend
4      string assetAction = 'Amend';
5
6      List<Apttus_CPQApi.CPQ.AssetActionDO> assetActDOList = new
    List<Apttus_CPQApi.CPQ.AssetActionDO>();
7
8  //If Asset is selected in the cart page, execute the loop
9  for(AssetWrapperClass objAssetWrapper : lstAssetWrapper)
10     {
11         if(objAssetWrapper.selected)
12         {
13             Apttus_CPQApi.CPQ.AssetActionDO assetActDO = new
    Apttus_CPQApi.CPQ.AssetActionDO();
14             assetActDO.AssetAction = assetAction;
15             assetActDO.AssetLineItemId = objAssetWrapper.AssetId ;
16 //Change the end date to five months from start date
```

```
            assetActDO.EndDate = Date.today().addMonths(5);
//assetAction pending is true
            assetActDO.Pending = true;
            assetActDO.Quantity = objAssetWrapper.Quantity;
//Amend the start date to include today's date
            assetActDO.StartDate = Date.today().addMonths(0);

            List<String> customFields = new List<String>();
            customFields.add('Apttus_Config2__Comments__c');
            customFields.add('Apttus_Config2__PricingStatus__c');

            assetActDO.CustomFields = customFields;

            Apttus_Config2__LineItem__c liSO = new
 Apttus_Config2__LineItem__c();
            liSO.Apttus_Config2__Comments__c = 'Comments Added my Modify
Asset API Code';
            liSO.Apttus_Config2__PricingStatus__c = 'Complete';
            assetActDO.CustomData = liSO;

            assetActDOList.add(assetActDO);
        }
    }

//Execute the API for a specific cart using CartID
    Apttus_CPQApi.CPQ.ModifyAssetRequestDO modifyRequest = new
 Apttus_CPQApi.CPQ.ModifyAssetRequestDO();
    modifyRequest.CartId = cartID;
    modifyRequest.AssetActions = assetActDOList;

    Apttus_CPQApi.CPQ.ModifyAssetResponseDO modifyReponse =
Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);

    List<Decimal> lineNumberList = modifyReponse.LineNumbers;

    List<Apttus_CPQApi.CPQ.AssetActionDO> assetList =
modifyReponse.AssetActions;

//Signifies the status of the assetAction
    Boolean bIsPending = false;

    Integer iCounter = 0;

```

```
55      while (iCounter < 1000)
56      {
57          bIsPending = false;
58          for(Apttus_CPQApi.CPQ.AssetActionDO objAsset: assetList)
59          {
60              if(objAsset.Pending == true)
61              {
62                  bIsPending = true;
63                  break;
64              }
65          }
66  //If the asset action is pending execute the loop below to change the
    status
67          if(bIsPending == true)
68          {
69              modifyReponse =
    Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);
70              lineNumberList = modifyReponse.LineNumbers;
71              assetList = modifyReponse.AssetActions;
72          }
73          else
74          {
75              break;
76          }
77          iCounter++;
78      }
79
80      bindAssets();
81  }
82  //This function enables you to bind updated assets to an account
    dynamically.
83  public void bindAssets()
84  {
85      List<Apttus_Config2__AssetLineItem__c> aliSOList = [select id,
    Apttus_Config2__Description__c,
86      Apttus_Config2__Quantity__c, Apttus_Config2__StartDate__c,
    Apttus_Config2__EndDate__c from Apttus_Config2__AssetLineItem__c
87      where Apttus_Config2__AccountID__c = :accountID];
88
89      lstAssetWrapper = new List<AssetWrapperClass>();
90
91      for(Apttus_Config2__AssetLineItem__c aliSO : aliSOList)
92      {
93          AssetWrapperClass objAssetWrapperClass = new AssetWrapperClass ();
94          objAssetWrapperClass.AssetId = aliSO.ID;
```

```
        objAssetWrapperClass.ProductId =
aliSO.Apttus_Config2__Description__c;
        objAssetWrapperClass.Quantity = aliSO.Apttus_Config2__Quantity__c;
        objAssetWrapperClass.StartDate =
aliSO.Apttus_Config2__StartDate__c;
        objAssetWrapperClass.EndDate = aliSO.Apttus_Config2__EndDate__c;

        lstAssetWrapper.Add(objAssetWrapperClass);
    }
}
```

You can renew a list of standalone or bundled products. You can choose to renew recurring items, overall bundle, and recurring options. You can also cancel or change the quantity, price, or an entire option. The sample below enables you to renew the assets. When the user selects an asset and clicks renew, invoke this API and use the sample below to extend the end date of the assets by 12 months.

```
public void ReNew()
{
//Declare assetAction variable with value as Renew
    string assetAction = 'Renew';
    List<Apttus_CPQApi.CPQ.AssetActionDO> assetActDOList = new
 List<Apttus_CPQApi.CPQ.AssetActionDO>();

    //For a selected asset you can execute the loop
    for(AssetWrapperClass objAssetWrapper : lstAssetWrapper)
    {
        if(objAssetWrapper.selected)
        {
            Apttus_CPQApi.CPQ.AssetActionDO assetActDO = new
 Apttus_CPQApi.CPQ.AssetActionDO();
            assetActDO.AssetAction = assetAction;
            assetActDO.AssetLineItemId = objAssetWrapper.AssetId ;

            //Enables you to set end date as 12 months from the existing
end date
            assetActDO.EndDate = objAssetWrapper.EndDate.addMonths(12);
            assetActDO.Pending = true;
            assetActDO.Quantity = objAssetWrapper.Quantity;
```

```
20
21              //Increment start date by One day from the existing start date
22              assetActDO.StartDate = objAssetWrapper.EndDate.addDays(1);
23
24              List<String> customFields = new List<String>();
25              customFields.add('Apttus_Config2__Comments__c');
26              customFields.add('Apttus_Config2__PricingStatus__c');
27              assetActDO.CustomFields = customFields;
28
29              //Fetch asset line item number
30              Apttus_Config2__LineItem__c liSO = new
 Apttus_Config2__LineItem__c();
31              liSO.Apttus_Config2__Comments__c = 'Comments Added my Modify
Asset API Code';
32              liSO.Apttus_Config2__PricingStatus__c = 'Complete';
33              assetActDO.CustomData = liSO;
34              assetActDOList.add(assetActDO);
35          }
36      }
37      //Invoke the modify assets API and specify the cart for which the API
is invoked.
38      Apttus_CPQApi.CPQ.ModifyAssetRequestDO modifyRequest = new
 Apttus_CPQApi.CPQ.ModifyAssetRequestDO();
39      modifyRequest.CartId = cartID;
40      modifyRequest.AssetActions = assetActDOList;
41
42      Apttus_CPQApi.CPQ.ModifyAssetResponseDO modifyReponse =
Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);
43      List<Decimal> lineNumberList = modifyReponse.LineNumbers;
44      List<Apttus_CPQApi.CPQ.AssetActionDO> assetList =
modifyReponse.AssetActions;
45      Boolean bIsPending = false;
46      Integer iCounter = 0;
47      while (iCounter < 1000)
48      {
49          bIsPending = false;
50          for(Apttus_CPQApi.CPQ.AssetActionDO objAsset: assetList)
51          {
52              if(objAsset.Pending == true)
53              {
54                  bIsPending = true;
55                  break;
56              }
57          }
58
```

```
59          //If asset Action is pending execute the following loop
60          if(bIsPending == true)
61          {
62              modifyReponse =
Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);
63              lineNumberList = modifyReponse.LineNumbers;
64              assetList = modifyReponse.AssetActions;
65          }
66          else
67          {
68              break;
69          }
70          iCounter++;
71      }
72      bindAssets();
73  }
74
75  //This function enables you to bind updated asset.
76  public void bindAssets()
77  {
78      List<Apttus_Config2__AssetLineItem__c> aliSOList = [select id,
Apttus_Config2__Description__c,
79      Apttus_Config2__Quantity__c, Apttus_Config2__StartDate__c,
Apttus_Config2__EndDate__c from Apttus_Config2__AssetLineItem__c
80      where Apttus_Config2__AccountID__c = :accountID];
81
82      lstAssetWrapper = new List<AssetWrapperClass>();
83
84      for(Apttus_Config2__AssetLineItem__c aliSO : aliSOList)
85      {
86          AssetWrapperClass objAssetWrapperClass = new AssetWrapperClass ();
87          objAssetWrapperClass.AssetId = aliSO.ID;
88          objAssetWrapperClass.ProductId =
aliSO.Apttus_Config2__Description__c;
89          objAssetWrapperClass.Quantity = aliSO.Apttus_Config2__Quantity__c;
90          objAssetWrapperClass.StartDate =
aliSO.Apttus_Config2__StartDate__c;
91          objAssetWrapperClass.EndDate = aliSO.Apttus_Config2__EndDate__c;
92
93          lstAssetWrapper.Add(objAssetWrapperClass);
94      }
95  }
```

You can increment the quantity or the dates of the asset items using the increment asset action. For example, the user selects an asset item and clicks Increment. You can use the sample below to update the quantity and the end date by five months and bind the updated assets to the account.

```
public void Increment()
{
    //Declare the assetAction variable with value as Increment
    string assetAction = 'Increment';

    List<Apttus_CPQApi.CPQ.AssetActionDO> assetActDOList = new
 List<Apttus_CPQApi.CPQ.AssetActionDO>();

    for(AssetWrapperClass objAssetWrapper : lstAssetWrapper)
    {
        if(objAssetWrapper.selected)
        {
            //For a selected asset increment the quantity by 5 and
increment the end date by 5.
            Apttus_CPQApi.CPQ.AssetActionDO assetActDO = new
 Apttus_CPQApi.CPQ.AssetActionDO();
            assetActDO.AssetAction = assetAction;
            assetActDO.AssetLineItemId = objAssetWrapper.AssetId ;
            assetActDO.EndDate = Date.today().addMonths(5);
            assetActDO.Pending = true;
            assetActDO.Quantity = objAssetWrapper.Quantity + 5;
            assetActDO.StartDate = Date.today().addMonths(0);

            List<String> customFields = new List<String>();
            customFields.add('Apttus_Config2__Comments__c');
            customFields.add('Apttus_Config2__PricingStatus__c');

            assetActDO.CustomFields = customFields;
            //Line number of the asset items
            Apttus_Config2__LineItem__c liSO = new
 Apttus_Config2__LineItem__c();
            liSO.Apttus_Config2__Comments__c = 'Comments Added my Modify
Asset API Code';
            liSO.Apttus_Config2__PricingStatus__c = 'Complete';
            assetActDO.CustomData = liSO;
```

```
            assetActDOList.add(assetActDO);
        }
    }
    //Invoke the modifyAsset API to increment the quantity and end date
    Apttus_CPQApi.CPQ.ModifyAssetRequestDO modifyRequest = new
 Apttus_CPQApi.CPQ.ModifyAssetRequestDO();
    modifyRequest.CartId = cartID;
    modifyRequest.AssetActions = assetActDOList;

    Apttus_CPQApi.CPQ.ModifyAssetResponseDO modifyReponse =
Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);
    List<Decimal> lineNumberList = modifyReponse.LineNumbers;
    List<Apttus_CPQApi.CPQ.AssetActionDO> assetList =
modifyReponse.AssetActions;
    Boolean bIsPending = false;
    Integer iCounter = 0;
    while (iCounter < 1000)
    {
        bIsPending = false;
        for(Apttus_CPQApi.CPQ.AssetActionDO objAsset: assetList)
        {
            if(objAsset.Pending == true)
            {
                bIsPending = true;
                break;
            }
        }

        //If asset action is pending, execute this loop
        if(bIsPending == true)
        {
            modifyReponse =
Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);
            lineNumberList = modifyReponse.LineNumbers;
            assetList = modifyReponse.AssetActions;
        }
        else
        {
            break;
        }
        iCounter++;
    }

```

```
    bindAssets();
}

//Bind updated asset items
public void bindAssets()
{
    List<Apttus_Config2__AssetLineItem__c> aliSOList = [select id,
Apttus_Config2__Description__c,
    Apttus_Config2__Quantity__c, Apttus_Config2__StartDate__c,
Apttus_Config2__EndDate__c from Apttus_Config2__AssetLineItem__c
    where Apttus_Config2__AccountID__c = :accountID];

    lstAssetWrapper = new List<AssetWrapperClass>();

    for(Apttus_Config2__AssetLineItem__c aliSO : aliSOList)
    {
        AssetWrapperClass objAssetWrapperClass = new AssetWrapperClass ();
        objAssetWrapperClass.AssetId = aliSO.ID;
        objAssetWrapperClass.ProductId =
aliSO.Apttus_Config2__Description__c;
        objAssetWrapperClass.Quantity = aliSO.Apttus_Config2__Quantity__c;
        objAssetWrapperClass.StartDate =
aliSO.Apttus_Config2__StartDate__c;
        objAssetWrapperClass.EndDate = aliSO.Apttus_Config2__EndDate__c;

        lstAssetWrapper.Add(objAssetWrapperClass);
    }
}
```

The terminate asset action enables you to terminate or cancel the selected asset items. The sample below enables you to terminate the asset and change its status by setting the end date as current date.

```
public void Terminate()
{
    //Declaring variable assetAction and assigning it a value Cancel
    string assetAction = 'Cancel';
    List<Apttus_CPQApi.CPQ.AssetActionDO> assetActDOList = new
 List<Apttus_CPQApi.CPQ.AssetActionDO>();

    //To terminate a selected Asset, set the end date as today
    for(AssetWrapperClass objAssetWrapper : lstAssetWrapper)
```

```
    {
        if(objAssetWrapper.selected)
        {
            Apttus_CPQApi.CPQ.AssetActionDO assetActDO = new
 Apttus_CPQApi.CPQ.AssetActionDO();
            assetActDO.AssetAction = assetAction;
            assetActDO.AssetLineItemId = objAssetWrapper.AssetId ;
            assetActDO.EndDate = Date.today().addMonths(0);
            assetActDO.Pending = true;
            assetActDO.Quantity = objAssetWrapper.Quantity;
            //assetActDO.StartDate = Date.today().addMonths(0);

            List<String> customFields = new List<String>();
            customFields.add('Apttus_Config2__Comments__c');
            customFields.add('Apttus_Config2__PricingStatus__c');

            assetActDO.CustomFields = customFields;

            Apttus_Config2__LineItem__c liSO = new
 Apttus_Config2__LineItem__c();
            liSO.Apttus_Config2__Comments__c = 'Comments Added my Modify
Asset API Code';
            liSO.Apttus_Config2__PricingStatus__c = 'Complete';
            assetActDO.CustomData = liSO;

            assetActDOList.add(assetActDO);
        }
    }
    //For a specific cart ID, invoke the API and modify the assets based
on the terminate function above.
    Apttus_CPQApi.CPQ.ModifyAssetRequestDO modifyRequest = new
 Apttus_CPQApi.CPQ.ModifyAssetRequestDO();
    modifyRequest.CartId = cartID;
    modifyRequest.AssetActions = assetActDOList;

    Apttus_CPQApi.CPQ.ModifyAssetResponseDO modifyReponse =
Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);

    List<Decimal> lineNumberList = modifyReponse.LineNumbers;

    List<Apttus_CPQApi.CPQ.AssetActionDO> assetList =
modifyReponse.AssetActions;

```

```
45      Boolean bIsPending = false;
46
47      Integer iCounter = 0;
48
49      while (iCounter < 1000)
50      {
51          bIsPending = false;
52          for(Apttus_CPQApi.CPQ.AssetActionDO objAsset: assetList)
53          {
54              if(objAsset.Pending == true)
55              {
56                  bIsPending = true;
57                  break;
58              }
59          }
60          //if asset action is pending execute the loop below
61          if(bIsPending == true)
62          {
63              modifyReponse =
Apttus_CPQApi.CPQWebservice.modifyAsset(modifyRequest);
64              lineNumberList = modifyReponse.LineNumbers;
65              assetList = modifyReponse.AssetActions;
66          }
67          else
68          {
69              break;
70          }
71          iCounter++;
72      }
73
74      bindAssets();
75  }
76
77  //Use the function below to bind the updated asset items.
78  public void bindAssets()
79  {
80      List<Apttus_Config2__AssetLineItem__c> aliSOList = [select id,
Apttus_Config2__Description__c,
81      Apttus_Config2__Quantity__c, Apttus_Config2__StartDate__c,
Apttus_Config2__EndDate__c from Apttus_Config2__AssetLineItem__c
82      where Apttus_Config2__AccountID__c = :accountID];
83
84      lstAssetWrapper = new List<AssetWrapperClass>();
85
86      for(Apttus_Config2__AssetLineItem__c aliSO : aliSOList)
```

```
    {
        AssetWrapperClass objAssetWrapperClass = new AssetWrapperClass ();
        objAssetWrapperClass.AssetId = aliSO.ID;
        objAssetWrapperClass.ProductId =
aliSO.Apttus_Config2__Description__c;
        objAssetWrapperClass.Quantity = aliSO.Apttus_Config2__Quantity__c;
        objAssetWrapperClass.StartDate =
aliSO.Apttus_Config2__StartDate__c;
        objAssetWrapperClass.EndDate = aliSO.Apttus_Config2__EndDate__c;

        lstAssetWrapper.Add(objAssetWrapperClass);
    }
}
```

The upgrade asset action enables you to update selected asset items. The sample below enables you to upgrade the asset. Using the sample below you can fetch the asset items using the getAssetsForSearchText API and upgrade the selected assetLineItems using AssetAction = Upgrade. You can upgrade the date and the quantity for an asset line item.

```
public static void processForUpgrade(ID cartId, ID accountId)
{
    //Declaration
    Set<ID> assetLineItemIds = new Set<ID>();

    Apttus_CPQApi.CPQ.AssetSearchResultDO result
     = Apttus_CPQApi.CPQWebService.getAssetsForSearchText(accountId, null, null);

    Apttus_CPQApi.CPQ.ModifyAssetRequestDO assetRequest = new
 Apttus_CPQApi.CPQ.ModifyAssetRequestDO();
    assetRequest.CartId = cartId;


    for (Apttus_Config2__AssetLineItem__c assetItemSO : result.AssetItems)
    {
        Apttus_CPQApi.CPQ.AssetActionDO assetAction = new
Apttus_CPQApi.CPQ.AssetActionDO();
        assetAction.AssetAction = 'Upgrade';
        assetAction.AssetLineItemId = assetItemSO.Id;
        assetAction.Quantity = assetItemSO.Apttus_Config2__Quantity__c.intValue();
        assetAction.EndDate = Date.today();
        assetAction.Pending = true;
```

```
        assetLineItemIds.add(assetItemSO.Id);
        assetRequest.AssetActions.add(assetAction);
    }

    Apttus_CPQApi.CPQ.ModifyAssetResponseDO assetResponse =
Apttus_CPQApi.CPQWebService.modifyAsset(assetRequest);

    Set<Decimal> lineNumbers = new Set<Decimal>(assetResponse.LineNumbers);

    for (Apttus_Config2__LineItem__c lineSO : [Select Id,
Apttus_Config2__AssetLineItemId__c from Apttus_Config2__LineItem__c Where
Apttus_Config2__ConfigurationId__c = :cartId])
    {

    }
}
```

Computing Shipping for Cart Line Items

This API enables you to calculate the shipping amount for an entire order and does not display the breakup for each line item.

API	Signature
computeShippingForCart	*webService static Apttus_CPQApi.CPQ.ComputeShippingResponseDO computeShippingForCart(Apttus_CPQApi.CPQ.ComputeShippingRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.ComputeShippingRequestDO	The compute shipping request.

Request Data Object - Apttus_CPQApi.CPQ.ComputeShippingRequestDO		
Field	**Type**	**Description**
CartID	ID	The id of the cart.

Response Data Object - Apttus_CPQApi.CPQ.ComputeShippingResponseDO		
Field	**Type**	**Description**
ShippingResults	Map<ID, Apttus_Config2.CustomClass.ShippingResult>	The shipping result object populated by the line item id. This is only available when called from Apex.
TotalShippingItem	Apttus_Config2__LineItem__c	The line item with total taxes.

Data Object - Apttus_Config2.CustomClass.ShippingResult		
Field	**Type**	**Description**
Handback	Object	Any Apex Object passed by the calling class and passed back to the calling class in the result. Handback is used only in case when the shipping charge needs to be calculated for an object other than Cart line item e.g. Billing Schedule line items. • For calculating shipping charges for a cart, there is no need to pass the Handback object. It can be blank • For calculating the shipping charges for objects other than line items, "Item" field will be blank and the actual object will be passed as handback object to identify the object for which shipping charges are calculated.

Data Object - Apttus_Config2.CustomClass.ShippingResult		
Field	**Type**	**Description**
ShippingCharge	Decimal	Shipping amount total of all the cart line items. This is the total shipping amount for the line item. Line item is identified by the sequence in which the shipping result is added. This needs to be calculated and populated by the implementation.

Code Sample

The sample below enables you to compute shipping for a cart with a valid cartID. Using the sample below, you can compute shipping for the entire cart using the values fetched from the shipping callback class. If the shipping charge is location dependent, ensure that the account associated with the order or proposal has a valid shipping to and billing to address. For more information about Tax and Shipping scenarios refer, Updating Taxes and Shipping for an Order.

```
public void computeShippingForCart()
{
    if(String.isNotBlank(cartId))
    {
        // Start timer
        startTime = System.currentTimeMillis();

        // Create the request
        Apttus_CPQApi.CPQ.ComputeShippingRequestDO request = new
 Apttus_CPQApi.CPQ.ComputeShippingRequestDO();

        // Add request parameters
        request.CartId = cartId;

        // Compute Shipping For Cart
        Apttus_CPQApi.CPQ.ComputeShippingResponseDO result =
Apttus_CPQApi.CPQWebService.computeShippingForCart(request);

        // End timer
        ResponseTime('computeShippingForCart');

        // Get Total Tax Item
```

```
21          ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Total Shipping Item : ' +
result.TotalShippingItem));
22
23          Map<ID, Apttus_Config2.CustomClass.ShippingResult> shippingResults
= new Map<ID, Apttus_Config2.CustomClass.ShippingResult>();
24
25          shippingResults = result.ShippingResults;
26
27          for(Id sId : shippingResults.keySet()) {
28
29              Apttus_Config2.CustomClass.ShippingResult shippingResult =
shippingResults.get(sId);
30
31              ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Shipping Charges : ' +
shippingResult.ShippingCharge));
32          }
33      }
34      else
35      {
36          ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Invalid or blank cart Id.'));
37      }
38  }
```

Computing Taxes for Cart Line Items

This API enables you to calculate tax breakups for cart line item.

API	Signature
computeTaxForCart	*webService static Apttus_CPQApi.CPQ.ComputeTaxResponseDO computeTaxForCart(Apttus_CPQApi.CPQ.ComputeTaxRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.ComputeTaxRequestDO	The compute tax request object.

Request Data Object - Apttus_CPQApi.CPQ.ComputeTaxRequestDO		
Field	**Type**	**Description**
CartID	ID	The id of the cart.

Response Data Object - Apttus_CPQApi.CPQ.ComputeTaxResponseDO		
Field	**Type**	**Description**
TaxResults	Map<ID, Apttus_Config2.CustomClass.TaxResult>	The Tax Result object populated by the line item id. This is only available when called from Apex.
TotalTaxItem	Apttus_Config2__LineItem__c	The line item with total taxes.

Data Object - Apttus_Config2.CustomClass.TaxResult		
Field	**Type**	**Description**
Handback	Object	Any Apex Object passed by the calling class and passed back to the calling class in the result. Handback is used only in case when the tax needs to be calculated for an object other than Cart line item e.g. Billing Schedule line items. • For calculating tax for cart line items, there is no need to pass the Handback object. It can be blank • For calculating the tax for object other than line items, "Item" field will be blank and the actual object will be passed as handback object to identify object for which tax is calculated.
TaxAmount	Decimal	This is the total amount for the line item. This is sum of the breakup tax amounts for the line item. Line item is identified by the sequence in which the tax result is added. This needs to be calculated and populated by the implementation.
TaxBreakups	List<Apttus_Config2__TaxBreakup_c>	Tax breakup provide the different components of the tax for a given line item. This needs to be implemented and provided by the implementation. Tax breakup has structure specified below.

Data Object - Apttus_Config2__TaxBreakup_c		
Field	**Type**	**Description**
BreakupType	String	If you specify the breakup type as Detail, then the quote or order line items show the detailed tax break up for a cart line item.
TaxAmount	Decimal	The tax amount calculated based on the formulae defined in the callback.
TaxAppliesTo	String	The field on which the tax is applied to. For example, net price.
TaxRate	Decimal	Tax percentage to be applied
TaxType	String	Type of tax to be applied. For example, sales, custom, excise.

Tax Breakup Type Example

Breakup Type	Tax Type	Tax Rate	Tax Applies to	Tax Amount
Detail	State Tax	10	Net Price	100
Detail	City Tax	1	Net Price	10
Total				110

Code Sample

The sample below enables you to compute tax for a cart with a valid cartID. Using the sample below, you can compute tax for the entire cart using the values fetched from the tax callback class. The fields Taxable, Tax Inclusive, and Tax Code are inherited from the quote or order line item. The tax code must be populated on the Price list item for each product and charge type combination. The Account and Account Location has a Tax Exempt indicator that overrides everything. For example, if the Account has a Tax Exempt flag enabled, then Tax is not calculated for the account. The system gives preference to location unless it is blank. If a product is not taxable or tax inclusive, the system will skip that line item.

For more information about Tax and Shipping scenarios refer, Updating Taxes and Shipping for an Order.

```
public void computeTaxForCart()
{
    if(String.isNotBlank(cartId))
    {

        // Create the request
        Apttus_CPQApi.CPQ.ComputeTaxRequestDO request = new
 Apttus_CPQApi.CPQ.ComputeTaxRequestDO();

        // Add request parameters
        request.CartId = cartId;

        // Compute Tax for cart
        Apttus_CPQApi.CPQ.ComputeTaxResponseDO result =
Apttus_CPQApi.CPQWebService.computeTaxForCart(request);


        // Get Total Tax Item
        ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Total Tax Item : ' +
result.TotalTaxItem));

        Map<ID, Apttus_Config2.CustomClass.TaxResult> taxResults = new
 Map<ID, Apttus_Config2.CustomClass.TaxResult>();
        lstTaxBreakUpWrapper = new List<TaxBreakUpWrapper>();

        taxResults = result.TaxResults;

        for(ID tId : taxResults.keySet())
        {
            Apttus_Config2.CustomClass.TaxResult taxResult =
taxResults.get(tId);

            // Tax Amount
            ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Tax Amount : ' +
taxResult.TaxAmount));

```

```
            // Tax BreakUps
            for(Apttus_Config2__TaxBreakup__c taxBreakUp :
taxResult.TaxBreakups)
            {

                TaxBreakUpWrapper temp = new TaxBreakUpWrapper();
                temp.TaxBreakUpId = taxBreakUp.Id;
                temp.TaxType = taxBreakUp.Apttus_Config2__TaxType__c;
                temp.TaxAppliedTo =
taxBreakUp.Apttus_Config2__TaxAppliesTo__c;
                temp.TaxRate = taxBreakUp.Apttus_Config2__TaxRate__c;
                temp.TaxAmount = taxBreakUp.Apttus_Config2__TaxAmount__c;
                temp.Sequence = taxBreakUp.Apttus_Config2__Sequence__c;

                lstTaxBreakUpWrapper.add(temp);
            }
        }
    }
    else
    {
        ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'Invalid or blank cart Id.'));
    }
}
```

Adding a Miscellaneous Item to the Cart

This API creates and assigns a miscellaneous item to the cart.

API	Signature
addMiscItem	*webService static Apttus_CPQApi.CPQ.AddMiscItemResponseDO addMiscItem(Apttus_CPQApi.CPQ.AddMiscItemRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.AddMiscItemRequestDO	This is the request data object.

Request Data Object - Apttus_CPQApi.CPQ.AddMiscItemRequestDO		
Field	**Type**	**Description**
AllocateGroupAdjustment	Boolean	Indicates whether group adjustment should be allocated.
AllowRemoval	Boolean	Indicates whether removing the miscellaneous item is allowed.
Amount	Decimal	The Base Price for the miscellaneous item.
CartId	Id	The id of the cart where you want to add the miscellaneous item.
ChargeType	String	The charge type of the miscellaneous item.
Description	String	Description of the miscellaneous item.

Response Data Object - Apttus_CPQApi.CPQ.AddMiscItemResponseDO		
Field	**Type**	**Description**
MiscItem	Apttus_Config2__LineItem__c	The miscellaneous item that is created.

Code Sample

The sample code below enables you to add a miscellaneous item to a cart by passing proposal, charge type, description, and base amount.

```
/**
 * The below method demonstrates how to add a misc to an existing cart (every quote has a cart)
 * Let's assume that the quote cart is already added Laptop as a bundle product and its two options are Keyboard and Mouse.
 * Inside this method, we will add delivery charge as misc item to the cart.
*/
public Apttus_CPQApi.CPQ.AddMiscItemResponseDO addMiscItem(String proposalName, String chargeType, String description, Decimal amount)
{
    Apttus_Config2__ProductConfiguration__c cart = [SELECT Id
                                                    FROM Apttus_Config2__ProductConfiguration__c
                                                    WHERE Apttus_QPConfig__Proposald__r.Name = :proposalName LIMIT 1];

    // Create the request object
    Apttus_CPQApi.CPQ.AddMiscItemRequestDO request = new Apttus_CPQApi.CPQ.AddMiscItemRequestDO();

    // add misc attributes to the request
    request.CartId = cart.Id;
    request.ChargeType = ChargeType;
    request.Description = Description;
    request.Amount = amount;
```

```
20      request.AllocateGroupAdjustment = true;
21      request.AllowRemoval = true;
22
23      // create the response object
24      Apttus_CPQApi.CPQ.AddMiscItemResponseDO response =
Apttus_CPQApi.CPQWebService.addMiscItem(request);
25
26      return response;
27  }
28
```

Creating Coupons for Incentives

You can use this API to create coupons for incentives.

> When you generate coupons using the API, if any duplicate coupon code is generated, the job fails and you need to execute the job again.

API	Signature
createCouponsForIncentives	*webService static Apttus_CPQApi.CPQ.CreateCouponsForIncentiveResponseDO createCouponsForIncentives(Apttus_CPQApi.CPQ.CreateCouponsForIncentiveRequestDO request)*

Request Parameter		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQ.CreateCouponsForIncentiveRequestDO	The request data object.

Request Data Object - Apttus_CPQApi.CPQ.CreateCouponsForIncentiveRequestDO		
Name	**Type**	**Description**
IncentiveIds	List	The IDs of the incentives for which you want to generate coupons.
NumCoupons	List	The number of coupons you want to generate. For example, if you enter 10000, CPQ will generate 10000 coupons for the specified incentive.

Reponse Parameter Data Object - Apttus_CPQApi.CPQ.CreateCouponsForIncentiveResponseDO		
Name	**Type**	**Description**
Coupons	List	The list of generated coupon codes.
IsAsync	Boolean	Indicates whether coupons were created in aysnc mode or not. • If there are <10k coupons, CPQ returns Async as False. • If there are >10k coupons, CPQ returns Async as True. • If there are multiple incentives, CPQ returns Async as True.
JobId	Id	The ID of the job created to generate coupons.

Code Sample

The sample code below demonstrates the coupons created in sync mode and async mode.

```
1  //Sync Test
2
```

```
Apttus_CPQApi.CPQ.CreateCouponsForIncentiveRequestDO request = new
 Apttus_CPQApi.CPQ.CreateCouponsForIncentiveRequestDO();
request.IncentiveIds.add(incentiveSO.Id);
// add corresponding number of coupons parameter
request.NumCoupons.add(5);

Apttus_CPQApi.CPQ.CreateCouponsForIncentiveResponseDO result =
Apttus_CPQApi.CPQWebService.createCouponsForIncentives(request);

//Async Test

Apttus_CPQApi.CPQ.CreateCouponsForIncentiveRequestDO request2 = new
 Apttus_CPQApi.CPQ.CreateCouponsForIncentiveRequestDO();
request2.IncentiveIds.add(incentiveSO.Id);
request2.IncentiveIds.add(incentiveSO2.Id);

// add corresponding number of coupons parameter
request2.NumCoupons.add(5);
request2.NumCoupons.add(5);

Apttus_CPQApi.CPQ.CreateCouponsForIncentiveResponseDO result2 =
Apttus_CPQApi.CPQWebService.createCouponsForIncentives(request2);
```

Retrieving Products Included by Auto-inclusion Constraint Rule

This processes and applies all the constraint rules.

> **Best Practice**
>
> It is recommended that you use the associateConstraintRules API before using this API. Directly using this API may not trigger the constraint rule.

API	Signature
applyConstraintRules	*webService static void applyConstraintRules(Id cartId, Boolean finalCheck)*

Parameters		
Name	**Type**	**Description**
cartId	ID	The id of the cart.
finalCheck	Boolean	If this is set to true, runs rules that are marked as check on finalization.

Code Sample

The sample below enables you to apply constraint rules to a cart with a specific cart ID. If you set the finalCheck flag as true, the constraint rules are run when you finalize the cart. If the flag is set as false, the rules are run before cart finalization. For example, if you want to add a Shipping Costs constraint rule only after the cart is finalized, set the finalCheck as true and the shipping costs constraint rule is applied once the user finalized the cart. If you want to apply constraint rules before the cart is finalized, set finalCheck as false. For example, if you set the finalCheck flag as false, installation charges are added along with a product selected before the cart is finalized.

```
public void applyConstraintRules()
{
    // For rules that are not marked as Check on Finalization
    Apttus_CPQApi.CPQWebService.applyConstraintRules(cartID, false);

    // For rules that are marked as Check on Finalization
    //Apttus_CPQApi.CPQWebService.applyConstraintRules(cartID, true);
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Associating Constraint Rules to a Cart

Response/Request XML

Example Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cpq="
http://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Header>
      <cpq:SessionHeader>
         <cpq:sessionId>00D4P000000z7dE!
AQQAQFudWwmQHKaOg8qt7T4KJ9MmTK0J055OXmfoUk9bUEL_idltBYg5muQuM4Pm0HVjinAgttLfi55uyxVSv
F5yrkoH.rH4</cpq:sessionId>
      </cpq:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <cpq:applyConstraintRules>
         <cpq:cartId>a1l4P00000Bg7CkQAJ</cpq:cartId>
         <cpq:finalCheck>false</cpq:finalCheck>
      </cpq:applyConstraintRules>
   </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="ht
tp://soap.sforce.com/schemas/class/Apttus_CPQApi/CPQWebService">
   <soapenv:Body>
      <applyConstraintRulesResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

Adding Line Items

You can use this API to add line items to a specified cart. The line items can be for bundles, standalone products, and options.

> This API does not create the associated Product Attribute Value record, you must call the **createNewAttributes** API right after executing this API.

API	Signature
addLineItems	*webService static Apttus_CPQApi.CPQ.AddMultiProductResponseDO addLineItems(Id cartId, List lineItems, Apttus_CPQApi.CPQ.AddLineItemsRequestDO optionalParams)*

Parameters		
Field	**Type**	**Description**
cartId	ID	The ID of the cart.
lineItems	List<Apttus_Config2__LineItem__c>	List of LineItems SObject records to be added to the Cart.
optionalParams	Apttus_CPQApi.CPQ.AddLineItemsRequestDO	Additional parameter that must not be used for this API.

Response Data Object - Apttus_CPQApi.CPQ.AddMultiProductResponseDO		
Field	**Type**	**Description**
LineNumbers	List<Decimal>	Contains List of Line Numbers of the newly added Line Items

Code Sample

The sample code below enables you to add a bundle and an option line item. You must provide values for the following mandatory fields:

- For bundle
  - ConfigurationId__c
  - ProductId__c
- For option
  - ConfigurationId__c
  - OptionId__c
  - ParentBundleNumber__c

You can set relevant values on any field when passing the Line Item record in the request. However, you should not set system fields here such as Line Number, Primary Line Number, and Pricing related fields. CPQ manages these fields internally.

```
// prepare bundle line for input
Product2 bundleProd = [SELECT Id FROM Product2 WHERE Name = 'Patient Accounting'];
Apttus_Config2__LineItem__c bundleLineItemSO = new LineItem__c(Apttus_Config2__ConfigurationId__c = getCartSO().Id);
bundleLineItemSO.Apttus_Config2__ProductId__c = bundleProd.Id;

// prepare option line for input
Product2 optionProd = [SELECT Id FROM Product2 WHERE Name = 'Dell 360'];
Apttus_Config2__LineItem__c optionLineItemSO = new Apttus_Config2__LineItem__c(Apttus_Config2__ConfigurationId__c = getCartSO().Id);
optionLineItemSO.Apttus_Config2__ProductId__c = bundleProd.Id;
optionLineItemSO.Apttus_Config2__OptionId__c = optionProd.Id;
optionLineItemSO.Apttus_Config2__ParentBundleNumber__c = 1;
```

```
12
13  // add bundle line item
14  List<Decimal> bundleLineNumbers = CPQWebService.addLineItems (
15  getCartSO().Id,
16  new List<Apttus_Config2__LineItem__c>{bundleLineItemSO});
17
18  // add option line item
19  List<Decimal> optionLineNumbers = Apttus_CPQApi.CPQWebService.addLineItems
    (
20  getCartSO().Id,
21  new List<Apttus_Config2__LineItem__c>{optionLineItemSO});
```

CPQ Web Service (Apttus_Config2)

The CPQ web service APIs account for the standard actions to configure, price, and quote.

You can invoke APIs in CPQ Web Service(Apttus_Config2) from the following command:

```
Apttus_Config2.CPQWebService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the CPQ web service APIs to complete the following tasks:

- Creating Product Line Items
- Registering Split Cart Parameters
- Finalizing the Configuration Version
- Finalizing the Configuration Version with Synchronization Tasks
- Abandoning a Configuration
- Cloning Primary Line Items
- Creating Bundle Line Items
- Creating and Updating Related Line Items
- Saving Related Line Items
- Splitting Related Line Items

Creating Product Line Items

This API creates product line items.

API	Signature
createProductLineItems	*webService static List createProductLineItems(Apttus_Config2__ProductConfiguration__c configSO, Apttus_Config2.CPQStruct.ProductCollDO prodCollDO)*

Request Parameter		
Name	**Type**	**Description**
configSO	Apttus_Config2__ProductConfiguration__c	The ID of product configuration sobject.
prodCollDO	Apttus_Config2.CPQStruct.ProductCollDO	The product details request data object

Request Data Object - Apttus_Config2.CPQStruct.ProductCollDO		
Name	**Type**	**Description**
Products	Apttus_Config2.CPQStruct.ProductLineItemDO	The bundle product request data object
HasOptionProducts	Boolean	Indicate whether has option products or not.
ProductIds	List<ID>	The list of product Ids

Data Object - Apttus_Config2.CPQStruct.ProductLineItemDO		
Name	**Type**	**Description**
AddedBy	String	Indicate how was the product added. For example, added by constraint rule, APIs.
Comments	String	The comments added in the product.
CustomData	Apttus_Config2__LineItem__c	The custom data based on list of custom fields to initialize Line Item record.
CustomFields	List<String>	List of Custom Fields.
EndDate	Date	The end date of the product.
productOptionComponentId	ID	The ID of the Product Option Component (only for option products).
ProductSO	Product2	The product sobject.
Quantity	Decimal	The quantity of the product.
SellingTerm	Decimal	The selling term of the product.
StartDate	Date	The start date of the product.

Response Parameters		
Field	**Type**	**Description**
productLineItems	List<Apttus_Config2__LineItem__c>	The list of IDs for the new product line items.

Code Sample

The sample code below enables you to create product line items for an existing cart. After the API is executed, the list of line items is returned without committing the changes into

Salesforce. These line items can be used for further modification and later associate to cart by performing insert operation.

```
/**
 * The below method demonstrates how to create bundle lineitems with option  (every quote has a cart)
 * Let’s assume the Quote's Cart is not blank and Laptop is a bundle product and its two options are Keyboard and Mouse
 * The input of this method is Quote Number and the bundle and related option product names
 * Inside the method we will create the line items for the Laptop bundle product and also its two options Keyboard and Mouse
 */
 public List<Apttus_Config2__LineItem__c> createProductLineItems(String proposalID, List<String> bundleProductName)
 {
    Apttus_Config2__ProductConfiguration__c cart = [SELECT Apttus_Config2__PriceListId__c,
                                                    Id,
Apttus_Config2__SummaryGroupType__c,
Apttus_Config2__ContractNumbers__c,
Apttus_Config2__EffectiveDate__c,
Apttus_Config2__PurchaseId__c,
Apttus_Config2__ExpectedStartDate__c,
Apttus_Config2__ExpectedEndDate__c,
Apttus_Config2__EffectivePriceListId__c,
Apttus_Config2__PriceListId__r.Apttus_Config2__CostModelId__c
                                                    FROM Apttus_Config2__ProductConfiguration__c
                                                    WHERE Apttus_QPConfig__Proposald__r.Name = :proposalID
                                                    LIMIT 1];

```

```
23      List<Apttus_Config2.CPQStruct.ProductLineItemDO> prodItems = new
 List<Apttus_Config2.CPQStruct.ProductLineItemDO>();
24      for(Product2 ProductSO : [SELECT ID,
25                                       Apttus_Config2__Uom__c,
26                                       Name,
27                                       Apttus_Config2__Customizable__c,
28                                       Apttus_Config2__HasDefaults__c,
29                                       Apttus_Config2__HasAttributes__c,
30                                       Apttus_Config2__HasOptions__c
31                              FROM Product2
32                              WHERE Name IN :bundleProductName])
33      {
34          prodItems.add(new
 Apttus_Config2.CPQStruct.ProductLineItemDO(ProductSO, 10, 1, null, null,
'Comment from API'));
35      }
36
37      Apttus_Config2.CPQStruct.ProductCollDO prodCollDO = new
 Apttus_Config2.CPQStruct.ProductCollDO(prodItems);
38      List<Apttus_Config2__LineItem__c> productLineItems =
Apttus_Config2.CPQWebService.createProductLineItems(cart,prodCollDO);
39
40      return productLineItems;
41  }
```

Registering Split Cart Parameters

The global method *setSplitCartInfo* is used to group the line items in the cart for pricing. You can define the parameters, namely the threshold and the split criteria based on which you want to split the line items in the cart through the API. If you do not define the parameters, the values of settings **Split Cart Threshold** and **Split Cart Criteria Fields** in Config System Properties are used to split the line items instead.

API	Signature
setSplitCartInfo	*static void setSplitCartInfo(Id cartId, Apttus_Config2.CustomClass.SplitCartInfo splitInfo)*

Parameters			
Name	**Type**	**Required?**	**Description**
cartId	ID	Yes	The id of the Cart you want to split.
splitInfo	Apttus_Config2.CustomClass.CustomClass.SplitCartInfo	Yes	The request object.

Request Object - Apttus_Config2.CustomClass.SplitCartInfo		
Field	**Type**	**Description**
SplitCriteriaFields	List	The list of API names of the fields to use as criteria.
SplitThreshold	Integer	The number of line items in a group.

Code Sample

The sample below enables you to set the values of **Split Cart Threshold** and **Split Cart Criteria Fields** to define the information required to split a cart.

```
1  //Setting Split info
2  Apttus_Config2.CustomClass.SplitCartInfo splitInfo = new
    Apttus_Config2.CustomClass.SplitCartInfo();
3  splitInfo.SplitCriteriaFields.add('Apttus_Config2_LocationId_c');
4  splitInfo.SplitThreshold = 10;
5  //Calling SplitInfo
6  // cartID can be Main Cart Id or Config Cart Id
7  ID cartID = 'a1I2f00000029Iy';
8  Apttus_Config2.CPQWebService.setSplitCartInfo(cartID , splitInfo);
```

Finalizing the Configuration Version

This API finalizes the configuration version and related line items. After the configuration is finalized, an email notification is sent when concurrent access enabled for the cart. The cart should be configured and priced before invoking this API. You can create a button to finalize the cart which the user can click to invoke this API once the products in the cart are adjusted and final.

API	Signature
finalizeConfiguration	*webService static void finalizeConfiguration(Id configId)*

Request Parameter		
Name	**Type**	**Description**
configId	ID	The ID of product configuration you want to finalize.

Code Sample

The sample code below enables you to finalize a cart with a specific configuration and synchronizes the products (line items) added to the cart with the quote or proposal used to generate the cart.

```
/**
 * Below method demonstrate to finalize the cart by passing the configuration request Id.
*/
public void finalizeCart(Id configId)
{
    // finalize the cart
    Apttus_Config2.CPQWebService.finalizeConfiguration(configId);
}
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

API Prerequisites

- Creating a Cart from a Quote
- Updating Price

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:cpq="http://soap.sforce.com/schemas/class/Apttus_Config2/CPQWebService">
    <soapenv:Header>
        <cpq:SessionHeader>
            <cpq:sessionId>00DZ000000NAEIA!
ASAAQN7k3xhhvMe.j.8gpR.ijcGq77HJeF4MEDfbxbBAqZ8r4WWNTv3OVb6o1bjtHJLbq5mvHcuAH_ie6sTC3
DzrWgnLdUuD</cpq:sessionId>
        </cpq:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <cpq:finalizeConfiguration>
            <cpq:configId>a1OZ0000002YrHD</cpq:configId>
        </cpq:finalizeConfiguration>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_Config2/CPQWebService">
    <soapenv:Body>
        <finalizeConfigurationResponse/>
```

```
    </soapenv:Body>
</soapenv:Envelope>
```

Finalizing the Configuration Version with Synchronization Tasks

This API finalizes the configuration version and finalizes the cart. This API also execute async Action Callback. Based on cart type (large cart/normal cart) the callback will be executed synchronously or asynchronously. The cart should be configured and priced before invoking this API. You can create a button to finalize the cart which the user can click to invoke this API once the products in the cart are adjusted and final.

API	Signature
finalizeConfigurationAndSync	*webService static void finalizeConfigurationAndSync(Id configId, Id requestId, String finalizeClassName)*

Request Parameter		
Name	**Type**	**Description**
configId	ID	The ID of product configuration you want to finalize.
requestId	ID	The ID of the request of the configuration you want to finalize.
finalizeClassName	String	The finalize Callback class name.

Code Sample

The sample code below enables you to finalize a cart with the configuration you provided. After the finalization product line items added to the cart are synchronized with the quote or proposal used to generate the cart. The action defined in the callback class you mentioned in the **finalizeClassName** parameter is also executed. Based on cart type (large cart/normal cart) the callback that implements *CustomClass.IActionCallback* and *CustomClass.IActionCallback2* interfaces are executed synchronously or asynchronously.

```
/**
 * Below method demonstrate to finalize the cart by passing the configuration request Id.
*/

public void finalizeConfigurationAndSync (Id configId, Id requestId, String finalizeClassName)
{
    // finalize the cart
    Apttus_Config2.CPQWebService.finalizeConfigurationAndSync(configId, requestId, finalizeClassName);
}
```

Abandoning a Configuration

This API abandons the product configuration using the ID. The record is kept or deleted based on the configuration in the **Keep Abandoned Carts** field in Config System Properties. If **Keep Abandoned Carts** is enabled, the API changes the status of the product configuration to *Abandoned*, removes unreferenced totaling groups, and writes activity logs by creating a task. The API cancels the Quote Collaboration requests along with abandoning the cart. The API also executes any callbacks that must be executed before deleting a line item.

API	Signature
abandonConfiguration	*webService static void abandonConfiguration(Id configId)*

Request Parameter		
Name	**Type**	**Description**
configId	ID	The ID of product configuration you want to abandon.

Code Sample

The below sample code allows you to delete the configuration record and the related line item associated with the configuration. In this case, the **Keep Abandoned Carts** field is disabled.

```
/**
 * The below method demonstrates how to delete product configuration.
*/
public void abondonConfigurationID (String proposalName)
{
    Apttus_Config2__ProductConfiguration__c productConfig = [SELECT Id
                                                             FROM
Apttus_Config2__ProductConfiguration__c
                                                             WHERE
Apttus_QPConfig__Proposald__r.Name = :proposalName
                                                             LIMIT 1];
    Apttus_Config2.CPQWebService.abandonConfiguration (productConfig.Id);
}
```

Cloning Primary Line Items

This global method clones line items from one configuration to another using Line Number.

API	Signature
clonePrimaryItemColl	*static Id clonePrimaryItemColl(Id configId, Integer lineNumber, Id targetConfigId)*

Request Parameter		
Name	**Type**	**Description**
configId	Id	The ID of source product configuration associated with the line items.

Request Parameter		
Name	**Type**	**Description**
lineNumber	Integer	The Line Number of the line items you want to clone.
targetConfigId	Id	The ID of the product configuration you want to clone the line items into.

Response Parameters		
Field	**Type**	**Description**
clonedPrimaryLineNumber	Id	The Line number of the cloned primary line item collection

Code Sample

The sample code below enables you to clone line items associated with a Line Number and add them to the target configuration. This returns the line number of the cloned line items.

```
1  /**
2    * Clones the given primary line item collection and adds it onto the
   target configuration
3    * @param parentConfigId the id of the product configuration associated
   with the line items.
4    * @param primaryLineNumber the line number associated with the item
   collection
5    * @param childConfigId the id of child product configuration
6    * @return the Id of the new primary line item collection
7    */
8
9   public Id clonePrimaryItemCollection(Id parentConfigId, Integer
   primaryLineNumber, Id childConfigId)
10     {
11       Id clonedPrimaryLineNumber;
```

```
12      clonedPrimaryLineNumber =
Apttus_Config2.CPQWebService.clonePrimaryItemColl(parentConfigId,
primaryLineNumber, childConfigId);
13      return clonedPrimaryLineNumber;
14    }
```

Creating Bundle Line Items

This API creates the bundle and option line items based on the bundle product data object.

API	Signature
createBundleLineItems	*webService static List createBundleLineItems(Apttus_Config2__ProductConfiguration__c configSO, Apttus_Config2.CPQStruct.ProductBundleDO bundleDO)*

Request Parameter		
Name	**Type**	**Description**
configSO	Apttus_Config2__ProductConfiguration__c	The ID of product configuration sobject.
bundleDO	Apttus_Config2.CPQStruct.ProductBundleDO	The bundle details request data object

Request Data Object - Apttus_Config2.CPQStruct.ProductBundleDO		
Name	**Type**	**Description**
BundleProduct	Apttus_Config2.CPQStruct.ProductLineItemDO	The bundle product request data object
HasOptionProducts	Boolean	Indicate whether has option products or not.

Request Data Object - Apttus_Config2.CPQStruct.ProductBundleDO		
Name	**Type**	**Description**
OptionProductIds	List<ID>	The list of option product Ids
OptionProducts	ListList<ProductLineItemDO>	The list of option product data objects.
Data Object - Apttus_Config2.CPQStruct.ProductLineItemDO		
Name	**Type**	**Description**
AddedBy	String	Indicate how was the product added. For example, added by constraint rule, APIs.
Comments	String	The comments added in the product.
CustomData	Apttus_Config2__LineItem__c	The custom data based on list of custom fields to initialize Line Item record.
CustomFields	List<String>	List of Custom Fields.
EndDate	Date	The end date of the product.
productOptionComponentId	ID	The ID of the Product Option Component (only for option products).
ProductSO	Product2	The product sobject.
Quantity	Decimal	The quantity of the product.
SellingTerm	Decimal	The selling term of the product.
StartDate	Date	The start date of the product.

Response Parameters		
Field	**Type**	**Description**
lineItems	List<Apttus_Config2__LineItem__c>	The list of IDs for the new bundle line items.

Code Sample

The sample code below enables you to create bundle and option line items for an existing cart. This API returns the list of line items without committing changes into Salesforce. These line items can be used for further modification and later associate to cart by performing insert operation.

```
/**
 * The below method demonstrates how to create bundle lineitems with option  (every quote has a cart)
 * Let's assume the Quote's Cart is not blank and Laptop is a bundle product and its two options are Keyboard and Mouse
 * The input of this method is Quote Number and the bundle and related option product names
 * Inside the method we will return the line items for the Laptop bundle product and also its two options Keyboard and Mouse
 */

public List<Apttus_Config2__LineItem__c> createBundleLineItems(String proposalID, String bundleProductName, List<String> optionProductNames)
{

    Apttus_Config2__ProductConfiguration__c cart = [SELECT Apttus_Config2__PriceListId__c,
                                                           Id,

Apttus_Config2__SummaryGroupType__c,

Apttus_Config2__ContractNumbers__c
                                                    FROM
Apttus_Config2__ProductConfiguration__c
                                                    WHERE
Apttus_QPConfig__Proposald__r.Name = :proposalID
```

```
                                                    LIMIT 1];
    //Add bundle product data object
    Product2 bundleProductSO = [SELECT ID,
                                       Apttus_Config2__Uom__c,
                                       Name,
Apttus_Config2__Customizable__c,
                                       Apttus_Config2__HasDefaults__c,
                                       Apttus_Config2__HasAttributes__c,
                                       Apttus_Config2__HasOptions__c
                                FROM Product2
                                WHERE Name = :bundleProductName
                                LIMIT 1];
    Apttus_Config2.CPQStruct.ProductLineItemDO BundleProduct = new
 Apttus_Config2.CPQStruct.ProductLineItemDO( bundleProductSO,
                                                                    10,   //
quantity assigned to the product
                                                                    1,    //
selling term for the product
                                                                    null, //
Start date
                                                                    null, //
End date
                                                                    'Comment
from API');
    List<Apttus_Config2.CPQStruct.ProductLineItemDO> optionProducts = new
 List<Apttus_Config2.CPQStruct.ProductLineItemDO>();
    for (Product2 optionProductSO : [SELECT ID,
                                     Apttus_Config2__Uom__c,
                                     Name,
                                     Apttus_Config2__Customizable__c,
                                     Apttus_Config2__HasDefaults__c,
                                     Apttus_Config2__HasAttributes__c,
                                     Apttus_Config2__HasOptions__c
                                    FROM Product2
                                    WHERE Name IN :optionProductNames])
    {
        optionProducts.add(new
 Apttus_Config2.CPQStruct.ProductLineItemDO(optionProductSO,5,1, null,null,'
Comment from API'));
    }
    Apttus_Config2.CPQStruct.ProductBundleDO bundleDO = new
 Apttus_Config2.CPQStruct.ProductBundleDO(BundleProduct, optionProducts);
```

```
48      List<Apttus_Config2__LineItem__c> lineItems =
Apttus_Config2.CPQWebService.createBundleLineItems(cart,bundleDO);
49      return lineItems;
50  }
```

Creating and Updating Related Line Items

You can use this global method to create related line items records of an asset and associate them with a given service product. You can also delete the related line item. You can create and delete related line items for multiple service products in a single call, however, the number of line items must adhere to the Salesforce Governor Limits.

API	Signature
updateRelatedLineItems	static Apttus_Config2.CPQStruct.UpdateRelatedLineItemResponseDO updateRelatedLineItems(Apttus_Config2.CPQStruct.UpdateRelatedLineItemRequestDO requestDO)

Parameters

Name	Type	Description
requestDO	Apttus_Config2.CPQStruct.UpdateRelatedLineItemRequestDO	Request related line item details data object that is invoked by the method.

Request Data Object - Apttus_Config2.CPQStruct.UpdateRelatedLineItemResponseDO

Name	Type	Description
CartId	ID	The ID of the product configuration.
RelatedLineItemColls	Map<Id, List<Apttus_Config2.RelatedLineItemColl>>	Map of service line item id to the list of data objects containing relevant details of the asset associated with the given service line item.

Data Object - Apttus_Config2.RelatedLineItemColl		
Name	**Type**	**Description**
Action	String	Define one of the following strings to specify whether associate or dissociate the asset from the service line item: • *add* • *remove*
AssetLineItemId	Id	The Id of the asset line item
RelatedLineItemSO	Apttus_Config2__RelatedLineItem__c	The Sobject with the updated values to be saved in the database.

Response Data Object - Apttus_Config2.CPQStruct.UpdateRelatedLineItemResponseDO		
Field	**Type**	**Description**
Errors	List <Strings>	List of errors encountered while creating or deleting the related line item
isSuccess	Boolean	Indicates whether the related line items were created or deleted successfully.

Code Sample

The code sample below helps you create and delete a related line item.

```
public void addRemoveRelatedLineItems(String lineItemId, String assetId1, String
assetId2, String assetId3, String cartId)
```

```
{

    // Use LineAction.Add to create new RLI to associate an Asset with Service Line
Item
    Apttus_Config2.CPQStruct.RelatedLineItemColl relatedLineItemDO1 = new
 Apttus_Config2.CPQStruct.RelatedLineItemColl();
    relatedLineItemDO1.AssetLineItemId = assetId1;
    relatedLineItemDO1.Action = 'Add';

    Apttus_Config2.CPQStruct.RelatedLineItemColl relatedLineItemDO2 = new
 Apttus_Config2.CPQStruct.RelatedLineItemColl();
    relatedLineItemDO2.AssetLineItemId = assetId2;
    relatedLineItemDO2.Action = 'Add';

    // Use LineAction.Remove to delete existing RLI for dissociating an Asset from
Service Line
    Apttus_Config2.CPQStruct.RelatedLineItemColl relatedLineItemDO3 = new
 Apttus_Config2.CPQStruct.RelatedLineItemColl();
    relatedLineItemDO3.AssetLineItemId = assetId3;
    relatedLineItemDO3.Action = 'Delete';

    // Prepare the  request structure to be passed to the API
    Apttus_Config2.CPQStruct.UpdateRelatedLineItemRequestDO requestDO = new
 Apttus_Config2.CPQStruct.UpdateRelatedLineItemRequestDO();
    requestDO.RelatedLineItemColls.put(lineItemId, new
 List<Apttus_Config2.CPQStruct.RelatedLineItemColl>{relatedLineItemDO1,
relatedLineItemDO2, relatedLineItemDO3});
    requestDO.CartId = cartId;

    // Invoke the API by passing the request formed above
    Apttus_Config2.CPQStruct.UpdateRelatedLineItemResponseDO responseDO =
Apttus_Config2.CPQWebService.updateRelatedLineItems(requestDO);

    // Check the response for errors if any
    if (!responseDO.IsSuccess) {
        System.debug('Errors thrown :');
        if (responseDO.Errors != null && responseDO.Errors.size() > 0) {
            for (String strError : responseDO.Errors) {
                System.debug('\nError - ' + strError);
            }
        }
    }
}
```

Saving Related Line Items

You can use this global method to save modified related line items. You can modify the following fields in the related line items:

- **Weightage Type**
- **Weightage Percent**
- **Weightage Amount**
- Any custom fields

API	Signature
saveRelatedLineItems	*static Apttus_Config2.CPQStruct.SaveRelatedLineItemResponseDO saveRelatedLineItems(Apttus_Config2.CPQStruct.SaveRelatedLineItemRequestDO requestDO)*

Parameters

Name	Type	Description
requestDO	Apttus_Config2.CPQStruct.SaveRelatedLineItemRequestDO	The related line item request data object.

Request Data Object - Apttus_Config2.CPQStruct.SaveRelatedLineItemRequestDO

Name	Type	Description
CartId	ID	The ID of the product configuration where you want to save the modified related line item.
RelatedLineItem	List<Apttus_Config2.RelatedLineItemColl>	List of related line items to be saved in the database.

Data Object - Apttus_Config2.RelatedLineItemColl		
Name	**Type**	**Description**
Action	String	Define one of the following strings to specify whether associate or dissociate the asset from the service line item: • *add* • *remove*
AssetLineItemId	Id	The Id of the asset line item
RelatedLineItemSO	Apttus_Config2__RelatedLineItem__c	The Sobject with the updated values of related line items to be saved in the database.

Response Data Object - Apttus_Config2.CPQStruct.*S*aveRelatedLineItemResponseDO		
Field	**Type**	**Description**
Errors	List <Strings>	List of errors encountered while saving the related line item
isSuccess	Boolean	Indicates whether the related line items were saved successfully.

Code Sample

The code sample below helps you save related line items.

```
public void saveRelatedLineItems(String cartId, String lineItemId) {

        // Prepare the request structure to be passed to the API
```

```
        Apttus_Config2.CPQStruct.SaveRelatedLineItemRequestDO requestDO = new
 Apttus_Config2.CPQStruct.SaveRelatedLineItemRequestDO();
        requestDO.CartId = cartId;

        List<Apttus_Config2__RelatedLineItem__c> relatedLineItemSOs = [SELECT
Apttus_Config2__LineItemId__c, Apttus_Config2__WeightageType__c,
Apttus_Config2__WeightageAmount__c FROM Apttus_Config2__RelatedLineItem__c WHERE
Apttus_Config2__LineItemId__c =: lineItemId];

        // below is simple example logic to set the Weightage Amount and setting
Weightage type to Amount
        // Customers can have their use cases implemented here
        for (Apttus_Config2__RelatedLineItem__c relatedLineItemSO :
relatedLineItemSOs) {
            relatedLineItemSO.Apttus_Config2__WeightageType__c = 'Amount';
            relatedLineItemSO.Apttus_Config2__WeightageAmount__c = 35;
            Apttus_Config2.CPQStruct.RelatedLineItemColl relatedLineItemDO = new
 Apttus_Config2.CPQStruct.RelatedLineItemColl();
            relatedLineItemDO.RelatedLineItemSO = relatedLineItemSO;

            requestDO.RelatedLineItems.add(relatedLineItemDO);
        }

        // Invoke the API by passing the request formed above
        Apttus_Config2.CPQStruct.SaveRelatedLineItemResponseDO responseDO =
Apttus_Config2.CPQWebService.saveRelatedLineItems(requestDO);

        // Check the response for errors if any
        if (!responseDO.IsSuccess) {
            System.debug('Errors thrown :');
            if (responseDO.Errors != null && responseDO.Errors.size() > 0) {
                for (String strError : responseDO.Errors) {
                    System.debug('\nError - ' + strError);
                }
            }
        }
}
```

Splitting Related Line Items

You can use this global method to clone a service line item for each related line item associated with that service line item.

API	Signature
splitRelatedLineItems	*static Apttus_Config2.CPQStruct.SplitRelatedLineItemResponseDO splitRelatedLineItems(Apttus_Config2.CPQStruct.SplitRelatedLineItemRequestDO requestDO)*

Parameters

Name	Type	Description
requestDO	Apttus_Config2.CPQStruct.SplitRelatedLineItemRequestDO	The request data object of related line item details invoked by the method.

Request Data Object - Apttus_Config2.CPQStruct.SplitRelatedLineItemRequestDO

Name	Type	Description
CartId	ID	The ID of the product configuration record.
Flow	String	The flow you want to use to validate the split criteria fields.
LineNumber	Integer	The **Line Number** of the service line item you want to split.
SplitCriteriaFields	List<String>	List of some or all fields defined in **Split Criteria Fields** in Config System Properties to be used as criteria to split the service line item.

Response Data Object - Apttus_Config2.CPQStruct.SplitRelatedLineItemResponseDO

Field	Type	Description
Errors	List <Strings>	List of errors encountered while splitting the service line item
isSuccess	Boolean	Indicates whether the splitting of service line item was successful.

Code Sample

The code sample below helps you split a service line item.

```
public void splitServiceCart(String cartId, String splitCriteriaFieldName)
{

    // Prepare the request structure to be passed to the API
    Apttus_Config2.CPQStruct.SplitRelatedLineItemRequestDO request = new
 Apttus_Config2.CPQStruct.SplitRelatedLineItemRequestDO();
    request.CartId = cartId;

    // Service Line number to split
    request.LineNumber = 1;

    // this can take OOTB CPQ fields
    request.SplitCriteriaFields.add(splitCriteriaFieldName);

    // flow to validate the split criteria fields
    request.Flow = 'QuotingCartGrid';

    // Invoke the API by passing the request formed above
    Apttus_Config2.CPQStruct.SplitRelatedLineItemResponseDO responseDO =
Apttus_Config2.CPQWebService.splitRelatedLineItems(request);

    // Check the response for errors if any
    if (!responseDO.IsSuccess) {
        System.debug('Errors thrown :');
        if (responseDO.Errors != null && responseDO.Errors.size() > 0) {
            for (String strError : responseDO.Errors) {
                System.debug('\nError - ' + strError);
            }
        }
    }
}
```

CPQ Admin Web Service

The CPQ Admin web service APIs account for the standard actions to configuring products.

You can invoke APIs in CPQ Admin Web Service from the following command:

```
Apttus_Config2.CPQAdminWebService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the CPQ Admin web service APIs to complete the following tasks:

- Creating New Products
- Creating New Categories
- Associating Products to Category
- Building Hierarchy of Products
- Checking if Product Exists in Hierarchy
- Removing Products from Hierarchy
- Retrieving the List Option Group
- Retrieving Options in an Option Group
- Retrieving of Product Hierarchy
- Retrieves List of Parent Product
- Retrieves List Product IDs
- Retrieves List of Category IDs
- Creating New Attribute Group
- Adding and Removing Attributes from an Attribute Group
- Associating Attribute Group and Products
- Dissociating Attribute Group and Products

Creating New Products

You can use this API to create products.

API	Signature
createProducts	*webService static Apttus_Config2.CPQAdminStruct.ProductResponseDO createProducts(Apttus_Config2.CPQAdminStruct.ProductRequestDO productRequestDO)*

Request Parameter		
Name	**Type**	**Description**
productRequestDO	Apttus_Config2.CPQAdminStruct.ProductRequestDO	The create products request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.ProductRequestDO		
Name	**Type**	**Description**
ProductDOs	List<Apttus_Config2.CPQAdminStruct.ProductDO>	The list of product data objects.
Data Object - Apttus_Config2.CPQAdminStruct.ProductDO		
Name	**Type**	**Description**
Name	String	The name of the product
ConfigType	String	Type of configuration of the product. You can use the following values: • Standalone • Bundle • Option
IsActive	Boolean	Indicates whether the product should be active or inactive.
OptionalFields	List<Apttus_Config2.CPQAdminStruct. MapDO>	The list of optional field values. You can use this parameter to associate additional field values with the product.
CategoryId	Id	The Id of the category to which the products must be associated to.

Response Parameter - Apttus_Config2.CPQAdminStruct.ProductResponseDO		
Field	**Type**	**Description**
ProductStructureJSON	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of key-value pairs. The key refers to the product name that was passed in the request and the value refers to the Id of the product.

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Key	The name of the product.
String	Value	The value refers to the Id of the product.

Code Sample

The sample code below enables you to create products

```
/**
 * The below code demonstrates how to create one standalone and one bundle products for the given category Id
 */

Public void createProducts()
{

    Apttus_Config2.CPQAdminStruct.ProductRequestDO request = new Apttus_Config2.CPQAdminStruct.ProductRequestDO();
    Apttus_Config2.CPQAdminStruct.ProductDO productDO = new Apttus_Config2.CPQAdminStruct.ProductDO();
    List<Apttus_Config2.CPQAdminStruct.MapDO> listOfMapDO = new List<Apttus_Config2.CPQAdminStruct.MapDO>();
    Apttus_Config2.CPQAdminStruct.MapDO m = new Apttus_Config2.CPQAdminStruct.MapDO();

    // ADD 1ST PRODUCT
    // required fields
```

```
    productDO.Name = Standalone Product Name';
    productDO.ConfigType = 'Standalone';
    productDO.IsActive = true;

    // additional fields (Please use namespace for the fields outside your
ORG)
    // 1st additional field
    m.Key = ' Description';
    m.Value = 'Standalone product description';
    listOfMapDO.add(m);

    // 2nd additional field
    m = new Apttus_Config2.CPQAdminStruct.MapDO();
    m.Key =  ' Apttus_Config2__HasOptions__c';
    m.Value = 'false';
    listOfMapDO.add(m);

    // 3rd additional field
    m = new Apttus_Config2.CPQAdminStruct.MapDO();
    m.Key = ' Apttus_Config2__EffectiveDate__c';
    m.Value = '2016-01-13 02:28:00';
    listOfMapDO.add(m);
    // add additional fields to the optionalFields list
    productDO.OptionalFields.addAll(listOfMapDO);

    // ADD 2ND PRODUCT
    request.ProductDOs.add(productDO);
    productDO = new Apttus_Config2.CPQAdminStruct.ProductDO();

    // required fields
    productDO.Name = ‘Bundle Product Name';
    productDO.ConfigType = 'Bundle';
    productDO.IsActive = true;

    // additional fields
    m = new Apttus_Config2.CPQAdminStruct.MapDO();

    // 1st additional field
    m.Key = 'Description';
    m.Value = 'Bundle product description';
    listOfMapDO.add(m);

    // 2nd additional field
```

```
57      m = new Apttus_Config2.CPQAdminStruct.MapDO();
58      m.Key = 'Apttus_Config2__HasOptions__c';
59      m.Value = 'true';
60      listOfMapDO.add(m);
61
62      // 3rd additional field
63      m = new Apttus_Config2.CPQAdminStruct.MapDO();
64      m.Key = 'Apttus_Config2__EffectiveDate__c';
65      m.Value = '2021-01-01 15:24:00';
66      listOfMapDO.add(m);
67
68      // add optional fields to the optionalFields list
69      productDO.OptionalFields.addAll(listOfMapDO);
70      productDO.categoryID = 'a0i6C000001tnac'; // Id of any existing
    category
71
72      // add product to request
73      request.ProductDOs.add(productDO);
74      Apttus_Config2.CPQAdminStruct.ProductResponseDO response =
    Apttus_Config2.CPQAdminWebService.createProducts(request);
75  }
```

Creating New Categories

You can use this API to create new categories with the minimum required parameters like name, type, and label.

API	Signature
createCategories()	*webService static Apttus_Config2.CPQAdminStruct.CategoryResponseDO createCategories(Apttus_Config2.CPQAdminStruct.CategoryRequestDO categoryRequestDO)*

Request Parameter		
Name	**Type**	**Description**
categoryRequestDO	Apttus_Config2.CPQAdminStruct.CategoryRequestDO	The category request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.CategoryRequestDO		
Name	**Type**	**Description**
CategoryDOs	List<Apttus_Config2.CPQAdminStruct.CategoryDO>	The list of category data objects.

Response Parameter - Apttus_Config2.CPQAdminStruct.CategoryResponseDO		
Field	**Type**	**Description**
CategoryDOs	List<Apttus_Config2.CPQAdminStruct.CategoryDO>	The list of category data objects.
CategoryResponseMap	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of key-value pairs. The key refers to the category name that was passed in the request and the value refers to the Id of the category.

Data Object - Apttus_Config2.CPQAdminStruct.CategoryDO		
Name	**Type**	**Description**
Label	String	The label of the category
Name	String	The name of the category
OptionalFields	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of optional field values. You can use this parameter to associate additional field values with the category.
Type	String	Type of the category. You can use the following values: • Offering • Option Group • Both

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Key	The name of the category.
String	Value	The value refers to the Id of the category.

Code Sample

The below sample enables you to create categories. You can also add optional fields, add the namespace to the field names.

```
/**
 * The below code create categories of type Option Group and Offering
 */

Public void createCategories()
{
    Apttus_Config2.CPQAdminStruct.CategoryResponseDO categsResponse = null;
    Apttus_Config2.CPQAdminStruct.CategoryRequestDO categsRequest = new
 Apttus_Config2.CPQAdminStruct.CategoryRequestDO();
    Apttus_Config2.CPQAdminStruct.CategoryDO categDO = null;
    List<Apttus_Config2.CPQAdminStruct.MapDO> mapDOList = null;
    Apttus_Config2.CPQAdminStruct.MapDO mapDO = null;

    // CREATE 3 CATEGORIES:
    // CREATING 1ST CATEGORY
    // necessary fields
    categDO = new Apttus_Config2.CPQAdminStruct.CategoryDO();
    categDO.Name = 'CategoryOptionGroup1';
    categDO.Label = 'CategoryOptionGroup1';
    categDO.Type = 'Option Group';

    mapDOList = new List<Apttus_Config2.CPQAdminStruct.MapDO>();

    // additional fields
    // 1st
    mapDO = new Apttus_Config2.CPQAdminStruct.MapDO();
    mapDO.Key = 'Apttus_Config2__GuidePage__c';
```

```
27      mapDO.Value = 'a page';
28      mapDOList.add(mapDO);
29
30      // 2nd
31      mapDO = new Apttus_Config2.CPQAdminStruct.MapDO();
32      mapDO.Key = 'Apttus_Config2__Active__c';
33      mapDO.Value = 'true';
34      mapDOList.add(mapDO);
35
36      // add categDO to categsRequest
37      categDO.OptionalFields = mapDOList;
38      categsRequest.CategoryDOs.add(categDO);
39
40
41      // CREATING 2ND CATEGORY
42      // necessary fields
43      categDO = new Apttus_Config2.CPQAdminStruct.CategoryDO();
44      categDO.Name = 'CategoryOffering1';
45      categDO.Label = 'CategoryOffering1';
46      categDO.Type = 'Offering';
47
48      // add categDO to categsRequest
49      categsRequest.CategoryDOs.add(categDO);
50
51
52      // CREATING 3RD CATEGORY
53      // necessary fields
54      categDO = new Apttus_Config2.CPQAdminStruct.CategoryDO();
55      categDO.Name = 'CategoryOptionGroup2';
56      categDO.Label = 'CategoryOptionGroup2';
57      categDO.Type = 'Option Group';
58
59      // add categDO to categsRequest
60      categsRequest.CategoryDOs.add(categDO);
61
62      // create categories
63      categsResponse =
Apttus_Config2.CPQAdminWebService.createCategories(categsRequest);
64  }
```

Associating Products to Category

You can use this API to associate a product to a category.

API	Signature
associateProductToCategory	*webService static Boolean associateProductToCategory(Apttus_Config2.CPQAdminStruct.ProductRequestDO productsRequestDO)*

Request Parameter

Name	Type	Description
productsRequestDO	Apttus_Config2.CPQAdminStruct.ProductRequestDO	The category request data object.

Data Object - Apttus_Config2.CPQAdminStruct.ProductDO

Name	Type	Description
ProductId	Id	The ID of the product you want to associate with the category
CategoryId	Id	The Id of the category to which the products must be associated to.

Response Parameter

Name	Type	Description
IsSuccessful	Boolean	Indicates whether the association of product to the category was successful or not.

Code Sample

The below sample code enables you to associate the product to a specific category.

```
/**
 * The below code associate the products with related category. Invoke
below method by passing map of productid and categoryId as parameter.
 */

Public Boolean associateProductToCategory(Map<Id, Id>
mapProductIdCategoryId)
{
    Apttus_Config2.CPQAdminStruct.ProductRequestDO prodsAssocRequest = new
 Apttus_Config2.CPQAdminStruct.ProductRequestDO();
    CPQAdminStruct.ProductDO prodForAssociation = null;
    For (Id productId : mapProductIdCategoryId. keySet()) {
        prodForAssociation.productID = prodIDs[i];
        prodForAssociation.categoryID = categIDs[i];
        prodsAssocRequest.ProductDOs.add(prodForAssociation);
    }

    return

Apttus_Config2.CPQAdminWebService.associateProductToCategory(prodsAssocReq
uest);
}
```

Building Hierarchy of Products

You can use this API to build a hierarchy of products based on a list of parent and child pairs.

API	Signature
buildHierarchy	*webService static Apttus_Config2.CPQAdminStruct.HierarchyResponseDO buildHierarchy(Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierarchyRequestDO)*

Request Parameter		
Name	**Type**	**Description**
hierarchyRequestDO	Apttus_Config2.CPQAdminStruct.HierarchyRequestDO	The hierarchy request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.HierarchyRequestDO		
Name	**Type**	**Description**
HierarchyDOs	List<Apttus_Config2.CPQAdminStruct.HierarchyDO>	the list of hierarchy data object
isCascadeGroupChanges	Boolean	Indicates whether to reflect the changes made to this instance of the option group to all of its sibling option groups across products.
OptionGroupId	Id	The Id of the option group.
OptionGroupIds	List <Id>	The list of option group Ids to be associated with the bundle product.
ProductId	Id	The Id of the product to be the parent in the hierarchy.
ProductIds	List <Id>	The list of child product Ids.

Data Object - Apttus_Config2.CPQAdminStruct.HierarchyDO		
Name	**Type**	**Description**
ChildOptionGroupId	Id	The Id of the child option child group.
ChildProductId	Id	The Id of the child product.
ParentOptionGroupId	Id	The Id of the parent option group.

Data Object - Apttus_Config2.CPQAdminStruct.HierarchyDO		
Name	**Type**	**Description**
ParentProductId	Id	The Id of the parent product.

Response Parameter - Apttus_Config2.CPQAdminStruct.HierarchyResponseDO		
Field	**Type**	**Description**
ProductOptionGroupDO	Apttus_Config2.CPQAdminStruct2.ProductOptionGroupDO	The parent option group data object.
Success	Boolean	Indicates whether building hierarchy was successful or not.

Data Object - Apttus_Config2.CPQAdminStruct2.ProductOptionGroupDO		
Field	**Type**	**Description**
ChildOptionGroupDOs	List<Apttus_Config2.CPQAdminStruct2.ProductOptionGroupDO>	The list of child option group data object.
ChildProductOptionGroupSOs	List<Apttus_Config2__ProductOptionGroup__c>	The list of child product option groups.
ComponentSOs	List<Apttus_Config2__ProductOptionComponent__c>	The list of product option components.

Data Object - Apttus_Config2.CPQAdminStruct2.ProductOptionGroupDO		
Field	**Type**	**Description**
ProductOptionComponentDOs	List<Apttus_Config2.CPQAdminStruct2.ProductOptionComponentDO>	The list of product option components data objects.
ProductOptionGroupSO	Apttus_Config2__ProductOptionGroup__c	The sObject of the product option group.

Data Object - Apttus_Config2.CPQAdminStruct2.ProductOptionComponentDO		
Field	**Type**	**Description**
DefaultQuantityExpressionSO	Apttus_Config2__FieldExpression__c	The sObject of default quantity expression.
MaxQuantityExpressionSO	Apttus_Config2__FieldExpression__c	The sObject of max quantity expression
MinQuantityExpressionSO	Apttus_Config2__FieldExpression__c	The sObject of the product option group.
ProductOptionComponentSO	Apttus_Config2__ProductOptionComponent__c	The sObject of the product option component.

Code Sample

The below sample code demonstrates how to build a hierarchy on parent and child products.

```
/**
 * The below method accepts parent productname and list of option child products and categoryhierarchy name as input and return true or false as status of the build.
*/
public Boolean buildHierarchy(String parentProductName, List<String> optionProducts, String categoryHierarchyName)
{
    Apttus_Config2.CPQAdminStruct.HierarchyDO hierDO = new Apttus_Config2.CPQAdminStruct.HierarchyDO();
    Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierRequest = new Apttus_Config2.CPQAdminStruct.HierarchyRequestDO();
    Apttus_Config2.CPQAdminStruct.HierarchyResponseDO hierResponse;
    List< Apttus_Config2__ProductOptionComponent__c> poc;

    // STEP 1 - Get bundle product and options
    Product2 bundleProduct = [SELECT Id, Name
                              FROM Product2
                              WHERE Name = :parentProductName
                              LIMIT 1];
    Product2 options = [SELECT Id, Name
                        FROM Product2
                        WHERE Name IN : optionProducts];

    // STEP 2 - Get an option group to create a bundle with the 3 options from above
    Apttus_Config2__ ClassificationHierarchy__c optionGroup = [SELECT Id, Name
                                                               FROM Apttus_Config2__ ClassificationHierarchy__c
                                                               WHERE Name = :categoryHierarchyName
                                                               LIMIT 1];

    // STEP 3 - Create Bundle by associating Option Group and Product
    hierRequest.ProductId = bundleProduct.Id;
    hierRequest.OptionGroupIds.add(optionGroup.Id);

    // send request and save response
```

```
31      hierResponse =
Apttus_Config2.CPQAdminWebService.buildHierarchy(hierRequest);
32
33      // STEP 4 - Add options to the newly created bundle
34      hierRequest = new Apttus_Config2.CPQAdminStruct.HierarchyRequestDO();
35      hierRequest.ProductId = bundleProduct.Id;
36      hierRequest.OptionGroupId = pog[0].Id;
37      for (Product2 option : options) {
38          hierRequest.ProductIds.add(option.Id);
39      }
40      hierResponse =
Apttus_Config2.CPQAdminWebService.buildHierarchy(hierRequest);
41
42      return hierResponse.Success;
43  }
```

Checking if Product Exists in Hierarchy

You can use this API to check if a product exist in a category.

API	Signature
productExistsInHierarchy	*webService static Apttus_Config2.CPQAdminStruct.HierarchyResponseDO productExistsInHierarchy(Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierarchyRequestDO)*

Request Parameter		
Name	**Type**	**Description**
hierarchyRequestDO	Apttus_Config2.CPQAdminStruct.HierarchyRequestDO	The category request data object.

Request Parameter - Apttus_Config2.CPQAdminStruct.HierarchyRequestDO		
Name	**Type**	**Description**
HierarchyDOs	List<Apttus_Config2.CPQAdminStruct.HierarchyDO>	The list of hierarchy data object

Data Object- Apttus_Config2.CPQAdminStruct.HierarchyDO		
Name	**Type**	**Description**
ChildProductId	Id	The Id of child product.
ParentProductId	Id	The Id of the parent product.

Response Data Object- Apttus_Config2.CPQAdminStruct.HierarchyResponseDO		
Name	**Type**	**Description**
Issuccess	Boolean	Indicates whether the product exists in the hierarchy or not.

Code Sample

The below sample code finds whether the child product is available in the hierarchy of its parent product.

```
1  public Boolean productExistsInHierarchy(String parentProductName, String
   childProductName)
2  {
3      Apttus_Config2.CPQAdminStruct.HierarchyDO hierDO = new
    Apttus_Config2.CPQAdminStruct.HierarchyDO();
4      Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierRequest = null;
5      Apttus_Config2.CPQAdminStruct.HierarchyResponseDO hierResponse = null;
6
```

```
7       // get child, parent products, and option group
8       Product2 parentProduct = [SELECT Id FROM Product2 WHERE Name
    = :parentProductName LIMIT 1];
9       Product2 childProduct = [SELECT Id FROM Product2 WHERE Name
    = :childProductName LIMIT 1];
10
11      hierRequest = new Apttus_Config2.CPQAdminStruct.HierarchyRequestDO();
12      hierDO = new Apttus_Config2.CPQAdminStruct.HierarchyDO();
13      hierDO.ParentProductId = parentProduct.Id;
14      hierDO.ChildProductId = childProduct.Id;
15      hierRequest.HierarchyDOs.add(hierDO);
16      hierResponse =
    Apttus_Config2.CPQAdminWebService.productExistsInHierarchy(hierRequest);
17      return hierResponse.Issuccess;
18  }
```

Removing Products from Hierarchy

You can use this API to remove a product from a category.

API	Signature
removeProductFromHierarchy	*webService static Apttus_Config2.CPQAdminStruct.HierarchyResponseDO removeProductFromHierarchy(Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierarchyRequestDO)*

Request Parameter		
Name	**Type**	**Description**
hierarchyRequestDO	Apttus_Config2.CPQAdminStruct.HierarchyRequestDO	The category request data object.

Request Parameter - Apttus_Config2.CPQAdminStruct.HierarchyRequestDO		
Name	**Type**	**Description**
HierarchyDOs	List<Apttus_Config2.CPQAdminStruct.HierarchyDO>	The list of hierarchy data object

Data Object- Apttus_Config2.CPQAdminStruct.HierarchyDO		
Name	**Type**	**Description**
ChildProductId	Id	The Id of child product.
ParentProductId	Id	The Id of the parent product.

Response Data Object- Apttus_Config2.CPQAdminStruct.HierarchyResponseDO		
Name	**Type**	**Description**
Issuccess	Boolean	Indicates whether the removal of the product from the hierarchy.

Code Sample

The below sample code enables you to remove the products from their relevant category.

```
1  /**
2   * The below code removes the products from category. Invoke below method
   by passing parent and child product names.
3   * Remove a child product from a hierarchy of a parent product.
4   * The input request contains the Child Product Id and the Parent Product
   Id.
5   * In case a list of products is passed, this will be an all-or-none
   scenario stored in
6   * the single Boolean return value.
```

```
 */

public Boolean removeProductFromHierarchy(String parentProductName, String childProductName)
{
    Apttus_Config2.CPQAdminStruct.HierarchyDO hierDO = new Apttus_Config2.CPQAdminStruct.HierarchyDO();
    Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierRequest = new Apttus_Config2.CPQAdminStruct.HierarchyRequestDO();
    Apttus_Config2.CPQAdminStruct.HierarchyResponseDO removeResponse = null;

    // STEP 1 - Retrieve data
    // get child, parent products, and option group
    Product2 parentProduct = [SELECT Id
                        FROM Product2
                        WHERE Name = :parentProductName
                               LIMIT 1];
    Product2 childProduct = [SELECT Id
                    FROM Product2
                    WHERE Name = :childProductName
                    LIMIT 1];

    // STEP 2 - Remove child from hierarchy
    // build request object
    hierDO = new Apttus_Config2.CPQAdminStruct.HierarchyDO();
    hierDO.ParentProductId = parentProduct.Id;
    hierDO.ChildProductId = childProduct.Id;
    hierDO.ParentOptionGroupId = null;
    hierDO.ChildOptionGroupId = null;
    // add hierarchy DO to request
    hierRequest.HierarchyDOs.add(hierDO);

    // send request for removal and save response
    removeResponse = Apttus_Config2.CPQAdminWebService.removeProductFromHierarchy(hierRequest);
    return removeResponse.Issuccess;
}
```

Retrieving the List Option Group

You can use this API to retrieve the list Option Group associated with a product.

API	Signature
getChildOptionGroups	*webService static Apttus_Config2.CPQAdminStruct.HierarchyResponseDO getChildOptionGroups(Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierarchyRequestDO)*

Request Parameter		
Name	**Type**	**Description**
hierarchyRequestDO	Apttus_Config2.CPQAdminStruct.HierarchyRequestDO	The option group request data object.

Request Parameter - Apttus_Config2.CPQAdminStruct.HierarchyRequestDO		
Name	**Type**	**Description**
ProductID	Ids	The ID of the product for which you want to retrieve option groups.

Response Data Object- Apttus_Config2.CPQAdminStruct.HierarchyResponseDO		
Name	**Type**	**Description**
OptionGroupId	List<Id>	The Id option associated with the product.

Code Sample

The below sample code returns the list of option group Ids that are associated with a given product name.

```
public Apttus_Config2.CPQAdminStruct.HierarchyResponseDO
getChildOptions(String productName, String optionGroupName)
{
    Apttus_Config2.CPQAdminStruct.HierarchyDO hierDO = new
 Apttus_Config2.CPQAdminStruct.HierarchyDO();
    Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierRequest = new
 Apttus_Config2.CPQAdminStruct.HierarchyRequestDO();
    Apttus_Config2.CPQAdminStruct.HierarchyResponseDO hierResponse = null;

    // get product and option group
    Product2 prod = [SELECT Id FROM Product2 WHERE Name = :productName
LIMIT 1];
    Apttus_Config2__ClassificationHierarchy__c optionGroup = [SELECT Id
FROM Apttus_Config2__ClassificationHierarchy__c WHERE Name
= :optionGroupName LIMIT 1];

    hierRequest.ProductId = prod.Id;
    hierRequest.OptionGroupId = optionGroup.Id;

    hierResponse =
Apttus_Config2.CPQAdminWebService.getChildOptions(hierRequest);

    return hierResponse;
}
```

Retrieving Options in an Option Group

You can use this API to retrieve the list of options associated with a given option group.

API	Signature
getChildOptions	*webService static Apttus_Config2.CPQAdminStruct.HierarchyResponseDO getChildOptions(Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierarchyRequestDO)*

Request Parameter		
Name	**Type**	**Description**
hierarchyRequestDO	Apttus_Config2.CPQAdminStruct.HierarchyRequestDO	The category request data object.

Request Parameter - Apttus_Config2.CPQAdminStruct.HierarchyRequestDO		
Name	**Type**	**Description**
HierarchyDOs	List<Apttus_Config2.CPQAdminStruct.HierarchyDO>	The list of hierarchy data object

Data Object- Apttus_Config2.CPQAdminStruct.HierarchyDO		
Name	**Type**	**Description**
ChildProductId	Id	The Id of child product.
ParentProductId	Id	The Id of the parent product.

Response Data Object- Apttus_Config2.CPQAdminStruct.HierarchyResponseDO		
Name	**Type**	**Description**
ProductStructureJSONs	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of map, where the key is the option group Id and value is the JSON string of the product

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Key	The option group ID

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Value	The value refers to the JSON String of the product.

Code Sample

The below sample code returns the list of options for the given product and the option group.

```
public Apttus_Config2.CPQAdminStruct.HierarchyResponseDO
getChildOptions(String productName, String optionGroupName)
{
    Apttus_Config2.CPQAdminStruct.HierarchyDO hierDO = new
 Apttus_Config2.CPQAdminStruct.HierarchyDO();
    Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierRequest = new
 Apttus_Config2.CPQAdminStruct.HierarchyRequestDO();
    Apttus_Config2.CPQAdminStruct.HierarchyResponseDO hierResponse = null;

    // get product and option group
    Product2 prod = [SELECT Id FROM Product2 WHERE Name = :productName
LIMIT 1];
    Apttus_Config2__ClassificationHierarchy__c optionGroup = [SELECT Id
FROM Apttus_Config2__ClassificationHierarchy__c WHERE Name
= :optionGroupName LIMIT 1];

    hierRequest.ProductId = prod.Id;
    hierRequest.OptionGroupId = optionGroup.Id;

    hierResponse =
Apttus_Config2.CPQAdminWebService.getChildOptions(hierRequest);

    return hierResponse;
}
```

Retrieving of Product Hierarchy

You can use this API to retrieve the product hierarchy for a given product in JSON array.

API	Signature
getProductStructure	*webService static Apttus_Config2.CPQAdminStruct.HierarchyResponseDO getProductStructure(Apttus_Config2.CPQAdminStruct.HierarchyRequestDO hierarchyRequestDO)*

Request Parameter

Name	Type	Description
hierarchyRequestDO	Apttus_Config2.CPQAdminStruct.HierarchyRequestDO	The hierarchy request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.HierarchyRequestDO

Name	Type	Description
ProductId	Id	The Id of the product for which you want to retrieve hierarchy.

Response Parameter - Apttus_Config2.CPQAdminStruct.HierarchyResponseDO

Field	Type	Description
ProductStructureJSONs	List<Apttus_Config2.CPQAdminStruct.MapDO>	The product hierarchy structure in JSON format.
Success	Boolean	Indicates whether building hierarchy was successful or not.

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO

Field	Type	Description
String	Key	The Id of the product.

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Value	The value refers to the JSON string of the product tree.

Code Sample

The below sample code returns the product hierarchy for the given product in JSON array.

```
1  //The below method returns the entire product hierarchy for the given
   product in JSON Array
2  public List<Apttus_Config2.CPQAdminStruct.MapDO>
   getProductStructure(String productName)
3  {
4      Product2 bundleProduct = [SELECT Id, Name
5                                FROM Product2
6                                WHERE Name = :productName
7                                LIMIT 1];
8
9      Apttus_Config2.CPQAdminStruct.HierarchyRequestDO request = new
    Apttus_Config2.CPQAdminStruct.HierarchyRequestDO();
10     request.ProductIds.add(bundleProduct.Id);
11     Apttus_Config2.CPQAdminStruct.HierarchyResponseDO response =
   Apttus_Config2.CPQAdminWebService.getProductStructure(request);
12
13     return(response != null && response.ProductStructureJSONs.size() > 0 ?
   response.ProductStructureJSONs : null);
14 }
```

Retrieves List of Parent Product

You can use this API to retrieve the list of product Ids that are parents to the given product.

API	Signature
getParentProducts	*webService static List getParentProducts(String productId)*

Request Parameter		
Name	**Type**	**Description**
productId	String	The id product for which you want to retrieve the parent product.

Response Parameter		
Name	**Type**	**Description**
parentProductIDs	List	The list of parent products

Code Sample

The below sample code retrieves the parent products of a given product.

```
public List<Id> getParentProducts(String productName) {
    // get child and parent products and option group
    Product2 childProduct = [SELECT Id FROM Product2 WHERE Name
= :productName LIMIT 1];

    // retrieve product IDs
    List<ID> parentProductIds =
CPQAdminWebService.getParentProducts(childProduct.Id);

    return parentProductIds;
}
```

Retrieves List Product IDs

You can use this API to retrieve the list of product Ids for the given product name.

API	Signature
getProductIds	*webService static List getProductIds(String productName)*

Request Parameter		
Name	**Type**	**Description**
productName	String	The name product for which you want to retrieve the product Ids.

Response Parameter		
Name	**Type**	**Description**
productId	List<String>	The list of product Ids.

Code Sample

The below sample code retrieves the product Ids of a given product name.

```
public List<Id> getParentProducts(String productName) {

    // retrieve product IDs
    List<ID> productIds =
Apttus_Config2.CPQAdminWebService.getProductIds(productName);
    return productIds;
}
```

Retrieves List of Category IDs

You can use this API to retrieve the list of category Ids that are associated with the given product.

API	Signature
getCategoryIds	*webService static List getCategoryIds(String productId)*

Request Parameter		
Name	**Type**	**Description**
productId	String	The Id product for which you want to retrieve the category.

Response Parameter		
Name	**Type**	**Description**
categoryIds	List	The list of category Ids

Code Sample

The below sample code retrieves the category associated with the given product.

```
public void getCategoryIDs(String productName) {
    List<ID> categoryIds;

    // retrieve the productSO
    Product2 productSO = [SELECT Id, Name
                          FROM Product2
                          WHERE Name = :productName
                          LIMIT 1];

    // retrieve associated category ids
    categoryIds =
Apttus_Config2.CPQAdminWebService.getCategoryIDs(productSO.Id);

    retrun ategoryIds;
}
```

Creating New Attribute Group

You can use this API to create attribute group for a set of attributes.

API	Signature
createAttributeGroup	*webService static Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO createAttributeGroup(Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO prodAttrRequestDO)*

Request Parameter		
Name	**Type**	**Description**
prodAttrRequestDO	Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO	The product attribute request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO		
Name	**Type**	**Description**
ProductAttributeDOs	List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>	The list of product attribute data objects.
ProductAttributeGroupMemberSOs	List<Apttus_Config2__ProductAttributeGroupMember__c>	The list of attributes for the product attribute group.

Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeDO		
Name	**Type**	**Description**
AttributeGroupName	String	The name of the attribute group you need to associate attributes to.
ProductAttributesToAdd	List <String>	The API name of the attributes you want to add to the attribute group.
ProductAttributesToRemove	List <String>	The API name of the attributes you want to remove from the attribute group.
ProductId	Id	The ID of the product.

Response Parameter - Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO		
Field	**Type**	**Description**
ProductAttributeGroupResponseMap	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of key-value pairs. The key refers to the attribute group name that you provided and the value contains the record ID.
Success	Boolean	Indicates whether the attribute group was created successfully or not.

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Key	The name of the attribute group.
String	Value	The value refers record of the newly created attribute group.

Code Sample

The sample code below enables you to create an attribute group for the attribute group name and a list of API names of the attributes.

```
public Map<Apttus_Config2__ProductAttributeGroup__c>
createAttributeGroups(String attributeGroupName, List<String>
attributeList)
{
    Apttus_Config2.CPQAdminStruct.ProductAttributeDO requestAttributeDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeDO();

    requestAttributeDO.AttributeGroupName = attributeGroupName;
    requestAttributeDO.ProductAttributesToAdd = attributeList;

    Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO requestDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO();
    requestDO.ProductAttributeDOs = new
 List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>
{requestAttributeDO };

    Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO response =
Apttus_Config2.CPQAdminWebService.createAttributeGroup(requestDO);

    List<Apttus_Config2.CPQAdminStruct.MapDO> attributeGroupDetails =
response.ProductAttributeGroupResponseMap;

    List<Id> attributGroupIds = new List<Id> ();
    for (Apttus_Config2.CPQAdminStruct.MapDO attributeGroup :
attributeGroupDetails)
    {
        attributGroupIds.add(attributeGroup.Value);
    }

    return [Select Id from Apttus_Config2__ProductAttributeGroup__c Where
Id IN :attributGroupIds];
}
```

Adding and Removing Attributes from an Attribute Group

You can use this API to add or remove attributes from attributes groups.

API	Signature
updateAttributeGroup	*webService static Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO updateAttributeGroup(Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO prodAttrRequestDO)*

Request Parameter		
Name	**Type**	**Description**
prodAttrRequestDO	Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO	The product attributes request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO		
Name	**Type**	**Description**
ProductAttributeDOs	List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>	The list of product attribute data objects.
ProductAttributeGroupMemberSOs	List<Apttus_Config2__ProductAttributeGroupMember__c>	The list of attributes to be added to or removed from the attribute group.
Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeDO		
Name	**Type**	**Description**
AttirbuteGroupId	Id	The ID of the attribute group.
AttributeGroupName	String	The name of the attribute group you need to associate attributes to.

Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeDO		
Name	**Type**	**Description**
ProductAttributesToAdd	List <String>	The API name of the attributes you want to add to the attribute group.
ProductAttributesToRemove	List <String>	The API name of the attributes you want to remove from the attribute group.
ProductId	Id	The ID of the product.

Response Parameter - Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO		
Field	**Type**	**Description**
ProductAttributeGroupResponseMap	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of key-value pairs. The key refers to the attribute group name that you provided and the value contains the record ID.
Success	Boolean	Indicates whether the attributes were added or removed successfully.

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Key	The name of the attribute group.
String	Value	The value refers record of the attribute group updated.

Code Sample

The sample code below enables you to add attributes in an attribute group

```
1  public Boolean addAttributesToAttributeGroups(Id attributeGroupId,
   List<String> attributeList)
```

```
{
    Apttus_Config2.CPQAdminStruct.ProductAttributeDO requestAttributeDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeDO();

    requestAttributeDO. AttributeGroupId = attributeGroupId;
    requestAttributeDO.ProductAttributesToAdd = attributeList;


    Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO requestDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO();
    requestDO.ProductAttributeDOs = new
 List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>
{requestAttributeDO };

    Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO response =
Apttus_Config2.CPQAdminWebService.updateAttributeGroup(requestDO);

    return response.Success;
}
```

The sample code below enables you to remove attributes from an attribute group

```
public Map<Apttus_Config2__ProductAttributeGroup__c>
removeAttributesFromAttributeGroups(Id attributeGroupId, List<String>
attributeList)
{
    Apttus_Config2.CPQAdminStruct.ProductAttributeDO requestAttributeDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeDO();

    requestAttributeDO. AttributeGroupId = attributeGroupId;
    requestAttributeDO.ProductAttributesToRemove = attributeList;

    Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO requestDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO();
    requestDO.ProductAttributeDOs = new
 List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>
    {requestAttributeDO};

    Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO response =
Apttus_Config2.CPQAdminWebService.updateAttributeGroup(requestDO);

```

```
    return response.Success;
}
```

Associating Attribute Group and Products

You can use this API to associate attribute group and products.

API	Signature
associateAttributeGroup	*webService static Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO associateAttributeGroup(Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO prodAttrRequestDO)*

Request Parameter		
Name	**Type**	**Description**
prodAttrRequestDO	Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO	The product attribute request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO		
Name	**Type**	**Description**
ProductAttributeDOs	List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>	The list of product attribute data objects.
ProductAttributeGroupMemberSOs	List<Apttus_Config2__ProductAttributeGroupMember__c>	The list of attributes for the product attribute group.

Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeDO		
Name	**Type**	**Description**
AttributeGroupName	String	The name of the attribute group you need to associate with the product.
ProductAttributesToAdd	List <String>	The API name of the attributes you want to add to the attribute group.
ProductAttributesToRemove	List <String>	The API name of the attributes you want to remove from the attribute group.
ProductId	Id	The ID of the product.

Response Parameter - Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO		
Field	**Type**	**Description**
ProductAttributeGroupResponseMap	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of key-value pairs. The key refers to the attribute group name that you provided and the value contains the record ID.
Success	Boolean	Indicates whether the attribute group was associated successfully or not.

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Key	The name of the attribute group.
String	Value	The value refers record of the associated attribute group.

Code Sample

The sample code below enables you associates products and an attribute group.

```
public List<Apttus_Config2__ProductAttributeGroupMember__c>
asscociateAttributeGroups(Id attributeGroupId, Id productId)
{
    Apttus_Config2.CPQAdminStruct.ProductAttributeDO requestAttributeDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeDO();

    requestAttributeDO.AttributeGroupId = attributeGroupId;
    requestAttributeDO.ProductId = productId;

    Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO requestDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO();

    requestDO.ProductAttributeDOs = new
 List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>
{requestAttributeDO};

    Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO response =
Apttus_Config2.CPQAdminWebService.associateAttributeGroup(requestDO);
    return response.ProductAttributeGroupMemberSOs;
}
```

Dissociating Attribute Group and Products

You can use this API to remove the association attribute group and products.

API	Signature
removeAttributeGroup	*webService static Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO removeAttributeGroup(Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO prodAttrRequestDO)*

Request Parameter		
Name	**Type**	**Description**
prodAttrRequestDO	Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO	The product attribute request data object.

Request Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO		
Name	**Type**	**Description**
ProductAttributeDOs	List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>	The list of product attribute data objects.
ProductAttributeGroupMemberSOs	List<Apttus_Config2__ProductAttributeGroupMember__c>	The list of attributes for the product attribute group.
Data Object - Apttus_Config2.CPQAdminStruct.ProductAttributeDO		
Name	**Type**	**Description**
AttributeGroupName	String	The name of the attribute group you need to dissociate with the product.
ProductAttributesToAdd	List <String>	The API name of the attributes you want to add to the attribute group.
ProductAttributesToRemove	List <String>	The API name of the attributes you want to remove from the attribute group.
ProductId	Id	The ID of the product.

Response Parameter - Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO		
Field	**Type**	**Description**
ProductAttributeGroupResponseMap	List<Apttus_Config2.CPQAdminStruct.MapDO>	The list of key-value pairs. The key refers to the attribute group name that you provided and the value contains the record ID.
Success	Boolean	Indicates whether the attribute group was dissociated successfully or not.

Response Data Object - Apttus_Config2.CPQAdminStruct.MapDO		
Field	**Type**	**Description**
String	Key	The name of the attribute group.
String	Value	The value refers record of the dissociated attribute group.

Code Sample

The sample code below enables you dissociates products and an attribute group.

```
public List<Apttus_Config2__ProductAttributeGroupMember__c>
disassociateAttributeGroups(Id attributeGroupId, Id productId)
{
    Apttus_Config2.CPQAdminStruct.ProductAttributeDO requestAttributeDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeDO();

    requestAttributeDO.AttributeGroupId = attributeGroupId;
    requestAttributeDO.ProductId = productId;

    Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO requestDO =
new Apttus_Config2.CPQAdminStruct.ProductAttributeRequestDO();

    requestDO.ProductAttributeDOs = new

List<Apttus_Config2.CPQAdminStruct.ProductAttributeDO>{requestAttributeDO}
;
```

```
11
12      Apttus_Config2.CPQAdminStruct.ProductAttributeResponseDO response =
    Apttus_Config2.CPQAdminWebService.removeAttributeGroup(requestDO);
13      return response.ProductAttributeGroupMemberSOs;
14  }
```

Batch Job Service

The Batch Job Service APIs account for the standard actions to configure, price, and quote.

You can invoke APIs in Batch Job Service using the following command:

```
Apttus_Config2.BatchJobService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the Batch Job Service APIs to complete the following tasks:

- Splitting a Smart Cart
- Updating Price for the Smart Cart
- Finalizing Configuration in Smart Cart
- Retrieving Apex Batch Job Information

Splitting a Smart Cart

This global method splits the line items in the cart into groups in a Smart Cart flow.

API	Signature
splitCart	*static Id splitCart(Id cartId)*

Request Parameter			
Name	**Type**	**Required?**	**Description**
cartID	ID	Yes	The Id of the cart to be split.

Response Parameter		
Field	**Type**	**Description**
jobId	ID	The Id of the job executed to split the cart.

Code Sample

This sample enables you to splits the line items in the cart into groups in a Smart Cart flow. When the method is executed, the task pending flag is set to true on the cart passed as the parameter. After the line items are split into groups, the flag is set to false.

```
1  //Api to split large cart (pass in either Main Cart Id or Config Cart Id).
   This would set the task pending flag on the cart passed as the parameter
   (either main or config). When the split is completed, the flag is cleared.
   Note that it will call the pricing once split is done.
2
3  ID jobId = Apttus_Config2.BatchJobService.splitCart(mainOrConfigCartId);
```

Updating Price for the Smart Cart

This global method updates the prices of the line items after the cart is split.

API	Signature
updatePriceForCart	*static Id updatePriceForCart(Id cartId)*

Request Parameter			
Name	**Type**	**Required?**	**Description**
cartID	ID	Yes	The Id of the cart for which you want to update prices of the line item

Response Parameter		
Field	**Type**	**Description**
jobId	ID	The Id of the batch job executed to update the pricing of the cart

Code Sample

This sample enables you to update the price of the line items after the cart is split. When the method is executed, the task pending flag is set to true on the cart passed as the parameter. After pricing is calculated for the line items, the flag is set to false.

```
1  //Api to price large cart (pass in either Main Cart Id or Config Cart Id),
   if there is a change in any pricing parameter post split.
2  This would set the task pending flag on the cart passed as the parameter
   (either main or config). When all carts are priced, the flag is cleared.
3
4  ID jobId =
   Apttus_Config2.BatchJobService.updatePriceForCart(mainOrConfigCartId);
```

Finalizing Configuration in Smart Cart

This global method finalizes a cart and executes post synchronization tasks in Smart Cart flow.

API	Signature
finalizeCart	*static Id finalizeCart(Id cartId)*

Request Parameter			
Name	**Type**	**Required?**	**Description**
cartID	ID	Yes	The Id of the cart for which you want to finalize

Response Parameter		
Field	**Type**	**Description**
jobId	ID	The Id of the batch job executed to finalize the cart

Code Sample

This sample enables you to finalize a cart in Smart Cart flow after the cart is split & updating price is complete. When the method is executed, the task pending flag is set to true on the cart passed as the parameter. After finalization is completed, the status of product configurations is updated to *Finalized*.

```
/**
*Finalizes and executes post synchronization tasks for the given cart
*@param cartId the cart id
*@return the batch job id
*/
global static ID finalizeCart (ID cartId)
{
    ID jobId =
Apttus_Config2.BatchJobService.finalizeCart(mainOrConfigCartId);
}
```

Retrieving Apex Batch Job Information

This global method retrieved the information of the asynchronous apex job running in the backend for a business object.

You can retrieve current information of the following apex jobs:

- Create and activate Agreement
- Clone proposal and agreement
- Finalize proposal and agreement
- Accept proposal
- Create and accept Order
- Asynchronous cart pricing

API	Signature
getBatchJobInfoForContext	*static Apttus_Config2.CPQStruct.JobInfo getBatchJobInfoForContext(Id ctxObjectId)*

Request Parameter

Name	Type	Required?	Description
ctxObjectId	ID	Yes	The Id of the business object like proposal or agreement.

Response Data Object - Apttus_Config2.CPQStruct.JobInfo

Field	Type	Description
ChildJobs	List	The child apex jobs that are associated the original apex jobs.
CompletedDate	Datetime	The date and time when the original or child of the apex job was completed.
ExtendedStatus	String	The messages of errors encountered during apex job runtime.
JobId	Id	The apex job Id
JobName	String	The functional and apex job name in the following format: <Function name>:<apex job name>
JobType	String	The type of the apex job. For example, queueable and batchable.
ParentJobId	Id	The Id of the given apex job's parent
PercentComplete	Integer	The percentage of completion of the apex job at the time when global method was invoked.

Response Data Object - Apttus_Config2.CPQStruct.JobInfo		
Field	**Type**	**Description**
Status	String	The status of the parent apex job. The following are the possible values: • Processing • Completed • Failed • Aborted
SubmittedDate	Datetime	The date and time when the apex job was submitted.

Code Sample

The above global method can be used in API polling in a specific interval to retrieve the current status of the apex jobs running in the backend. For example, you are building a custom Lightning Web Component to show the live progress bar with the percentage of completion and status of the job in the proposal object. The Below apex controller calls the global method and returns the percentage of completion and the status to the UI controller to handle.

```
1  public static final String STATUS_FAILED = 'Failed';
2  public static final String STATUS_API_CALLOUT_ERROR = 'APIError';
3      public static final String GENERIC_JOB_NAME = 'asynchronous';
4          public static Boolean isLightning = null;
5          public static Boolean isDeployedInPackage = null;
6      public ProposalBannerController(ApexPages.StandardController
   stdController) {
7      }
8      /**
9       * Gets the async job info for the propoal
10      * @param proposalID id of the quote/proposal record
11      * @return map of status, percentcomplete and jobId to the LWC.
12      */
13     @AuraEnabled
14     public static Map<String,Object> getBatchJobInfo(String proposalID){
15         Map<String,Object> responseDO = new Map<String,Object>();
16         String functionalJobName = GENERIC_JOB_NAME;
17         String asyncClassName = '';
```

```
        try {
            Apttus_Config2.CPQStruct.JobInfo parentJobInfo =
Apttus_CPQAPI.BatchUpdateService.getBatchJobInfoForContext (proposalID);
            responseDO.put('parentJobStatus', parentJobInfo.Status);
            responseDO.put('parentPercentComplete',
parentJobInfo.PercentComplete);
            responseDO.put('parentJobId', parentJobInfo.JobId);

            if (parentJobInfo.JobName != null) {
                functionalJobName = parentJobInfo.JobName.split(':')[0];
                asyncClassName = parentJobInfo.JobName.split(':')[1] + ':';
            }
            responseDO.put('jobName', functionalJobName);

            if(parentJobInfo.status == STATUS_FAILED){
                responseDO.put('ExtendedStatus', asyncClassName +
parentJobInfo.ExtendedStatus);

            }
        } catch (Exception e) {
            // We have to ignore this error in LWC
            responseDO.put('ExtendedStatus', e.getMessage());
            responseDO.put('parentJobStatus', STATUS_API_CALLOUT_ERROR);
            responseDO.put('jobName', GENERIC_JOB_NAME);
        }

        return responseDO;
    }
}
```

Constraint Web Service 2

The Constraint web service APIs account for the actions to constraint rules.

You can invoke APIs in Constraint Web Service 2 from the following command:

```
Apttus_Config2.ConstraintWebService2.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the constraint web service APIs to complete the following tasks:

Associating Constraint Rule to Products Added to the Cart

You can use this API to execute constraint rules on a list of products.

API	Signature
associateConstraintRulesForProducts	*webService static Boolean associateConstraintRulesForProducts(Id cartId, List productIds)*

Parameters		
Name	**Type**	**Description**
cartId	Id	The Id of the cart you added products to.
productIds	List<Id>	The list of Ids of the products you added to the cart.

Response Parameter		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates whether the association was successful.

Code Sample

The sample below enables you to executes a constraint rule on a list of line items in the given cart.

```
public Boolean associateConstraintRules(String proposalID)
{

    Apttus_Config2__ProductConfiguration__c cart = [SELECT Apttus_Config2__PriceListId__c,  Id
                                                    FROM Apttus_Config2__ProductConfiguration__c
                                                    WHERE Apttus_QPConfig__Proposald__c =: proposalID
                                                    LIMIT 1];
    //Fetch All products from Product Configuration
    List<Apttus_Config2__LineItem__c> liSOList = [SELECT Apttus_Config2__ProductID__c
                                                  FROM Apttus_Config2__LineItem__c
                                                  WHERE Apttus_Config2__ConfigurationId__c = :cart.Id];
    List<Id> productIds = new List<Id> ();
    for (Apttus_Config2__LineItem__c liSO: liSOList)
    {
            productIds.add(liSO.Apttus_Config2__ProductID__c);
    }
    Boolean isSuccess = Apttus_Config2.ConstraintWebService2.associateConstraintRulesForProducts(cart.Id, productIds);
    return isSuccess;
}
```

Applying Constraint Rule to All Products Added to the Cart

You use this API to applying constraint rules to all products you added to the cart.

API	Signature
applyConstraintRulesOnAddAll	*webService static Apttus_Config2.ConstraintWebService2.ConstraintResult2 applyConstraintRulesOnAddAll(Id cartId)*

Parameters		
Name	**Type**	**Description**
cartId	Id	The Id of the cart you added products to.

Response Parameter - Apttus_Config2.ConstraintWebService2.ConstraintResult2		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates whether the association was successful.

Code Sample

The sample below enabled you to associate a constraint rule to all the line items in the given cart.

```
1  public Boolean associateConstraintRulesOnCart(String proposalID)
2  {
3
4      Apttus_Config2__ProductConfiguration__c cart = [SELECT
   Apttus_Config2__PriceListId__c, Id
5                                                  FROM
   Apttus_Config2__ProductConfiguration__c
6                                                  WHERE
   Apttus_QPConfig__Proposald__c =: proposalID
```

```
    Apttus_Config2.ConstraintWebService2.ConstraintResult2 response =
Apttus_Config2.ConstraintWebService2.applyConstraintRulesOnAddAll(cart.Id)
;
    return response.isSuccess;
}
```

Applying Constraint Rule to Bundles Added to the Cart

You use this API to apply constraint rules to the bundles you added to the cart.

API	Signature
applyConstraintRulesOnAddBundle	*webService static Apttus_Config2.ConstraintWebService2.ConstraintResult2 applyConstraintRulesOnAddBundle(Id cartId, Decimal lineNumber)*

Parameters		
Name	**Type**	**Description**
cartId	Id	The Id of the cart you added products to.
lineNumber	Decimal	The line number of the bundle added to the cart

Response Parameter - Apttus_Config2.ConstraintWebService2.ConstraintResult2		
Field	**Type**	**Description**
IsSuccess	Boolean	Indicates whether the CPQ successfully applied the constraint rule

Code Sample

The sample below enables you to apply a constraint rule to the bundle line item in the given cart.

```
public Boolean associateConstraintRulesOnAddBundle(String proposalID,
String productId)
{

    Apttus_Config2__ProductConfiguration__c cart = [SELECT
Apttus_Config2__PriceListId__c, Id
                                                    FROM
Apttus_Config2__ProductConfiguration__c
                                                    WHERE
Apttus_QPConfig__Proposald__c =: proposalID ]

    //Fetch All products from Product Configuration
    Apttus_Config2__LineItem__c lineItemSO =    [SELECT
Apttus_Config2__ProductID__c,  Apttus_Config2__LineNumber__c
                                                    FROM
Apttus_Config2__LineItem__c
                                                    WHERE
Apttus_Config2__ConfigurationId__c = :cart.Id
                                                    AND
Apttus_Config2__ProductID__c =: productid limit 1];

    Apttus_Config2.ConstraintWebService2.ConstraintResult2 response =
Apttus_Config2.ConstraintWebService2.applyConstraintRulesOnAddBundle(cart.
Id, lineItemSO.Apttus_Config2__LineNumber__c);
    return response.isSuccess;
}
```

Retrieving Products Auto-included to the Cart

You use this API to retrieve IDs of the products added to the cart by the auto-inclusion constraint rule.

API	Signature
getAutoIncludedProductIdsForCart	*webService static List<Id> getAutoIncludedProductIdsForCart(Id cartId)*

Parameters		
Name	**Type**	**Description**
cartId	Id	The Id of the cart you added products to.

Response Parameter		
Field	**Type**	**Description**
response	List<list<ID>>	The IDs of products auto-included to the cart

Code Sample

The sample below retrieves the ID of the products auto-included to cart by the auto-inclusion type constraint rules. The list of ID list is returned after the execution of this API. The ID list contains the ID of the condition products and the list of the ID of associated action products.

```
1  public Map<Id, Set<Id>> getAutoIncludedProductsForCart(String proposalID)
2  {
3
4      Apttus_Config2__ProductConfiguration__c cart = [SELECT
   Apttus_Config2__PriceListId__c, Id
5                                                      FROM
   Apttus_Config2__ProductConfiguration__c
6                                                      WHERE
   Apttus_QPConfig__Proposald__c =: proposalID
7
8      List<list<id>> response = Apttus_Config2.ConstraintWebService2.
   getAutoIncludedProductIdsForCart (cart.Id);
9
10     Map<Id, Set<Id>> autoIncludedIdsByTriggeringProductId  = new Map<Id,
   Set<Id>> ();
11     for (list<id> productIds : response)
12     {
```

```
13
14          for (integer index=1; index <productIds.size() ; index ++)
15          {
16              if
     (autoIncludedIdsByTriggeringProductId.containsKey(productIds[0]))
17              {
18                  autoIncludedIdsByTriggeringProductId.get(productIds[0]).ad
    d(productIds[index]);
19              }
20              else
21              {
22                  autoIncludedIdsByTriggeringProductId.put(productIds[0],
    new set<Id> {productIds[index]});
23              }
24          }
25      }
26      return autoIncludedIdsByTriggeringProductId;
27  }
```

Quote Collaboration Service

The Quote Collaboration Service global methods account for the standard action to Quote Collaboration feature.

Global methods are not like APIs. They can only be invoked in an Apex Code.

You can invoke the method in the Quote Collaboration service using the following command:

```
QuoteCollaborationService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

In this Section:

- Adding Products to a Collaboration Request

Adding Products to a Collaboration Request

This method adds products to an existing collaboration request. This global method is not an API. It can only be invoked in an Apex Code.

Method	Signature
addProductsToCollaboration	*static Apttus_Config2.CollabStruct.AddProductCRResponseDO addProductsToCollaboration(Apttus_Config2.CollabStruct.AddProductCRRequestDO requestDO)*

Parameters

Name	Type	Required?	Description
request	Apttus_Config2.CollabStruct.AddProductCRRequestDO	Yes	The request data object.

Request Data Object - Apttus_Config2.CollabStruct.AddProductCRRequestDO

Field	Type	Description
collaborationRequestId	ID	The id of the collaboration request to which you want add products.
lineItemIds	List	The list of line item IDs you want to add to the collaboration request

Response Data Object - Apttus_Config2.CollabStruct.AddProductCRResponseDO

Field	Type	Description
errorMessages	List	The list of errors occurred while adding products to the collaboration request.
hasErrors	Boolean	Indicates that the errors occurred while adding products to the collaboration request.
isSuccess	Boolean	Indicated that the products are successfully added to the collaboration request.

Code Sample

The following sample enables you to add products to an existing collaboration request with a Collaboration ID. Provide a list of line item IDs of the products you want to add to the collaboration request. If the products are successfully added to the collaboration request, the API returns true, otherwise, the API returns the list of errors occurred while adding the products.

```
/**
 * The below method demonstrates how to add a product in an existing collaboration request
 */
    CollabStruct.AddProductCRRequestDO request = new CollabStruct.AddProductCRRequestDO();
    request.collaborationRequestId = 'a3B1S0000005V3wUAE';
    request.lineItemIds = new List<Id> {'a0a1S000005YwWj'};
    CollabStruct.AddProductCRResponseDO response = QuoteCollaborationService.addProductsToCollaboration(request);
    if (!response.isSuccess)
    {
        system.debug(response.errorMessages);
    }
```

Favorite Configuration Global Service

The Favorite Configuration Global Service API account for the standard actions on the configuration saved as a favorite.

You can invoke APIs in Favorite Configuration Global Service using the following command:

```
Apttus_Config2.FavoriteConfigurationGlobalService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the APIs to complete the following tasks:

- Creating Favorite Configuration

Creating Favorite Configuration

This API creates a favorite configuration from the Catalog page referenced by cartID.

API	Signature
saveFavoriteConfiguration	*static Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteResponseDO saveFavoriteConfiguration(Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteRequestDO requestDO)*

Parameters		
Name	**Type**	**Description**
request	Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteRequestDO	The request data object.

Request Data Object - Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteRequestDO		
Field	**Type**	**Description**
cartID	ID	The identifier of the cart.
favoriteConfigurationSO	Apttus_Config2__FavoriteConfiguration__c	Reference to the Favorite Configuration Object
lineNumbers	List <Integer>	The list of line numbers to be added to the favorite configuration.

Response Data Object - Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteResponseDO		
Field	**Type**	**Description**
favoriteConfigurationSO	Apttus_Config2__FavoriteConfiguration__c	Reference to the Favorite Configuration Object
processedLineNumbers	List<Integer>	List of line numbers added as a favorite configuration.

Data Object - Apttus_Config2__FavoriteConfiguration_c		
Field	**Type**	**Description**
Name	String	Enter a suitable name for your favorite configuration. This name is displayed as a record under your Favorites category.
PriceListId	String	The pricelist id associated with a favorite configuration.
Scope	String	Choose an option for the visibility of your configuration record. The available values are, *Public* and *Private*. If you set this to *Private*, this record is visible to only the owner of the record. If you set this to *Public*, this record is accessible to all the users.

Code Sample

The sample below enables you to create a favorite configuration for a cart with a cartId. You can invoke this API in use cases when you want to show a Favorite configuration page based on the cartID and pricelist.

```
public class QAddToFavorite implements Queueable {
    public Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteRequestDO
requestDO;
```

```
    public
 QAddToFavorite(Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteRequestDO
requestDO)
    {
        this.requestDO = requestDO;

    }

    public void execute(QueueableContext context)
    {
        Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteResponseDO responseDO
= new Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteResponseDO();
        responseDO =
Apttus_Config2.FavoriteConfigurationGlobalService.saveFavoriteConfiguration(requestDO
);
        List<Integer> processedLineNumbers = responseDO.processedLineNumbers;
        List<Integer> unProcessedLineNumbers = new List<Integer>();

        for (Integer lineNumber : requestDO.lineNumbers)
        {
            if (!processedLineNumbers.contains(lineNumber))
            {
                unProcessedLineNumbers.add(lineNumber);
            }
        }

        if (!unProcessedLineNumbers.isEmpty())
        {
            requestDO.favoriteConfigurationSO.id =
responseDO.favoriteConfigurationSO.id ;
            requestDO.lineNumbers = unProcessedLineNumbers;
            system.debug('Request DO is ' + requestDO);
            System.enqueueJob(new QAddToFavorite(requestDO));
        }
    }
}

//Sample code to execute from the anonymous window.

Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteRequestDO requestDO = new
 Apttus_Config2.FavoriteConfigurationStruct.SaveFavoriteRequestDO();
```

```
requestDO.favoriteConfigurationSO.Name = 'QueueableAPI-1 Fav BundleandStd'; // Name
of favorite configuration
requestDO.favoriteConfigurationSO.Apttus_Config2__Active__c = true;
requestDO.favoriteConfigurationSO.Apttus_Config2__Scope__c = 'Public';
requestDO.favoriteConfigurationSO.Apttus_Config2__PriceListId__c = 'a4QP0000000wNyC';
 // priceList ID
requestDO.favoriteConfigurationSO.Apttus_Config2__Description__c = 'Sample
description';
requestDO.cartId = ID.valueOf('a4jP0000000M23t'); // product configuration or cartID
requestDO.lineNumbers = new List<Integer>{1,2,3,4,5,6,7,8,9,10};
System.enqueueJob(new QAddToFavorite(requestDO));
```

Quote/Proposal Config Web Service

The Quote/Proposal Config web service API account for the standard actions to configure, price, and quote.

You can invoke APIs in Quote/Proposal Web Service from the following command:

```
Apttus_QPConfig.QPConfigWebService.<Name of the Function>
where the name of the function is API Name and it parameters.
```

You can use the Quote/Proposal Config web service API to complete the following tasks:

- Accepting a Quote
- Synchronizing Cart
- Copying Product Configuration to the Proposal

Accepting a Quote

This API accepts a quote or proposal.

API	Signature
acceptQuote	*webService static Boolean acceptQuote(Id quoteOrProposalId)*

Request Parameter			
Name	**Type**	**Required?**	**Description**
QuoteID	ID	Yes	The id of the quote/proposal you want to accept.

Response Parameter		
Field	**Type**	**Description**
isSuccess	Boolean	Indicates whether the quote is accepted or not. The ID of the newly created cart object

Code Sample

The following sample enables you to accepts quote with a Quote ID. Provide a Quote ID of the Quote or Proposal you want to accept. If the quote is successfully accepted, the API returns true, otherwise, the API returns false.

```
1  /**
2    * Accepts the given quote/proposal
3    * @param quoteOrProposalId the id of the quote/proposal sobject to
   accept
4    * @return <code>true</code> if the operation was successful,
   <code>false</code> otherwise
5
6   */
7
8      WebService static Boolean acceptQuote(ID quoteOrProposalId) {
9          Id quoteOrProposalId = 'a0Y3C000000u2xv';
10         Boolean isSuccess =
   Apttus_QPConfig.QPConfigWebService.acceptQuote(quoteOrProposalId);
11     }
```

Integration Details

Use the following information in your integrations with CPQ Web Services API. Refer to Integrating Conga with External Systems for information on how to get started.

Response/Request XML

Example Request

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:qpc="http://soap.sforce.com/schemas/class/Apttus_QPConfig/
QPConfigWebService">
    <soapenv:Header>
        <qpc:SessionHeader>
            <qpc:sessionId>00DZ000000NAEIA!
ASAAQNJKTI9h4.FgZutgi4zWkwHRIvIJFNu3_Qh4XPyiODbj2Q3XzSjmRb76.sgyY.eOutBVPu5wFakw8WmXH
CIsOjzI375z</qpc:sessionId>
        </qpc:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <qpc:acceptQuote>
            <qpc:quoteOrProposalId>a0eZ0000005pF4h</qpc:quoteOrProposalId>
        </qpc:acceptQuote>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Response

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns="http://soap.sforce.com/schemas/class/Apttus_QPConfig/QPConfigWebService">
    <soapenv:Body>
        <acceptQuoteResponse>
            <result>true</result>
        </acceptQuoteResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Synchronizing Cart

This API synchronizes the given product configuration

API	Signature
syncConfiguration	*webService static Boolean syncConfiguration(Id configId)*

Request Parameter			
Name	**Type**	**Required?**	**Description**
configId	ID	Yes	The id of the product configuration to synchronize.

Response Parameter		
Field	**Type**	**Description**
isSuccess	Boolean	Indicates whether the synchronization is successful or not.

Code Sample

The sample enables you to synchronize the products (line items) added to the cart with the quote or proposal used to generate the cart.

```
/**
 * Below method demonstrate to finalize the cart by passing the
configuration request Id and execute.
*/

public void syncConfigurationAndSync (Id configId)
{
    // finalize the cart
    Apttus_QPConfig.QPConfigWebService. syncConfiguration(configId);
}
```

Copying Product Configuration to the Proposal

You can use this API to copy product configuration from a saved or finalized proposal to another proposal.

API	Signature
copyProductConfiguration	*webService static Boolean copyProductConfiguration(Id oldProposalId, Id newProposalId)*

Request Parameter

Name	Type	Required?	Description
newProposalId	ID	Yes	The ID of the proposal from which you want to copy the product configuration.
oldProposalId	ID	Yes	The ID of the proposal where you want to copy the product configuration.

Response Parameter

Field	Type	Description
isSuccess	Boolean	Indicates whether the product configuration was copied or not.

Code Sample

The sample enables you to copy product configuration from one proposal to another.

```
1  /**
2  * The below method demonstrates how to copy product configuration from one
   proposal to another one.
3  */
4  public static Boolean copyProductConfiguration(Id oldProposalID, Id
   newProposalId )
5  {
6      Boolean isSuccess = false;
```

```
7       isSuccess =
Apttus_qpconfig.QPConfigWebService.copyProductConfiguration(oldProposalID,
newProposalId); return isSuccess ;
8   }
```

Asset Service

The Asset Service method accounts for the standard action to Asset-based Operation.

You can invoke methods in Asset Service using the following command:

```
Apttus_Config2.AssetService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

Global methods are not APIs. They can only be invoked in an Apex Code.

In this Section:

Fetching Asset Line Items

This global method fetches all Asset Line Items for various Accounts.

API	Signature
getAssetLineItems	*static Apttus_Config2.CPQStruct.QueryAssetsResponseDO getAssetLineItems(Apttus_Config2.CPQStruct.QueryAssetsRequestDO request)*

Parameters		
Name	Type	Description
request	Apttus_Config2.CPQStruct.QueryAssetsRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.QueryAssetsRequestDO		
Field	**Type**	**Description**
AccountIds	List <Id>	List of Account Ids
CustomFilter	String	SOQL condition expression used as a custom filter. For example, user could request only assets after a given billing end date. This is optional.
Descending	Boolean	This indicates whether the assets are sorted in descending order
FieldNames	List	This can be used to include fields you have added to the Asset Line Item object.
Nrecord	Integer	This can be used to limit the number of records returned
Offset	Integer	This can be used to include an offset
SortFields	List	This can be used sort the result based on a list of fields.

Response Data Object - Apttus_Config2.CPQStruct.QueryAssetsResponseDO		
Field	**Type**	**Description**
AssetCount	Integer	The number of assets returned
AssetLineItems	List	List of Asset Line Items.

Code Sample

The code sample below helps you get Asset Line Items for a list of Accounts. Use this global method to fetch all asset line items associated with an account.

```
// create list of account ids
List<ID> listAccount = new List<ID>();
listAccount.add(accountId);

// create and populate request object
Apttus_Config2.CPQStruct.QueryAssetsRequestDO request =  new
 Apttus_Config2.CPQStruct.QueryAssetsRequestDO();
request.AccountIds = listAccount;

// retrieve all fields in Asset Line Item sObject
request.FieldNames = null;

// sort by billing end date (optinal)
request.SortFields = new List<String>{'Apttus_Config2__BillingEndDate__c'};
request.Descending = false; // sort ascending
request.CustomFilter = 'Apttus_Config2__BillingEndDate__c > 2011-04-30'; // optional

// do not offset and do not limit number of records returned
request.Offset = null;
request.Nrecord = null;

// call getAssetLineItems API
Apttus_Config2.CPQStruct.QueryAssetsResponseDO response =
Apttus_Config2.AssetService.getAssetLineItems(request);

List<Apttus_Config2__AssetLineItem__c> listItems = response.AssetLineItems;
```

Getting a count of Asset Line Items

This global method gets the count of Asset Line Item in one or more accounts.

API	Signature
countAssetLineItems	*static Apttus_Config2.CPQStruct.QueryAssetsResponseDO countAssetLineItems(Apttus_Config2.CPQStruct.QueryAssetsRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_Config2.CPQStruct.QueryAssetsRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.QueryAssetsRequestDO		
Field	**Type**	**Description**
AccountIds	List <Id>	List of Account Ids
CustomFilter	String	SOQL condition expression used as a custom filter. For example, user could request only assets after a given billing end date. This is optional.
Descending	Boolean	This indicates whether the assets are sorted in descending order
FieldNames	List	This can be used to include fields you have added to the Asset Line Item object.
Nrecord	Integer	This can be used to limit the number of records returned
Offset	Integer	This can be used to include an offset
SortFields	List	This can be used sort the result based on a list of fields.

Response Data Object - Apttus_Config2.CPQStruct.QueryAssetsResponseDO		
Field	**Type**	**Description**
AssetCount	Integer	The number of assets returned
AssetLineItems	List	List of Asset Line Items.

Code Sample

The code sample below helps you get a count of Asset Line Items for a particular account. Use this global method to fetch all asset line items associated with an account.

```
// create list of account ids
List<ID> listAccount = new List<ID>();
listAccount.add(accountId);

// create and populate request object
Apttus_Config2.CPQStruct.QueryAssetsRequestDO request = new
 Apttus_Config2.CPQStruct.QueryAssetsRequestDO();
request.AccountIds = listAccount;
request.CustomFilter = 'Apttus_Config2__BillingEndDate__c > 2019-04-30'; // optional

// call countAssetLineItems API
Apttus_Config2.CPQStruct.QueryAssetsResponseDO response =
Apttus_Config2.AssetService.countAssetLineItems(request);

ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,'Asset Line Item
Count: ' + response.AssetCount));
```

Swapping Assets

You can invoke this API for swapping Assets.

API	Signature
swapAssets	*static Apttus_Config2.CPQStruct.SwapAssetsResponseDO swapAssets(Apttus_Config2.CPQStruct.SwapAssetsRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_Config2.CPQStruct.SwapAssetsRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.SwapAssetsRequestDO		
Name	**Type**	**Description**
NewStartDate	Date	The new Asset Start Date on Swapping
ProductIds	List<ID>	Id of the product with which the Asset is being swapped.
AssetIds	List<ID>	The Id of the Asset which is swapped.
CartIds	Id	The Id of Cart for which an Asset is being swapped

Response Data Object - Apttus_Config2.CPQStruct.SwapAssetsResponseDO		
Field	**Type**	**Description**
LineItemMap	Map<ID, Apttus_Config2__LineItem__c>	Map of line items of the assets that are being swapped.

Code Sample

The code sample below helps you swap an Asset with another product.

```
// create list of asset ids
List<ID> listAssetId = new List<ID>();
for (AssetLineItemWrapperClass record : wrapperAssetLineItemList)
{
    if (record.selected)
    {
        listAssetId.add(record.assetId);
    }
}

// create list of product ids
List<ID> listProductId = new List<ID>();
for (ProductWrapperClass product : wrapperProductList)
{
    if (product.selected)
    {
```

```
        listProductId.add(product.productId);
    }
}

// create and populate request object
if (objAssetLineItem.Apttus_Config2__StartDate__c != null)
{
    Apttus_Config2.CPQStruct.SwapAssetsRequestDO request = new
 Apttus_Config2.CPQStruct.SwapAssetsRequestDO();
    request.AssetIds = listAssetId;
    request.ProductIds = listProductId;
    request.NewStartDate = objAssetLineItem.Apttus_Config2__StartDate__c;
    request.CartId = cartId;
}

// call swapAssets API
Apttus_Config2.CPQStruct.SwapAssetsResponseDO response =
Apttus_Config2.AssetService.swapAssets(request);

ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,'swapAssets: ' +
response));
```

Getting a list of Products to be swapped with Assets

The global method fetches a list of products that are to be swapped.

> ⚠ Ensure that the Replacement Rule is active for Swapping Assets.

API	Signature
getSwappedProducts	*static Apttus_Config2.CPQStruct.RecommendationResponseDO getSwappedProducts(Apttus_Config2.CPQStruct.RecommendationRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_Config2.CPQStruct.RecommendationRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.RecommendationRequestDO()		
Name	**Type**	**Description**
CartId	Id	Id of Cart for which Swap action is performed
ProductIds	List<ID>	Id of the product with which the Asset is being swapped.

Response Data Object - Apttus_Config2.CPQStruct.RecommendationResponseDO		
Field	**Type**	**Description**
ProductIds	List<ID>	Ids of products that can be swapped with assets.

Code Sample

The code sample below helps you fetch the list of Product Ids that can be swapped with Assets.

```
// create list of product ids
List<ID> listProductId = new List<ID>();
for (ProductWrapperClass product : wrapperProductList)
{
    if (product.selected)
    {
        listProductId.add(product.productId);
    }
}

// create and populate request object
Apttus_Config2.CPQStruct.RecommendationRequestDO request = new
 Apttus_Config2.CPQStruct.RecommendationRequestDO();
request.cartId = cartId;
request.ProductIds = listProductId;

// call getSwappedProducts API
Apttus_Config2.CPQStruct.RecommendationResponseDO response =
Apttus_Config2.AssetService.getSwappedProducts(request);
```

```
ApexPages.addMessage(new
ApexPages.Message(ApexPages.severity.info,'getSwappedProducts: ' + response));
```

Terminating Assets

You can invoke this global method for asset cancellation.

API	Signature
cancelAssets	*static Apttus_Config2.CPQStruct.CancelAssetsResponseDO cancelAssets(Apttus_Config2.CPQStruct.CancelAssetsRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_Config2.CPQStruct.CancelAssetsRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.CancelAssetsRequestDO

Name	Type	Description
CancelDate	Date	The termination date of the Asset
AssetIds	List<ID>	List of asset ids to terminate.
CartId	ID	The id of Cart for which an asset is being canceled.

Response Data Object - Apttus_Config2.CPQStruct.CancelAssetsResponseDO

Field	Type	Description
LineItemMap	Map <ID, Apttus_Config2__LineItem__c>	Line Items that are to be terminated.

Code Sample

The code sample below helps you terminate Assets based on the Termination Date.

```
// create list of asset ids
List<ID> listAssetId = new List<ID>();
for (AssetLineItemWrapperClass record : wrapperAssetLineItemList)
{
    if (record.selected)
    {
        listAssetId.add(record.assetId);
    }
}

// create and populate request object
if (objAssetLineItem.Apttus_Config2__CancelledDate__c != null)
    Apttus_Config2.CPQStruct.CancelAssetsRequestDO request = new
 Apttus_Config2.CPQStruct.CancelAssetsRequestDO();
    request.CancelDate = objAssetLineItem.Apttus_Config2__CancelledDate__c.date();
    request.AssetIds = listAssetId;
    request.CartId = cartId;
}

// call cancelAssets API
Apttus_Config2.CPQStruct.CancelAssetsResponseDO response =
Apttus_Config2.AssetService.cancelAssets(request);

ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,'cancelAssets: ' +
response));
```

Renewing Assets

You can invoke this global method to renew Assets.

API	Signature
renewAssets	*static Apttus_Config2.CPQStruct.RenewAssetsResponseDO renewAssets(Apttus_Config2.CPQStruct.RenewAssetsRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_Config2.CPQStruct.RenewAssetsRequestDO()	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.RenewAssetsRequestDO		
Name	**Type**	**Description**
FarthestAssetEndDate	Boolean	Setting this value to true renews assets using the farthest end date.
RenewEndDate	Date	The Asset End Date for Renewal
RenewTerm	Integer	The Renewal Term for an asset
AssetIds	List<ID>	List of asset ids
CartId	ID	The id of Cart for which an asset is being renewed

Response Data Object- Apttus_Config2.CPQStruct.RenewAssetsResponseDO		
Field	**Type**	**Description**
LineItemMap	Map<ID, Apttus_Config2__LineItem__c>	Returns all line items with all their field values.

Code Sample

The code sample below helps you renew Assets based on the specified Renewal Date or Renewal Term.

```
// create list of asset ids
List<ID> listAssetId = new List<ID>();
for (AssetLineItemWrapperClass record : wrapperAssetLineItemList)
{
    if (record.selected)
```

```
    {
        listAssetId.add(record.assetId);
    }
}

// renew assets using an asset line item record with valid Renewal Date and Term
if (objAssetLineItem.Apttus_Config2__RenewalDate__c != null ||
objAssetLineItem.Apttus_Config2__RenewalTerm__c != null)
{

    // create and populate request object
    Apttus_Config2.CPQStruct.RenewAssetsRequestDO request = new
 Apttus_Config2.CPQStruct.RenewAssetsRequestDO();
    request.CartId = cartId;
    request.AssetIds = listAssetId;
    request.RenewEndDate = objAssetLineItem.Apttus_Config2__RenewalDate__c;
    request.RenewTerm = objAssetLineItem.Apttus_Config2__RenewalTerm__c;
    request.FarthestAssetEndDate = false;

    // call renewAssets API
    Apttus_Config2.CPQStruct.RenewAssetsResponseDO response =
Apttus_Config2.AssetService.renewAssets(request);

    ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info,'renewAssets:
' + response));
}
```

Incrementing Assets

You can invoke **incrementAssets**, a global method to increment assets.

Method	Signature
incrementAssets	*static Apttus_Config2.CPQStruct.IncrementAssetsResponseDO incrementAssets(Apttus_Config2.CPQStruct.IncrementAssetRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_Config2.CPQStruct.IncrementAssetRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.IncrementAssetRequestDO		
Field	**Type**	**Description**
IncrementAssetDOs	List <Apttus_Config2.CPQStructIncrementAssetDO>	List of increment assets data objects.
CartId	ID	The ID of the cart that consists of assets to be incremented.

Data Object - Apttus_Config2.CPQStruct.IncrementAssetDO		
Field	**Type**	**Description**
AssetId	Id	The ID of the Asset you want to apply the increment.
LineAction	String	This defines the update made to the asset. There are two types. • *Increment*: creates a new asset • *Increment or* Merge: updates the existing asset.
NewStartDate	Date	New start date for the updated or new asset.
NewEndDate	Date	New end date for the updated or new asset.
Quantity	Integer	The amount of quantity to be incremented.

Response Data Object- Apttus_Config2.CPQStruct.IncrementAssetsResponseDO		
Field	**Type**	**Description**
LineItemMap	Map<ID, Apttus_Config2__LineItem__c>	Returns all line items with all their field values.

Code Sample

The following sample enables you to increment the quantity of standalone products. You can also provide new start and end dates for the products. If you define **LineAction** as *Increment*, new assets are created and, if you define **LineAction** as *Increment and Merge*, existing assets are updated.

```
Apttus_Config2.CPQStruct.IncrementAssetRequestDO request = new
 Apttus_Config2.CPQStruct.IncrementAssetRequestDO();

for (AssetLineItemWrapperClass record : wrapperAssetLineItemList)
{
     if (record.selected)
    {
          // create and populate request object
          Apttus_Config2.CPQStruct.IncrementAssetDo assetDO = new
 Apttus_Config2.CPQStruct.IncrementAssetDO();
          assetDO.AssetId = record.assetId;
          assetDO.NewStartDate =
record.objAssetLineItem.Apttus_Config2__StartDate__c;
          assetDO.NewEndDate = record.objAssetLineItem.Apttus_Config2__EndDate__c;
          assetDO.Quantity = record.objAssetLineItem.Apttus_Config2__Quantity__c;
          assetDO.LineAction = 'Increment'; // Or 'Increment and Merge'
          request.IncrementAssetDOs.add(assetDO);
     }
}
request.CartId = cartId;

// call incrementAssets API
Apttus_Config2.CPQStruct.IncrementAssetsResponseDO response =
Apttus_Config2.AssetService.incrementAssets(request);
ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info, incrementAssets:
' + response));
```

Changing Assets

You can invoke this global method for changing Assets.

API	Signature
changeAssets	*static Apttus_Config2.CPQStruct.ChangeAssetsResponseDO changeAssets(Apttus_Config2.CPQStruct.ChangeAssetsRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_Config2.CPQStruct.ChangeAssetsRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.ChangeAssetsRequestDO		
Name	**Type**	**Description**
AssetId	List<ID>	List of Asset IDs to change.
CartId	ID	The id of Cart for which an asset is being changed.

Response Data Object - Apttus_Config2.CPQStruct.ChangeAssetsResponseDO		
Field	**Type**	**Description**
Map <ID, LineItem__c>	LineItemMap	Returns all line items with all their field values.

Code Sample

The code sample below helps you make changes to an Asset by invoking this global method.

```
// create list of asset ids
List<ID> listAssetId = new List<ID>();
for (AssetLineItemWrapperClass record : wrapperAssetLineItemList)
{
    if (record.selected)
    {
        listAssetId.add(record.assetId);
    }
}

// create and populate request object
```

```
Apttus_Config2.CPQStruct.ChangeAssetsRequestDO request = new
 Apttus_Config2.CPQStruct.ChangeAssetsRequestDO();
request.AssetIds = listAssetId;
request.CartId = cartId;

// call changeAssets API
Apttus_Config2.CPQStruct.ChangeAssetsResponseDO response =
Apttus_Config2.AssetService.changeAssets(request);

ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info, 'changeAssets: '
 + response));
```

Merging Duplicate Assets

This global method enables you to merge duplicate assets automatically.

Method	Signature
mergeAssets	*static Apttus_Config2.CPQStruct.MergeAssetsResponseDO mergeAssets(Apttus_Config2.CPQStruct.MergeAssetsRequestDO request)*

Parameters		
Name	**Type**	**Description**
Apttus_Config2.CPQStruct.MergeAssetsResponseDO	request	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct.MergeAssetsResponseDO		
Field	**Type**	**Description**
AssetIds	List	The list of the ID of assets you want to merge.
CartId	ID	The ID of the cart that consists of assets to be incremented.

Request Data Object - Apttus_Config2.CPQStruct.MergeAssetsResponseDO		
Field	**Type**	**Description**
MergeAssetItemDO	Apttus_Config2.CPQStruct.MergeAssetDO	The merge asset data objects.

Data Object - Apttus_Config2.CPQStruct.MergeAssetDO		
Field	**Type**	**Description**
EndDate	Date	The asset end date for merge.
ProductId	Id	The ID of the assets to be merged.
Quantity	Integer	The quantity of the asset.
StartDate	Date	The asset start date for merge.

Response Parameter - Apttus_Config2.CPQStruct.**MergeAssetsResponseDO**		
Field	**Type**	**Description**
Errors	List	Returns the list of errors if any.
LineItemMap	Map<ID, Apttus_Config2__LineItem__c>	Returns all line items with all their field values.

Code Sample

The following sample enables you to merge a list of assets in a cart.

```
Apttus_Config2.CPQStruct.MergeAssetsRequestDO request = new
 Apttus_Config2.CPQStruct.MergeAssetsRequestDO();
for (AssetLineItemWrapperClass record : wrapperAssetLineItemList)
{
     if (record.selected)
     {
        request.AssetIds.add(record.assetId);
```

```
    }
}

request.CartId = 'a1I3J000000BKJeUAO';
request.MergeAssetItemDO.startdate = Date.newInstance (2017, 06, 01);

Apttus_Config2.CPQStruct.MergeAssetsResponseDO response =
Apttus_Config2.AssetService.mergeAssets(request);
ApexPages.addMessage(new ApexPages.Message(ApexPages.severity.info, mergeAssets: ' +
response));
```

Suspending Assets

You can invoke this global method to suspend assets. You can process a maximum of 200 line items in each Suspend API call.

API	Signature
suspendAssets	*static Apttus_Config2.CPQStruct.SuspendAssetResponseDO suspendAssets(Apttus_Config2.CPQStruct.SuspendAssetRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_Config2.CPQStruct.SuspendAssetRequestDO	Request object invoked by the method.

Request Data Object - Apttus_Config2.CPQStruct.SuspendAssetRequestDO

Field	Type	Description	Is Required
assetIds	List<Id>	List of asset IDs to be suspended. All the asset IDs must be valid and all the assets must be activated.	Yes
CartId	Id	Product configuration ID.	Yes

Request Data Object - Apttus_Config2.CPQStruct.SuspendAssetRequestDO			
Field	**Type**	**Description**	**Is Required**
CustomData	Map<String, String>	Custom field values to be updated. For example, you can pass the reason for suspension as custom data. You can pass more than one custom field.	No
NewEndDate	Date	Date until which assets must remain suspended. This field is optional in the input. CPQ copies this date to the End Date field in the cart line item. ⓘ The end date passed for asset suspension must be less the end date of the asset. If you provide both start date and end date, CPQ considers the end date you provide in this field as the end date for asset suspension. If you do not provide both end date and term, CPQ considers the asset end date as the end date for asset suspension.	No
NewStartDate	Date	Date from which assets must be suspended. This field is mandatory in the input.	Yes

Request Data Object - Apttus_Config2.CPQStruct.SuspendAssetRequestDO			
Field	**Type**	**Description**	**Is Required**
Term	Integer	Term for which assets must be suspended. This field is optional in the input. CPQ copies this date to the Start Date field in the cart line item. ⓘ If you do not provide the end date but provide the term, CPQ calculates the end date for asset suspension based on the asset suspension start date and term (according to the selling frequency). If you do not provide both end date and term, CPQ considers the asset end date as the end date for asset suspension.	No

Response Data Object - Apttus_Config2.CPQStruct.SuspendAssetResponseDO		
Field	**Type**	**Description**
Errors	List<String>	Contains errors messages if any operations failed.
IsSuccess	Boolean	Indicates whether the suspend asset operation is successful.
LineItemMap	Map<ID,Apttus_Config2__LineItem__c>	Indicates the line items to be suspended.

Code Sample

The following code sample helps you suspend assets.

```
List<Apttus_Config2__AssetLineItem__c> assetLinesToSuspend = [SELECT Id,
Apttus_Config2__StartDate__c
```

```
                                                                FROM
Apttus_Config2__AssetLineItem__c
                                                                WHERE
Apttus_Config2__AccountId__c = '001630000142XpT'
                                                                AND
Apttus_Config2__ProductId__r.Name = 'AutoABO_ORN_Standalone04'];

// asset ids to be suspended
List<Id> assetIds = new List<Id>();
for (Apttus_Config2__AssetLineItem__c assetLineSO : assetLinesToSuspend)
{
    assetIds.add(assetLineSO.Id);
}

Apttus_Config2.CPQStruct.SuspendAssetRequestDO request = new
 Apttus_Config2.CPQStruct.SuspendAssetRequestDO();
request.NewStartDate = assetLinesToSuspend[0].Apttus_Config2__StartDate__c.addDays(90)
;
request.NewEndDate = assetLinesToSuspend[0].Apttus_Config2__StartDate__c.addDays(180)
;
request.CartId='a5663000002DlFN';
request.assetIds = assetIds;

//input custom field values
Map<String, String> CustomData = new Map<String, String>();
CustomData.put('Apttus_Config2__Comments__c', 'Suspending for 3 months due to late
payment');
request.CustomData = CustomData;

Apttus_Config2.CPQStruct.SuspendAssetResponseDO response =
Apttus_Config2.AssetService.suspendAssets(request);
system.debug(response.Errors);
system.debug(response.IsSuccess);
```

Resuming Assets

You can invoke this global method to resume suspended assets.

API	Signature
resumeAssets	*static Apttus_Config2.CPQStruct.ResumeAssetResponseDO resumeAssets(Apttus_Config2.CPQStruct.ResumeAssetRequestDO request)*

Parameters

Name	Type	Description
Request	Apttus_Config2.CPQStruct. ResumeAssetRequestDO	Request object invoked by the method

Request Data Object - Apttus_Config2.CPQStruct. ResumeAssetRequestDO

Field	Type	Description	Is Required
assetIds	List<Id>	List of asset IDs to be resumed.	Yes
CartId	Id	Product configuration ID.	Yes
CustomData	Map<String, String>	Custom field values to be updated. For example, you can pass the reason for resumption as custom data. You can pass more than one custom field.	No
NewStartDate	Date	New start date for the resumed assets. ⓘ This parameter will be effective in future when Conga CPQ supports the resumption of assets from the date specified in this parameter.	No

Response Data Object - Apttus_Config2.CPQStruct. ResumeAssetResponseDO		
Field	**Type**	**Description**
Errors	List<String>	Contains errors messages if any operations failed.
IsSuccess	Boolean	Indicates whether the resume asset operation is successful.
LineItemMap	Map <ID, Apttus_Config2__LineItem__c>	Indicates the line items to be resumed.

Code Sample

The following code sample helps you suspend assets.

```
List<Apttus_Config2__AssetLineItem__c> assetLinesToResume = [SELECT Id,
Apttus_Config2__StartDate__c
                                                            FROM
Apttus_Config2__AssetLineItem__c
                                                            WHERE
Apttus_Config2__AccountId__c = '001630000142XpT'
                                                            AND
Apttus_Config2__ProductId__r.Name = 'AutoABO_ORN_Standalone04'];

// asset ids to be suspended
List<Id> assetIds = new List<Id>();
for (Apttus_Config2__AssetLineItem__c assetLineSO :  assetLinesToResume)
{
    assetIds.add(assetLineSO.Id);
}

CPQStruct.ResumeAssetRequestDO request = new CPQStruct.ResumeAssetRequestDO();
request.CartId = 'a5663000002DQKs';
request.assetIds = assetIds;
request.NewStartDate = Date.today();
CPQStruct.ResumeAssetResponseDO response = AssetService.resumeAssets(request);
system.debug( response );
```

Remote CPQ Admin Controller

The Remote CPQ Admin Controller methods account for the standard operation in CPQ Admin.

You can invoke methods in using the following command:

```
Apttus_Config2.RemoteCPQAdminController.<Name of the Function>
where the name of the function is API Name and it parameters.
```

Global methods are not APIs. They can only be invoked in an Apex Code.

In this Section:

- Searching Custom Fields in Objects

Searching Custom Fields in Objects

You can invoke this global method to retrieve the list of objects in which a given custom field is used.

API	Signature
findCriteriaFieldInCustomSettings	static Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchResponse findCriteriaFieldInCustomSettings(Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchRequest request)

Parameters

Name	Type	Description
request	Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchRequest	Request object invoked by the method.

Resquest Data Object - Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchResponse

Field	Type	Description
FieldName	String	The name field that you want to search.

Response Data Object - Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchRequest		
Name	Type	Description
CustomSettingNames	List	The list of custom settings in which the requested field is present.

Code Sample

The following code sample allows you to fetch the list of objects in which the request custom field is used.

```
Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchRequest request = new
 Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchRequest();
request.FieldName = 'Quantity__c'; // namespace prefixed field name to search
Apttus_Config2.CPQAdminStruct2.CriteriaFieldSearchResponse response =
Apttus_Config2.RemoteCPQAdminController.findCriteriaFieldInCustomSettings(request);
System.debug('*** Matching Custom Setting Names: ' +
JSON.serialize(response.CustomSettingNames));
/* Matching Custom Setting Names: ["Config Asset Pricing Criteria Fields","Config
LineItem Criteria Fields","Config Expression Criteria Fields","Config Custom Display
Columns","Config Data Cache","Config Field Set"]*/
```

Asset Web Services

This section describes the following ABO web services:

- Change Assets
- Renew Assets
- Get a List of Products to be Swapped with Assets
- Swap Assets
- Terminate Assets
- Get a Count of Assets
- Get a List of Assets

Change Assets

You can invoke this API to change the value of assets.

API	Signature
changeAssets	*WebService static Apttus_CPQApi.CPQAsset.ChangeAssetsResponseDO changeAssets(Apttus_CPQApi.CPQAsset.ChangeAssetsRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_CPQApi.CPQAsset.ChangeAssetsRequestDO	Request object passed by the invoker method.

Request Data Object - Apttus_CPQApi.CPQAsset.ChangeAssetsRequestDO

Name	Type	Description
AssetIds	List<ID>	List of asset IDs to be amended.
CartId	ID	ID of the cart for which an asset is being amended.

Response Data Object - Apttus_CPQApi.CPQAsset.ChangeAssetsResponseDO

Field	Type	Description
LineItemMap	List<LineItemEntryDO>	Line items that are amended.
Errors	List<String>	List of error messages if any error occurred.

LineItemEntryDO

Name	Type	Description
LineItemId	ID	Lineitem ID
LineItemSO	Apttus_Config2__LineItem__c	LineItem SObject

Code Sample

The following code sample helps you change the value of assets.

```
Public Apttus_CPQApi.CPQAsset.ChangeAssetsResponseDO changeAssets()
{
    // create and populate request object
    Apttus_CPQApi.CPQAsset.ChangeAssetsRequestDO request = new
 Apttus_CPQApi.CPQAsset.ChangeAssetsRequestDO();

    request.CartId = 'a1I6C000000k6lmUAA';
    // list of primary asset ids to be amended
    request.AssetIds = new List<ID> {'a0e6C000001uATS', 'a0e6C000001aULM'};

    // call changeAssets API
    Apttus_CPQApi.CPQAsset.ChangeAssetsResponseDO response =
Apttus_CPQApi.CPQAssetWebService.changeAssets(request);
    return response;
}
```

Renew Assets

You can invoke this API to renew assets.

API	Signature
renewAssets	*WebService static Apttus_CPQApi.CPQAsset.RenewAssetsResponseDO renewAssets(Apttus_CPQApi.CPQAsset.RenewAssetsRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_CPQApi.CPQAsset.RenewAssetsRequestDO	Request object passed by the invoker method.

Request Data Object - Apttus_CPQApi.CPQAsset.RenewAssetsRequestDO

Name	Type	Description
RenewEndDate	Date	Asset end date for renewal.

Request Data Object - Apttus_CPQApi.CPQAsset.RenewAssetsRequestDO		
Name	**Type**	**Description**
FarthestAssetEndDate	Boolean	Setting this value to true renews assets using the farthest end date.
RenewTerm	Integer	Renewal term for an asset.
AssetIds	List<ID>	List of asset IDs to be renewed.
CartId	ID	ID of the cart for which an asset is being renewed.

Response Data Object - Apttus_CPQApi.CPQAsset.RenewAssetsResponseDO		
Field	**Type**	**Description**
LineItemMap	List<LineItemEntryDO>	Line items that are renewed.
Errors	List<String>	List of error messages if any error occurred.

LineItemEntryDO		
Name	**Type**	**Description**
LineItemId	ID	Lineitem ID
LineItemSO	Apttus_Config2__LineItem__c	LineItem SObject

Code Sample

The following code sample helps you renew assets based on the renewal date, farthest asset end date, or renewal term.

```
Public Apttus_CPQApi.CPQAsset.RenewAssetsResponseDO renewAssets()
{
    // create and populate request object
```

```
    Apttus_CPQApi.CPQAsset.RenewAssetsRequestDO request = new
 Apttus_CPQApi.CPQAsset.RenewAssetsRequestDO();
    request.CartId = 'a1I6C000000k6lmUAA';
    // list of primary asset ids to be renewed
    request.AssetIds = new List<ID> {'a0e6C000001uATS'};
    request.renewenddate = Date.newInstance(YYYY, MM, DD);

    // call renewAssets API
    Apttus_CPQApi.CPQAsset.RenewAssetsResponseDO response =
Apttus_CPQApi.CPQAssetWebService.renewAssets(request);
    return response;
}
```

Get a List of Products to be Swapped with Assets

You can invoke this API to fetch a list of products to be swapped with assets. Relevant replacement rules should be active for getting the list of products that can be swapped with assets.

API	Signature
getSwappedProducts	*WebService static Apttus_CPQApi.CPQAsset.RecommendationResponseDO getSwappedProducts(Apttus_CPQApi.CPQAsset.RecommendationRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_CPQApi.CPQAsset.RecommendationRequestDO	Request object passed by the invoker method.

Request Data Object - Apttus_CPQApi.CPQAsset.RecommendationRequestDO

Name	Type	Description
ProductIds	List<ID>	List of product IDs with which an asset is being swapped.

Request Data Object - Apttus_CPQApi.CPQAsset.RecommendationRequestDO		
Name	**Type**	**Description**
CartId	ID	ID of the cart for which the swap action is performed.

Response Data Object - Apttus_CPQApi.CPQAsset.RecommendationResponseDO		
Field	**Type**	**Description**
ProductIds	List<ID>	List of product IDs that can be swapped with assets.
Errors	List<String>	List of error messages if any error occurred.

Code Sample

The following code sample helps you fetch the list of product IDs that can be swapped with assets.

```
Public Apttus_CPQApi.CPQAsset.RecommendationResponseDO getSwappedProducts()
{
    // create and populate request object
    Apttus_CPQApi.CPQAsset.RecommendationRequestDO request = new
 Apttus_CPQApi.CPQAsset.RecommendationRequestDO();
    request.ProductIds = new List<ID> {'01t3C000000l8qi', '01t2f000000dV1d'};
    request.CartId = 'a1I6C000000k6lmUAA';
    // call getSwappedProducts API
    Apttus_CPQApi.CPQAsset.RecommendationResponseDO response =
Apttus_CPQApi.CPQAssetWebService.getSwappedProducts(request);
    return response;
}
```

Swap Assets

You can invoke this API to swap assets.

API	Signature
swapAssets	*webService static Apttus_CPQApi.CPQAsset.SwapAssetsResponseDO swapAssets(Apttus_CPQApi.CPQAsset.SwapAssetsRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_CPQApi.CPQAsset.SwapAssetsRequestDO	Request object passed by the invoker method.

Request Data Object - Apttus_CPQApi.CPQAsset.SwapAssetsRequestDO

Name	Type	Description
NewStartDate	Date	The new Asset Start Date on Swapping
ProductIds	List<ID>	Id of the product with which an Asset is being swapped
AssetIds	List<ID>	ID of an asset that is swapped.
CartId	ID	ID of the cart for which an asset is being swapped.

Response Data Object - Apttus_CPQApi.CPQAsset.SwapAssetsResponseDO

Field	Type	Description
LineItemMap	List<LineItemEntryDO>	Line items that are cancelled and swapped.
Errors	List<String>	List of error messages if any error occurred.

LineItemEntryDO		
Name	**Type**	**Description**
LineItemId	ID	Lineitem ID
LineItemSO	Apttus_Config2__LineItem__c	LineItem SObject

Code Sample

The following code sample helps you swap assets.

```
Public Apttus_CPQApi.CPQAsset.SwapAssetsResponseDO swapAssets()
{
    // create and populate request object
    Apttus_CPQApi.CPQAsset.SwapAssetsRequestDO request = new
 Apttus_CPQApi.CPQAsset.SwapAssetsRequestDO();
    request.CartId = 'sa1I6C000000k6lmUAA';
    request.NewStartDate = Date.newInstance(YYYY, MM, DD);
    request.ProductIds  = new List<ID>{'01t3C000002oEVL'};
    // only one asset can be swapped per API call
    request.AssetIds = new List<ID> {'a0e6C000001uATS'};

    // call swapAssets API
    Apttus_CPQApi.CPQAsset.SwapAssetsResponseDO response =
Apttus_CPQApi.CPQAssetWebService.swapAssets(request);
    return response;
}
```

Terminate Assets

You can invoke this API to terminate or cancel assets.

API	Signature
cancelAssets	*WebService static Apttus_CPQApi.CPQAsset.CancelAssetsResponseDO cancelAssets(Apttus_CPQApi.CPQAsset.CancelAssetsRequestDO request)*

Parameters		
Name	**Type**	**Description**
request	Apttus_CPQApi.CPQAsset.CancelAssetsRequestDO	Request object passed by the invoker method.

Request Data Object - Apttus_CPQApi.CPQAsset.CancelAssetsRequestDO		
Name	**Type**	**Description**
CancelDate	Date	The termination date of an asset.
AssetIds	List<ID>	List of asset IDs to be terminated.
CartId	ID	ID of the cart for which an asset is being terminated.

Response Data Object - Apttus_CPQApi.CPQAsset.CancelAssetsResponseDO		
Field	**Type**	**Description**
LineItemMap	List<LineItemEntryDO>	Line items that are terminated.
Errors	List<String>	List of error messages if any error occurred.

LineItemEntryDO		
Name	**Type**	**Description**
LineItemId	ID	Lineitem ID
LineItemSO	Apttus_Config2__LineItem__c	LineItem SObject

Code Sample

The following code sample helps you terminate assets.

```
Public Apttus_CPQApi.CPQAsset.CancelAssetsResponseDO cancelAssets()
{
    // create and populate request object
    Apttus_CPQApi.CPQAsset.CancelAssetsRequestDO request = new
Apttus_CPQApi.CPQAsset.CancelAssetsRequestDO();

    request.CartId = 'a1I6C000000ju66';
    // list of primary asset ids
    request.AssetIds = new List<ID> {'a0e6C000001tWxA', 'a0e6C000001wVWB'};
    request.CancelDate = Date.newInstance(YYYY, MM, DD);

    // call cancelAssets API
    Apttus_CPQApi.CPQAsset.CancelAssetsResponseDO response =
Apttus_CPQApi.CPQAssetWebService.cancelAssets(request);
    return response;
}
```

Get a Count of Assets

This API returns the count of asset line items of various accounts.

API	Signature
countAssetLineItems	*WebService static Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO countAssetLineItems (Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO	Request object passed by the invoker method.

Request Data Object - Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO

Name	Type	Description
AccountIds	List<ID>	List of account IDs.

Request Data Object - Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO		
Name	**Type**	**Description**
CustomFilter	String	SOQL condition expression used as a custom filter. For example, you can request only assets after a specified billing end date. This is optional.
Descending	Boolean	N/A
FieldNames	List<String>	N/A
SortFields	List<String>	N/A
Offset	Integer	N/A
Nrecord	Integer	N/A

Response Data Object - Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO		
Field	**Type**	**Description**
AssetCount	Integer	Number of assets returned.
AssetLineItems	List<Apttus_Config2__AssetLineItem__c>	N/A
Errors	List<String>	List of error messages if any error occurred.

Code Sample

The following code sample helps you fetch the count of asset line items of various accounts.

```
Public Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO countAssetLineItems()
{
```

```
    Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO request = new
 Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO();
    request.AccountIds = new List<ID>{'0012f00000BQd03'};
    request.CustomFilter = 'Apttus_Config2__PriceType__c = \'Recurring\''; //
optional
    // call countAssetLineItems API
    Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO response =
    Apttus_CPQApi.CPQAssetWebService.countAssetLineItems(request);
    return response;
}
```

Get a List of Assets

This API returns the list of asset line items of various accounts.

API	Signature
getAssetLineItems	*WebService static Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO getAssetLineItems(Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO request)*

Parameters

Name	Type	Description
request	Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO	Request object passed by the invoker method.

Request Data Object - Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO

Name	Type	Description
AccountIds	List<ID>	List of account IDs.
FieldNames	List<String>	Optional. Use this to include fields you have added to an Asset Line Item object.
CustomFilter	String	Optional. SOQL condition expression used as a custom filter. For example, you can request only assets after a specified billing end date.

Request Data Object - Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO		
Name	**Type**	**Description**
Descending	Boolean	Optional. This indicates whether assets are sorted in descending order.
SortFields	List<String>	Optional. This can be used sort the result based on a list of fields.
Offset	Integer	Optional. This can be used to include an offset.
Nrecord	Integer	Optional. This can be used to limit the number of records returned.

Response Data Object - Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO		
Field	**Type**	**Description**
AssetCount	Integer	N/A
AssetLineItems	List<Apttus_Config2__AssetLineItem__c>	List of asset line items.
Errors	List<String>	List of error messages if any error occurred.

Code Sample

The following code sample helps you get a list of asset line items of various accounts.

```
Public Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO getAssetLineItems ()
{
```

```
    Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO request = new
 Apttus_CPQApi.CPQAsset.QueryAssetsRequestDO();
    request.AccountIds = new List<ID>{'0012f00000BQd03'};
    request.CustomFilter = 'Apttus_Config2__PriceType__c = \'Recurring\''; //
optional
    // retrieve all fields in Asset Line Item sObject
    request.FieldNames = null;
    // sort by billing end date (optional)
    request.SortFields = new List{'Apttus_Config2__BillingEndDate__c'};

    // call countAssetLineItems API
    Apttus_CPQApi.CPQAsset.QueryAssetsResponseDO response =
    Apttus_CPQApi.CPQAssetWebService.getAssetLineItems (request);
    return response;
}
```

Merge Web Service

The Merge web service APIs account for the standard actions on quotes.

You can invoke APIs in Web Service from the following command:

```
Apttus_Proposal.MergeWebService.<Name of the Function>
where the name of the function is API Name and it's parameters.
```

You can use the Merge web service APIs to complete the following tasks:

- Creating Proposal Document
- Creating Proposal Document with Draft Indication
- Generating Documents for Proposal with Large Number of Line Items
- Generating Documents Asynchronously

> The Salesforce API version used in CPQ SOAP APIs is upgraded to 50.0.

Creating Proposal Document

You can use this API to create proposal documents for the given format, template name, document type, and protection level. As a callback action, the API creates a task for further action.

API	Signature
generateDoc	*webService static Id generateDoc(Id templateId, Id proposalId, String pLevel, String docFormat, String sessionId, String serverUrl)*

Request Parameter

Name	Type	Required?	Description
templateId	ID	Yes	The ID of the template you want to use for the proposal.
proposalId	ID	Yes	The ID of the proposal for which you want to generate the document.
pLevel	String	Yes	The protection level you want to apply on the document
docFormat	String	Yes	The format in which you want to generate the document. The valid values for this parameter are listed below: • PDF • DOC • DOCX • RTF
sessionId	String	Yes	The session ID you want to use for the callback.
serverUrl	String	Yes	The server URL you want to use for the callback.

Response Parameter

Field	Type	Description
documentID	ID	The ID of the generated document.

Code Sample

The sample code below enables you to create a proposal document by providing a valid proposal ID, template name, protection level, and format. After the execution, it returns the proposal document ID.

```
/**
 * The below method demonstrates how to create proposal document by passing the proposal ID(auto generated ID as record name),
 * template name, protection level and document format.
 * Possible protection levels are:
 *  - Full access,
 *  - Insert comments and tracked changes only,
 *  - Insert comments only,
 *  - Fill in form fields only and
 *  - Read only
*/
public Id createDocumentForQuote(String ProposalID, String templateName, String protectionLevel, String docFormat)
{
    Id documentId;
    Id proposalSOId = [SELECT ID FROM Apttus_Proposal__Proposal__c WHERE Name = :ProposalID LIMIT 1].Id;
    Id templateSOId = [SELECT Id, Name
                        FROM Apttus__APTS_Template__c
                        WHERE Name = :templateName AND
                        Apttus__IsActive__c = TRUE LIMIT 1].Id;
    if (proposalSOId != null &&  templateSOId != null)
    {
        String serverUrl;
        Apttus_Proposal__ProposalSystemProperties__c prop = Apttus_Proposal__ProposalSystemProperties__c.getInstance('System Properties');
        if (prop.Apttus_Proposal__EnableFile__c )
        {
            serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
            + '/services/Soap/u/50.0/' + UserInfo.getOrganizationId();

        }
        else
        {
            serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
            + '/services/Soap/u/50.0/' + UserInfo.getOrganizationId();
        }
```

```
34
35          documentId =
    Apttus_Proposal.MergeWebService.generateDoc(templateSOId,
36
    proposalSOId,
37
    protectionLevel,
38                                                                  docFormat,
39
    userInfo.getSessionId(),
40
    serverUrl);
41      }
42      return documentId;
43  }
```

Creating Proposal Document with Draft Indication

You can use this API to create proposal documents for the given format, template name, document type, protection level, and an indicator of whether the document is a draft or final document. As a callback action, the API creates a task for further action.

API	Signature
generateDoc2	*webService static Id generateDoc2(Id templateId, Id proposalId, String pLevel, String docFormat, Boolean isDraft, String sessionId, String serverUrl)*

Request Parameter

Name	Type	Required?	Description
templateId	ID	Yes	The ID of the template you want to use for the proposal.
proposalId	ID	Yes	The ID of the proposal for which you want to generate the document.
pLevel	String	Yes	The protection level you want to apply on the document

Request Parameter			
Name	**Type**	**Required?**	**Description**
docFormat	String	Yes	The format in which you want to generate the document. The valid values for this parameter are listed below: • PDF • DOC • DOCX • RTF
isDraft	Boolean	Yes	Indicate whether the generated document is a draft or a final document.
sessionId	String	Yes	The session ID you want to use for the callback.
serverUrl	String	Yes	The server URL you want to use for the callback.

Response Parameter		
Field	**Type**	**Description**
documentID	ID	The ID of the generated document.

Code Sample

The sample code below enables you to create a proposal document by providing a valid proposal ID, template name, protection level, format, draft state. After the execution, it returns the proposal document ID.

```
1  /**
2   * The below method demonstrates how to create draft or final proposal
   document by passing the proposal ID(auto generated ID as record name),
3   * template name, protection level, document format and isDraft indicator.
4   * Possible protection levels are:
5   *  - Full access,
```

```
 *  - Insert comments and tracked changes only,
 *  - Insert comments only,
 *  - Fill in form fields only and
 *  - Read only
*/
public Id createDocumentForQuote(String ProposalID, String templateName,
String protectionLevel, String docFormat, Boolean isDraft){
    Id documentId;
    Id proposalSOId = [SELECT ID FROM Apttus_Proposal__Proposal__c WHERE
Name = :ProposalID LIMIT 1].Id;
    Id templateSOId = [SELECT Id, Name FROM Apttus__APTS_Template__cWHERE
Name = :templateName AND Apttus__IsActive__c = TRUE LIMIT 1].Id;
    if (proposalSOId != null &&  templateSOId != null)
    {
        String serverUrl;
        Apttus_Proposal__ProposalSystemProperties__c prop =
Apttus_Proposal__ProposalSystemProperties__c.getInstance('System
Properties');

        if (prop.Apttus_Proposal__EnableFile__c )
        {
            serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
                        + '/services/Soap/u/50.0/' +
UserInfo.getOrganizationId();

        }
        else
        {
            serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
                      + '/services/Soap/u/50.0/' +
UserInfo.getOrganizationId();
        }

        documentId =
Apttus_Proposal.MergeWebService.generateDoc2(templateSOId,

proposalSOId,

protectionLevel,

docFormat,
                                                                isDraft,
```

```
37
    userInfo.getSessionId(),
38
    serverUrl);
39      }
40      return documentId;
41  }
```

Generating Documents for Proposal with Large Number of Line Items

You can use this API to generate documents for proposals with a large number of line items, asynchronously for the given parameters.

API	Signature
submitGenerateLargeDoc	*webService static Id submitGenerateLargeDoc(Id templateId, Id proposalId, String pLevel, String docFormat, Boolean isDraft, String sessionId, String serverUrl)*

Request Parameter

Name	Type	Required?	Description
templateId	ID	Yes	The ID of the template you want to use for the proposal.
proposalId	ID	Yes	The ID of the proposal for which you want to generate the document.
pLevel	String	Yes	The protection level you want to apply on the document
docFormat	String	Yes	The format in which you want to generate the document. The valid values for this parameter are listed below: • PDF • DOC • DOCX • RTF

Request Parameter			
Name	**Type**	**Required?**	**Description**
isDraft	Boolean	Yes	Indicate whether the generated document is a draft or a final document.
sessionId	String	Yes	The session ID you want to use for the callback.
serverUrl	String	Yes	The server URL you want to use for the callback.

Response Parameter		
Field	**Type**	**Description**
asynccallRequestId	ID	The ID of the asynchronous merge call SObject.

Code Sample

The sample code below enables you to create a proposal document asynchronously for a proposal with a large number of line items, by providing a valid proposal ID, template name, protection level, format, draft state. After the execution, it returns the ID of the asynchronous merge call SObject.

```
/**
 * The below method demonstrates how to create draft or final proposal document asynchronously by passing the proposal ID(auto generated ID as record name),
 * template name, protection level, document format and isDraft indicator.
 * Possible protection levels are:
 *  - Full access,
 *  - Insert comments and tracked changes only,
 *  - Insert comments only,
 *  - Fill in form fields only and
 *  - Read only
*/
```

```
11  public Id submitGenerateLargeDoc (String ProposalID, String templateName,
    String protectionLevel, String docFormat, Boolean isDraft){
12      Id asynccallRequestId;
13      Id proposalSOId = [SELECT ID FROM Apttus_Proposal__Proposal__c WHERE
    Name = :ProposalID LIMIT 1].Id;
14      Id templateSOId = [SELECT Id, Name
15                         FROM Apttus__APTS_Template__c
16                         WHERE Name = :templateName AND
17                         Apttus__IsActive__c = TRUE
18                         LIMIT 1].Id;
19      if (proposalSOId != null &&  templateSOId != null)
20      {
21          String serverUrl;
22          Apttus_Proposal__ProposalSystemProperties__c prop =
    Apttus_Proposal__ProposalSystemProperties__c.getInstance('System
    Properties');
23
24          if (prop.Apttus_Proposal__EnableFile__c )
25          {
26              serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
27                          + '/services/Soap/u/50.0/' +
    UserInfo.getOrganizationId();
28
29          }
30          else
31          {
32              serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
33                        + '/services/Soap/u/50.0/' +
    UserInfo.getOrganizationId();
34          }
35
36         asynccallRequestId =
    Apttus_Proposal.MergeWebService.submitGenerateLargeDoc(templateSOId,
    proposalSOId, protectionLevel, docFormat, isDraft,
    userInfo.getSessionId(), serverUrl);
37      }
38      return asynccallRequestId;
39  }
```

Generating Documents Asynchronously

You can use this API to generate documents asynchronously for the given parameters.

API	Signature
submitGenerateDoc	*webService static Id submitGenerateDoc(Id templateId, Id proposalId, String pLevel, String docFormat, Boolean isDraft, String sessionId, String serverUrl)*

Request Parameter

Name	Type	Required?	Description
templateId	ID	Yes	The ID of the template you want to use for the proposal.
proposalId	ID	Yes	The ID of the proposal for which you want to generate the document.
pLevel	String	Yes	The protection level you want to apply on the document
docFormat	String	Yes	The format in which you want to generate the document. The valid values for this parameter are listed below: • PDF • DOC • DOCX • RTF
isDraft	Boolean	Yes	Indicate whether the generated document is a draft or a final document.
sessionId	String	Yes	The session ID you want to use for the callback.
serverUrl	String	Yes	The server URL you want to use for the callback.

Response Parameter		
Field	**Type**	**Description**
asynccallRequestId	ID	The ID of the asynchronous merge call SObject.

Code Sample

The sample code below enables you to create a proposal document asynchronously by providing a valid proposal ID, template name, protection level, format, draft state. After the execution, it returns the ID of the asynchronous merge call SObject.

```
/**
 * The below method demonstrates how to create draft or final proposal document asynchronously by passing the proposal ID(auto generated ID as record name),
 * template name, protection level, document format and isDraft indicator.
 * Possible protection levels are:
 *  - Full access,
 *  - Insert comments and tracked changes only,
 *  - Insert comments only,
 *  - Fill in form fields only and
 *  - Read only
*/
public Id generateDocAsync (String ProposalID, String templateName, String protectionLevel, String docFormat, Boolean isDraft){
    Id asynccallRequestId;
    Id proposalSOId = [SELECT ID FROM Apttus_Proposal__Proposal__c WHERE Name = :ProposalID LIMIT 1].Id;
    Id templateSOId = [SELECT Id, Name
                       FROM Apttus__APTS_Template__c
                       WHERE Name = :templateName AND
                       Apttus__IsActive__c = TRUE
                       LIMIT 1].Id;
    if (proposalSOId != null &&  templateSOId != null)
    {
        String serverUrl;
        Apttus_Proposal__ProposalSystemProperties__c prop = Apttus_Proposal__ProposalSystemProperties__c.getInstance('System Properties');

```

```
24          if (prop.Apttus_Proposal__EnableFile__c )
25          {
26              serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
27              + '/services/Soap/u/50.0/' + UserInfo.getOrganizationId();
28
29          }
30          else
31          {
32              serverUrl = URL.getSalesforceBaseUrl().toExternalForm()
33                        + '/services/Soap/u/50.0/' +
UserInfo.getOrganizationId();
34          }
35
36         asynccallRequestId =
Apttus_Proposal.MergeWebService.submitGenerateDoc(templateSOId,
37
proposalSOId,
38
protectionLevel,
39
docFormat,
40
isDraft,
41
userInfo.getSessionId(),
42
serverUrl);
43      }
44      return asynccallRequestId;
45  }
```

Async APIs Using Batch Apex

Async APIs are used to execute certain operations using Batch Apex Jobs internally. These Batch jobs are usually used for carts with a large number of line items. You can use standard Salesforce APIs like Database.executeBatch, to trigger the Apex Jobs. IDs of these apex Jobs can be used to track the status of the job. You are notified through email about the completion of the job, as there is no synchronous response from the API.

The following Batch Apex job is described in this section:

- Repricing the Cart

Repricing the Cart

You can use this Batch Apex Job to price the Line Items in a given Cart asynchronously. This Batch Apex creates one batch for each primary line item in the cart.

API	Signature of the Constructor
RepriceCartJob	*global RepriceCartJob(ID cartId, String cartTitle);*

Request Parameter			
Name	**Type**	**Required?**	**Description**
cartId	ID	Yes	The ID of cart you want to reprice.
cartTitle	String	Yes	The title of the cart.

Code Sample

The below code sample returns job ID and reprices the line item one by one based on the batch size. The cart used in the sample code has products that added both manually and through an API. After the job is completed, CPQ sends you the email confirming the completion of the repricing action. If the Apex class encounters any run-time errors, the repricing is aborted.

```
1  public Id rePriceCartJob(Id cartId, String cartTitle) {
2  // create the reprice cart batch job
3  ID jobId = Database.executeBatch(new Apttus_Config2.RepriceCartJob(cartId,
   cartTitle), 1);
4      return jobId;
5  }
```

Scenarios

- Working with Products in the Shopping Cart
  After you set up the user roles and use either sites or communities to build the basic framework for your cart page, create a visual force page and a sample CSS to define

the look and feel for your cart. You can then proceed with using the CPQ APIs to create a working catalog page.

Working with Products in the Shopping Cart

After you set up the user roles and use either sites or communities to build the basic framework for your cart page, create a visual force page and a sample CSS to define the look and feel for your cart. You can then proceed with using the CPQ APIs to create a working catalog page.

- Searching Products
- Configuring and Adding Products
  In the product catalog, you can see the list of products that a vendor offers. Using this list, you can add a product to your cart, configure the product attributes, such as color and memory. If the product is a bundled product, you can select the options with which you want to bundle your product. This section lists the scenarios using which the customer can view the configuration options for a product.
- Displaying Recommendations
  In the product catalog, you can see the list of products that a vendor offers. When the customer selects a product, you can display the list of recommended products on the same page. This section lists the scenarios using which the recommended products appear. To configure recommendations for a product use the following APIs:
- Updating Price
- Viewing Price Break-up Details
- Adding Price Ramps to a Cart
  After you add a line item to a cart, this API enables you to add primary and secondary ramp line items for the line item. Once the ramp line items are created, you can also update the ramp line item details or delete ramp line item details using standard SOQL queries.
- **Viewing Configured Order in the Shopping Cart**
  After the customer has configured and added all the products to the cart, display the detailed shopping cart items. This section lists all the scenarios using which the customer can view the finalized shopping cart.
- Updating Taxes and Shipping for an Order
- Creating and Updating Quote and Quote Line Items
- Creating and Updating Order and Order Line Items
- Asset-Based Ordering
  The end-user or a renewals manager can view and modify products that the

organization owns and should be able to filter the assets to renew orders. Display and modify the assets that belong to a company based on the following criteria:

- Finalizing the Cart
- Updating Quote Terms to Modify Dates

Searching Products

A customer browsing a product catalog can navigate for a product and select the required product manually or can search for a product. You can enable searches using the search text field in the cart page and enable the user to search for a desired product by category, sub-category, or options available with the product. This section lists all the possible scenarios for which a customer can search or view a product listing.

To browse products

1. Create a cart using Creating a Cart from a Quote.
2. Browse products by categories and subcategories using the search text entered by the user by utilizing Retrieving Products and List Prices For a Price List Category and Search Text and Retrieving Products and List Prices For a Price List and Search Text. Using this API, the customer can search for a product directly or select a category and enter search text to refine the search further.
3. Browse for categories and option groups using Retrieving Categories for a Price List and Retrieving Option Groups, Options, and List Prices for a Price List Product.
4. Browse for products using Retrieving Products and List Prices for a Price List and Retrieving Products and List Prices for a Price List and Category.
5. Compare the displayed products using Comparing Products.

Configuring and Adding Products

In the product catalog, you can see the list of products that a vendor offers. Using this list, you can add a product to your cart, configure the product attributes, such as color and memory. If the product is a bundled product, you can select the options with which you want to bundle your product. This section lists the scenarios using which the customer can view the configuration options for a product.

When you configure and add products to your cart, you can do the following:

- Show Product / Bundle Details
- Create Cart (Create Quote first and then Cart for the Quote)
- Configure Bundles

- Add Single Product to Cart
- Add Multiple Products to Cart
- Show Recommendations for Selected Product
- Show Validation and other messages
- Auto-Include Products
- Delete single product from Cart
- Delete Multiple products from Cart
- Apply any constraint rules related to Cart
- Remove any constraint rules for a deleted product.

To add products and apply constraint rules in the cart

1. Create a cart using createCart.
2. Add products and options to the cart using addBundle , addMultiProducts, or Adding Options to a Bundle.
3. Associate constraint rules for the added products, by querying the right set of rules, using associateConstraintRules.
4. Apply constraint rules to the products in the cart using applyConstraintRules.
5. View the results of applying the constraint rules using getConstraintRuleResult. If the results return *NeedMoreProcessing*, then applyConstraintRules must be run again.

To delete products and apply constraint rules for deleted line items

1. Delete products from the cart using removeBundle or Removing Multiple Bundles from a Cart
2. Remove constraint rules for a deleted product using Applying Constraint Rules to Deleted Products
3. View the results of removing the constraint rules using getConstraintRuleResult.

Displaying Recommendations

In the product catalog, you can see the list of products that a vendor offers. When the customer selects a product, you can display the list of recommended products on the same page. This section lists the scenarios using which the recommended products appear. To configure recommendations for a product use the following APIs:

Scenario 1

After a customer searches a product, the customer selects a product by clicking the product

name or image in the result view. The recommendations for that product along with product details should appear.

Scenario 2
After a customer searches a product, the customer adds the product to the cart using the Add to Cart button. On the Cart Details page, the recommendations for that product along with product details should appear along with the cart line items.

For each recommended product, display the product image, name, description, and price.

Updating Price

If you want to update price for a cart after a specific interval use the Updating Price For A Cart API.

You can use the Update Price API for your cart in the following scenarios:

- When the quantity of a product is incremented or decremented.
- When an additional product is added to your cart.
- If any additional charges apply for the product added.
- If any Adjustment whether Markup or Discounts are applied to the cart.

Viewing Price Break-up Details

If the customer wants to view the breakup of how the tiered pricing rules are applied for a given quantity / term of the line item use the Price Breakup for a Cart or Specific Line Item.

The customer might want to see the breakup of how the tiered pricing rules are applied for a given quantity / term of the line item. For example:

There is a tiered pricing structure for a given quantity as follows:

- 0-20 - 5% Discount
- 21-30 - 10% Discount
- > 30 - 15% Discount

If the line item has 25 quantities then show the pricing breakup as

- Product 1 - List Price: $10 Quantity: 20 Discount: 5% Net Price: $190
- Product 1 - List Price: $10 Quantity: 5 Discount: 10% Net Price: $180

Tiered pricing is applicable in the following scenarios:

- Tiered Pricing for a Single Larger Tiers

    - Tiered Pricing defined in Price List
    - Tiered Pricing negotiated in the quote
    - Tiered pricing negotiated in pricing agreement
- Tiered Pricing for a Single Order leading to higher tier e.g. single order of quantity 25 in above case
- Tiered Pricing for new and add-on order leading to higher tier e.g. initial order of 15 and then add-on (increment) asset based order of 10 quantity
- Tiered for different price types
    - Usage Tiers
    - One time Charge Tiers
    - Recurring subscription tiers
- Ramps

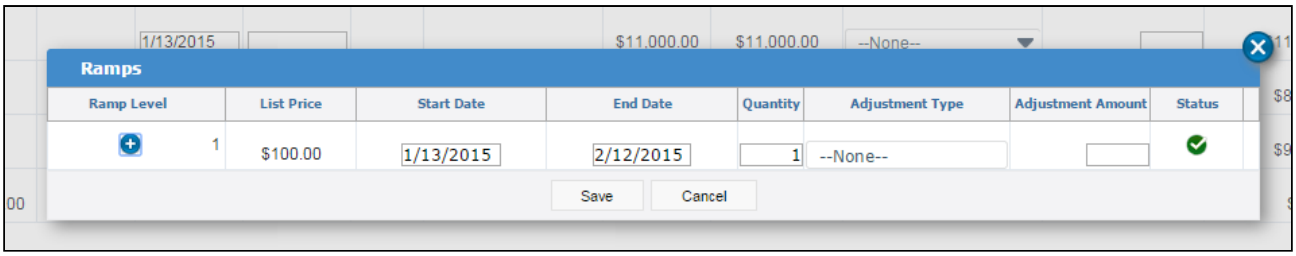
The breakup details for each line item should display a cart id, line id, and price break up details, such as quantity tier, list price, base price, extended price, adjustment, and net price.

Adding Price Ramps to a Cart

After you add a line item to a cart, this API enables you to add primary and secondary ramp line items for the line item. Once the ramp line items are created, you can also update the ramp line item details or delete ramp line item details using standard SOQL queries.

When you use CPQ out of the box, invoke the ramp using the red icon to the left of the primary line item.

Once you click the ramp icon, the ramp dialog appears:



The ramp dialog allows you to add, edit dates and quantity, make adjustments, save the changes, and cancel the changes.

Once the customer adds a ramp to a cart line item, they can do the following:

- Edit the start date, end date, quantity, adjustment type and adjustment amount based on the custom setting.
- The start date of a ramp line item defaults to the end date+1 of the previous line item.
- The end date of a ramp line item defaults to a date such that the difference between the start date and end date is the same as that of the previous line item.
- The user can add more ramp line items after or in between the ramp line items.
- The user can remove the new line before saving by clicking on the icon in the right most column.

Use the addMultiProducts API to add products to the cart.

```
addMultiProducts
```

Parameters		
Name	**Type**	**Description**
request	CPQ.AddMultiProduct RequestDO	The request data object.

Request Data Object - CPQ.AddMultiProductRequestDO		
Field	**Type**	**Description**
CartId	ID	The Id of the cart.
SelectedProducts	List <CPQ.SelectedProductDO>	The list of selected product data objects.

Response Data Object - CPQ.AddMultiProductResponseDO		
Field	**Type**	**Description**
LineNumbers	List<Decimal>	The list of line numbers added to the cart.

Data Object - CPQ.SelectedProductDO		
Field	**Type**	**Description**
ProductId	ID	Id of the product bundle.
Quantity	Decimal	The bundle quantity.
SellingTerm	Decimal	The bundle selling term.
StartDate	Date	The start date. You should ensure you use the correct date format.
EndDate	Date	The end date.
Comments	String	Comments associated with the record.
CustomFields	List<String> CustomFields	List of custom fields created for your product.
CustomData	Apttus_Config2__LineItem__c CustomData	This can be used to include the list of custom fields you have added to the product.

Code Sample

The sample below enables you to add ramp line items after you have:

- Added Products to the cart using the AddMultiProducts APIs,
- Updated the Price for the added products using the updatePriceforCart API.
- Selected the products for which you want to add a ramp for.

Using the sample below you fetch the list of selected products for which you want to add a ramp. You also fetch the parameters for each of the selected products. For all the ramps you create, set PriceGroup as Price Ramp and PricingStatus as Pending. For a primary line item, set IsPrimaryLine__c = true, IsPrimaryRampLine__c = true, and PrimaryLineNumber__c = 1.

```
public void createRampLineItems()
    {
        List<String> rampLineItems = new List<String>();

        if(lstWrapItems.size() > 0)
        {
            // Create a list of selected products for which you want to create a ramp for
            for(LineItemWrapperClass objLineItemWrapperClass : lstWrapItems)
            {
                if(objLineItemWrapperClass.Selected)
                {
                    rampLineItems.add(objLineItemWrapperClass.Name);
                }
            }

            // Sort Ramp Line Items by name
            rampLineItems.sort();

            Integer rampLineItemIndex = 1;

            for(String rampLineItemName : rampLineItems)
            {
                // Get Line Item parameters for the selected products
                Apttus_Config2__LineItem__c lineItem = [SELECT Apttus_Config2__ItemSequence__c, Apttus_Config2__PricingStatus__c, Apttus_Config2__PriceGroup__c,
                Apttus_Config2__IsPrimaryRampLine__c, Apttus_Config2__IsPrimaryLine__c, Apttus_Config2__LineNumber__c, Apttus_Config2__PrimaryLineNumber__c from Apttus_Config2__LineItem__c WHERE
                Name=:rampLineItemName];

                //Set the parameters for each of the line items
                lineItem.Apttus_Config2__PriceGroup__c = 'Price Ramp';
```

```
                lineItem.Apttus_Config2__PricingStatus__c = 'Pending';
                lineItem.Apttus_Config2__LineNumber__c = 1;
                lineItem.Apttus_Config2__PrimaryLineNumber__c = 1;
                lineItem.Apttus_Config2__ItemSequence__c =
rampLineItemIndex;
                //For a primary line item set the following
            `   if(rampLineItemIndex == 1)
                {
                    lineItem.Apttus_Config2__IsPrimaryLine__c = true;
                    lineItem.Apttus_Config2__IsPrimaryRampLine__c = true;
                }

                //For all secondary line items set the following
parameters
                else
                {
                    lineItem.Apttus_Config2__IsPrimaryLine__c = false;
                    lineItem.Apttus_Config2__IsPrimaryRampLine__c = false;
                }

                // Update Line Items
                update lineItem;

                rampLineItemIndex++;
            }
        }
        else
        {
            ApexPages.addMessage(new
 ApexPages.Message(ApexPages.severity.info, 'No line items available.'));
        }
    }
```

Updating Taxes and Shipping for an Order

After the user has added products to the cart, you can apply taxes to each of the cart line items and shipping charges for the entire order.

- Tax and shipping can be calculated for the Quotes (Estimated tax / Shipping amount), but will be more generally used with E-Commerce orders with Pay Now scenario as the actual tax and shipping amount.

After the user has added the products to the cart, the sequence of actions is as follows:

- Customer specifies the Ship-to location.
- Once the ship to account or location is specified at the header (line item level in case of multiple ship-to locations)level, invoke the APIs to calculate Shipping as well as tax.
- Once the shipping and tax charges are calculated, the total price, shipping, and tax amount is displayed for a customer to review.
- Customer will then specify Payment option and Place the order.

Tax Calculation Workflow:

- Create Tax callback class.
- Implement tax interface *ITaxCallback2* in the callback class to call the tax engine to calculate the total tax for the cart (Quote / Order) and breakup for each line item.
- Invoke *computeTaxforCart* API to which we just pass the "CartId". The API will create the tax breakup line items as well as the total tax line for the entire cart.

Shipping Callback Workflow:

- Create Shipping callback class.
- Implement shipping interface IShippingCallback in the callback class to call the shipping engine to calculate the total shipping charges for the cart (Quote / Order).
- Invoke computeShippingForCart to which we just pass the "CartId". The API will create the shipping charges for the entire cart.

Sample Callback for Calculating Shipping

This section comprises the sample callback class that enables you to apply an appropriate shipping charge to a cart.

Before you define the shipping callback ensure that you have set the following:

- If shipping charges change based on the ship to or shipping address of the account, ensure that the account associated with the proposal has both specified.

In addition, you can add additional custom fields on the quote or order line item such as Shipping Speed to calculate the shipping charge.

The sample callback class comprises the following objects:

Data Object - Apttus_Config2.ShippingInput		
Field	**Type**	**Description**
ShippingAddress	Address	Shipping Address. The address for shipping charge calculation obtained from the "Ship To" account for the line item. Accordingly, Ship To Account should be populated on each line item by the implementation.
Item	SObject	Apttus_Config2__LineItem__c. The Cart Line item. This is the reference to the cart line items. Implementation can use this cart line item to get any additional custom fields that might be needed to calculate Shipping.
Handback	Object	Any Apex Object passed by the caller and passed back to the caller in the result. Handback is used only in case when the shipping needs to be calculated for an object other than Cart line item. For example, Billing Schedule line items. • For calculating shipping for cart , there is no need to pass the Handback object. It can be blank • For calculating the Shipping for object other than line items, "Item" field will be blank and the actual object will be passed as handback object to identify object for which shipping is calculated.

Data Object - Apttus_Config2.CustomClass.ShippingResult		
Field	**Type**	**Description**
ShippingCharge	Decimal	Shipping amount total of all the cart line items. This is the total shipping amount for the line item. Line item is identified by the sequence in which the shipping result is added. This needs to be calculated and populated by the implementation.
Handback	Object	Any Apex Object passed by the calling class and passed back to the calling class in the result. Handback is used only in case when the shipping charge needs to be calculated for an object other than Cart line item e.g. Billing Schedule line items. • For calculating shipping charges for a cart, there is no need to pass the Handback object. It can be blank • For calculating the shipping charges for objects other than line items, "Item" field will be blank and the actual object will be passed as handback object to identify the object for which shipping charges are calculated.

Once you have set shipping to, shipping address or shipping speed to your quote or order, create a Shipping Callback class and specify the name of the callback class in **Custom Settings > Config Custom Classes > (Edit) Custom Classes > Shipping Callback Class**.

In the sample callback class below, we fetch and create a list of all the products added to the cart. Then we fetch the shipping charges applicable for a product using an SOQL query. If a corresponding shipping code and a valid shipping charge exist for a product in the order or cart, the total shipping amount for a cart is added directly.

Sample Callback for Calculating Taxes

This section comprises the sample callback class that enables you to apply an appropriate tax to a cart line item.

Before you define the tax callback ensure that you have set the following:

- Go to the product price list item, click **Edit > Tax & Billing** and select the **Taxable?** checkbox. Also, ensure that the **Tax Inclusive?** checkbox is deselected. If it is selected, the product price specified includes the tax amount.

- From the **Tax & Billing** tab, specify the tax code applicable for a product using a lookup. The tax code maps to a code in an external system where various taxes, tax charges, tax types are mapped against a tax code.
- If a tax is applicable based on the billing or shipping address of the account, ensure that the account associated with the proposal has both specified.

The sample callback class comprises the following objects:

Data Object - Apttus_Config2.TaxInput		
Field	**Type**	**Description**
TaxCode	String	Tax code for the line item as inherited from the Price list item or overridden at the line item level.
TaxAddress	Address	Tax Address. Address as obtained for the line item based on precedence hierarchy specified. For example, Line Item Location ==> Line Item Ship To ==> Line Item Bill To ==> Cart Header Ship To ==> Cart Header Bill To ==> Cart Header Sold to
TaxableAmount	Decimal	Net Price from the Line Item in case the tax needs to be calculated on the Net Price. In case the tax needs to be calculated based on List or Base Price then the total amount calculated based on list or base price needs to be used from the line item to calculate tax.
Item	SObject	Apttus_Config2__LineItem__c. This is the reference to the cart line items. Implementation can use this cart line item to get any additional custom fields that might be needed to calculate tax.

Data Object - Apttus_Config2.TaxInput		
Field	**Type**	**Description**
Handback	Object	Any Apex Object passed by the caller and passed back to the caller in the result. Handback is used only in case when the tax needs to be calculated for an object other than Cart line item e.g. Billing Schedule line items. For calculating tax for cart line items, there is no need to pass the Handback object. It can be blank. For calculating the tax for object other than line items, "Item" field will be blank and the actual object will be passed as handback object to identify object for which tax is calculated.

Data Object - Apttus_Config2.CustomClass.TaxResult		
Field	**Type**	**Description**
TaxAmount	Decimal	This is the total amount for the line item. This is sum of the breakup tax amounts for the line item. Line item is identified by the sequence in which the tax result is added. This needs to be calculated and populated by the implementation.
TaxBreakups	List<Apttus_Config2__TaxBreakup>	Tax breakup provide the different components of the tax for a given line item. This needs to be implemented and provided by the implementation. Tax breakup has structure specified below.

Data Object - Apttus_Config2.CustomClass.TaxResult		
Field	**Type**	**Description**
Handback	Object	Any Apex Object passed by the calling class and passed back to the calling class in the result. Handback is used only in case when the tax needs to be calculated for an object other than Cart line item e.g. Billing Schedule line items. • For calculating tax for cart line items, there is no need to pass the Handback object. It can be blank • For calculating the tax for object other than line items, "Item" field will be blank and the actual object will be passed as handback object to identify object for which tax is calculated.

Data Object - Apttus_Config2__TaxBreakup		
Field	**Type**	**Description**
TaxType	String	Type of tax to be applied. For example, sales, custom, excise
TaxRate	Decimal	Tax percentage to be applied
TaxAppliesTo	String	The field on which the tax is applied to. For example, net price.
TaxAmount	Decimal	The tax amount calculated based on the formulae defined in the callback.

Data Object - Apttus_Config2__TaxBreakup		
Field	**Type**	**Description**
BreakupType	String	If you specify the breakup type as Detail, then the quote or order line items show the detailed tax break up for a cart line item.

Tax Breakup Type Example

Breakup Type	Tax Type	Tax Rate	Tax Applies to	Tax Amount
Detail	State Tax	10	Net Price	100
Detail	City Tax	1	Net Price	10
Total				110

Data Object - Apttus_Config2. CustomClass.Address	
Field	**Type**
Street	String
City	String
State	String
County	String
PostalCode	String
Country	String

Once you have set a Tax Code for each price list item, create a Tax Callback class and specify the name of the callback class in **Custom Settings > Config Custom Classes > (Edit) Custom Classes > Tax Callback Class**.

In the sample callback class below, we fetch and create a list of all the products added to the cart. Then we fetch the tax charges applicable for a product using an SOQL query. If a corresponding tax code and a valid tax charge exists for a product in the order or quote line item, the total tax amount for a product is calculated using the formula- taxAmount += (taxRate != null ? (taxRate * input.TaxableAmount) / 100 : 0). After the total tax on a product is calculated, you can list the tax break up based on Tax types applied for each of the order or quote line items. The sample callback lists the parameters applicable for a tax break up. If you want to apply tax to shipping charges as well, you can capture shipping amount for each line item in a custom field and use the total amount (Net Price + Shipping Amount) from the line item to calculate tax. In this case shipping should be calculated first and then tax.

In the sample callback class, tax is calculated using hardcoded rates. For actual implementations, tax rates and tax amounts are fetched from external Tax calculation engines such as Avalara and Vertex to calculate tax.

```
/**
 *  Apttus Config & Pricing
 *  QATaxCallBack
 *
 *  @2013-2014 Apttus Inc. All rights reserved.
 */
global with sharing class QATaxCallBack implements
 Apttus_Config2.CustomClass.ITaxCallback2 {

    /**
     * Callback invoked to compute tax based on the given input
     * @param input the tax input
     * @return the tax result
     */
    global Apttus_Config2.CustomClass.TaxResult
computeTax(Apttus_Config2.CustomClass.TaxInput input) {
      // Compute Tax
      system.debug('Entering into computeTax');
      List<Apttus_Config2.CustomClass.TaxResult> results =
computeTaxMultiple(new Apttus_Config2.CustomClass.TaxInput[]{input});
```

```
18        return (!results.isEmpty() ? results[0] : new
 Apttus_Config2.CustomClass.TaxResult());
19      }
20
21      /**
22       * Callback invoked to compute tax based on the given list of inputs
23       * @param inputs the list of tax inputs
24       * @return the list of tax results
25       */
26      global List<Apttus_Config2.CustomClass.TaxResult>
computeTaxMultiple(List<Apttus_Config2.CustomClass.TaxInput> inputs) {
27
28          List<Apttus_Config2.CustomClass.TaxResult> results = new
 List<Apttus_Config2.CustomClass.TaxResult>();
29
30          for (Apttus_Config2.CustomClass.TaxInput input : inputs) {
31
32              Decimal taxAmount = 0;
33
34              Apttus_Config2.CustomClass.TaxResult result = new
 Apttus_Config2.CustomClass.TaxResult();
35
36              List<Apttus_Config2__TaxBreakup__c> taxBreakUps = new
 List<Apttus_Config2__TaxBreakup__c>();
37
38              // Add the handback object to correlate the result with the
input
39              result.Handback = input.Handback;
40
41              // Compute tax based on tax code and tax address
42
43              // Tax Code is required
44              if (input.TaxCode == null || input.TaxCode.trim().length() ==
0 || input.TaxableAmount == null || input.TaxableAmount == 0) {
45
46                // Tax Amount
47                result.TaxAmount = 0;
48              } else {
49
50                  List<QA_Tax_Rate__c> accountTaxRates = new
 List<QA_Tax_Rate__c>();
51
52                  SObject sObj = input.Item;
53                  Id taxCodeId = (Id)
sObj.get('Apttus_Config2__TaxCodeId__c');
```

```
54
55                  accountTaxRates =  [SELECT Id, Name, Tax_Applies_To__c,
    Tax_Rate__c, Tax_Type__c
56                  FROM QA_Tax_Rate__c
57                  WHERE Tax_Code__c =:  taxCodeId];
58
59                  for(QA_Tax_Rate__c accountTaxRate : accountTaxRates) {
60
61                      Decimal taxRate = accountTaxRate.Tax_Rate__c;
62                      String taxAppliesTo =
    accountTaxRate.Tax_Applies_To__c;
63                      String taxType = accountTaxRate.Tax_Type__c;
64
65                      Apttus_Config2.CustomClass.Address addr =
    input.TaxAddress;
66
67                      // Calculate Tax Amount
68                      taxAmount += (taxRate != null ? (taxRate *
    input.TaxableAmount) / 100 : 0);
69
70                      // Calculate Tax BreakUps
71                      Apttus_Config2__TaxBreakup__c taxBreakUp = new
     Apttus_Config2__TaxBreakup__c();
72                      taxBreakUp.Apttus_Config2__TaxType__c = taxType;
73                      taxBreakUp.Apttus_Config2__TaxRate__c = taxRate;
74                      taxBreakUp.Apttus_Config2__TaxAppliesTo__c =
    taxAppliesTo;
75                      taxBreakUp.Apttus_Config2__TaxAmount__c = taxRate *
    input.TaxableAmount / 100;
76                      taxBreakUp.Apttus_Config2__BreakupType__c = 'Detail';
77
78                      taxBreakUps.add(taxBreakUp);
79                  }
80
81
82                  result.TaxBreakups = taxBreakUps;
83                  result.TaxAmount = taxAmount;
84              }
85              results.add(result);
86          }
87          return results;
88      }
89  }
```

To apply taxes and shipping charges

1. Create a cart using the Creating a Cart from a Quote.
2. Browse products by categories and subcategories using the search text entered by the user by utilizing Retrieving Products and List Prices For a Price List Category and Search Text and Retrieving Products and List Prices For a Price List and Search Text. Using this API, the customer can search for a product directly or select a category and enter search text to refine the search further.
3. Browse for categories and option groups using Retrieving Categories for a Price List and Retrieving Option Groups, Options, and List Prices for a Price List Product.
4. Browse for products using Retrieving Products and List Prices for a Price List and Retrieving Products and List Prices for a Price List and Category.
5. Compare the displayed products using Comparing Products.
6. Add products to the cart by using Adding Products to a Cart and Adding a Bundle to a Cart.
7. Update the price using Updating Price For A Cart.
8. Finalize the cart using Finalizing a Cart.
9. Define the Tax and Shipping Callbacks using Sample Callback for Calculating Taxes and Sample Callback for Calculating Shipping.
10. Specify the Callback class names at **Custom Settings > Config Custom Classes (Edit) > Tax Callback and Shipping Callback**.
11. Apply the taxes and shipping charges using Computing Taxes for Cart Line Items and Computing Shipping for Cart Line Items. The Tax API creates and populates - the tax breakup for each applicable line item in the cart and the total Tax line and related breakup. The shipping API calculates the shipping amount for the entire order.
12. Update the Order Line Items using Synchronizing a Cart.

Creating and Updating Quote and Quote Line Items

A customer with a valid account can create a quote from an existing opportunity. Using the Quote ID and a valid price list, you can create a cart from scratch and by browsing the product catalog you can navigate for a product and select the required product manually or can search for a product. You can enable searches using the search text field in the cart page and enable the user to search for a desired product by category, sub-category, or options available with the product. This section lists all the possible scenarios for which a customer can create a quote and or view a product listing.

To create and update quote and quote line items

1. Create a cart using the Creating a Cart from a Quote.
2. Browse products by categories and subcategories using the search text entered by the user by utilizing Retrieving Products and List Prices For a Price List Category and Search Text and Retrieving Products and List Prices For a Price List and Search Text. Using this API, the customer can search for a product directly or select a category and enter search text to refine the search further.
3. Browse for categories and option groups using Retrieving Categories for a Price List and Retrieving Option Groups, Options, and List Prices for a Price List Product.
4. Browse for products using Retrieving Products and List Prices for a Price List and Retrieving Products and List Prices for a Price List and Category.
5. Compare the displayed products using Comparing Products.
6. Add products to the cart by using Adding Products to a Cart and Adding a Bundle to a Cart.
7. Update the price using Updating Price For A Cart.
8. Finalize the cart using Finalizing a Cart.
9. Update the Quote Line Items using Synchronizing a Cart.

Creating and Updating Order and Order Line Items

A customer with a valid account can create an order from an existing opportunity. Using the order ID and a valid price list, you can create a cart from scratch and by browsing the product catalog you can navigate for a product and select the required product manually or can search for a product. You can enable searches using the search text field in the cart page and enable the user to search for a desired product by category, sub-category, or options available with the product. This section lists all the possible scenarios for which a customer can create a order and or view a product listing.

To create and update order and order line items

1. Create an order using Creating an Order
2. Create a cart for an order using Creating a Cart for an Order.
3. Browse products by categories and subcategories using the search text entered by the user by utilizing Retrieving Products and List Prices For a Price List Category and Search Text and Retrieving Products and List Prices For a Price List and Search Text. Using this API, the customer can search for a product directly or select a category and enter search text to refine the search further.

4. Browse for categories and option groups using Retrieving Categories for a Price List and Retrieving Option Groups, Options, and List Prices for a Price List Product.
5. Browse for products using Retrieving Products and List Prices for a Price List and Retrieving Products and List Prices for a Price List and Category.
6. Compare the displayed products using Comparing Products.
7. Add products to the cart by using Adding Products to a Cart and Adding a Bundle to a Cart.
8. Update the price using Updating Price For A Cart.
9. Update the order line items using Synchronizing a Cart to an Order.
10. Create Asset Line Items using Create Asset Line Items for an Order.
    For more information on the Orders APIs, see Order Web Service.

Asset Based Ordering

The end user or a renewals manager can view and modify products that the organization owns and should be able to filter the assets to renew orders. Display and modify the assets that belong to a company based on the following criteria:

- All assets with recurring, usage, and one-time charge types for which renewal date is approaching.
- All licenses that expire within 30, 60, or 90 days.
- Expired licenses.
- All services that expire in 30 days.
- All assets that are in the End of Life or End of Sale state by the vendor.
- All services that are no longer active or are cancelled should be displayed.
- All software licenses setup for a site.
- All products under the CPQ product category.

The assets should be filtered and displayed based on asset status, price type, and subscription expiration.

The displayed results of the asset list the product name, ID, asset name, asset ID, SKU ID, quantity, price type, price UOM, purchase date, start and end date, base price, adjustment, net price, asset status, and renewal term.

To view assets

Retrieve asset results that fulfill the criteria using the Retrieving Asset Line Items API. Using this API, you can fetch the assets.

To modify assets

1. Retrieve asset results that fulfill the criteria using the Retrieving Asset Line Items API. Using this API, you can fetch the assets.
2. You can renew, amend, increment, or terminate the assets using the code snippets for Modifying Assets (Deprecated).

To create assets from orders

Create assets from an order using the Create Asset Line Items for an Order API. Using this API, you can create assets.

Finalizing the Cart

The customer can finalize the cart in the following scenarios:

- All the required products have been added.
- All requisite prices have been adjusted accordingly.

Invoke the Finalizing the Cart to enable the user to finalize the cart.

Updating Quote Terms to Modify Dates

For a finalized quote, a customer can choose to apply add-ons. The term,validity period, and pricing have to be recalculated after the quote is finalized, approved, and a subsequent proposal is generated. Use the Updating Quote Terms to achieve the scenarios highlighted.

The *updateQuoteTerms* API behavior is as follows:

- Update the carts in Finalized or Ready for Finalization status. The latest cart from the quote in any of the two statuses are used as basis to updating the quote.
- Create a new version of the cart in case of Finalized cart and supersede the current version of the cart. The status of the new version of the cart will be same as the status of the current version of the cart. If the cart was in Finalized status, the new cart will also be in finalized status and system will update the "Finalized date" on the quote header.
- Auto-Synch cart line items with the Proposal Line Items.

To synchronize cart and update quote

1. Finalize the cart using the Finalizing the Cart API.
2. Synchronize the finalized cart with the quote using the Synchronize API.
3. Update the selling terms of the products associated to the quote using the Update Quote Terms API.

Troubleshooting CPQ SOAP APIs

Refer to this topic for descriptions of API error messages thrown by Salesforce and suggested actions to take. An exception is an error caused during code execution. There are a number of built-in Apex exceptions as well as custom exceptions provided by Conga.

The following exceptions that are thrown by Salesforce that can be handled by Conga and converted to custom error messages:

- System.Exception: Exception thrown under most circumstances.
- System.CalloutException: Exception thrown when communicating with external systems.
- DMLException: Exception thrown when performing CRUD ('Create', 'Read', 'Update', 'Delete') actions.

Refer to the following table for a list of custom error messages provided by Conga for some SOAP APIs, included suggested actions to take to resolve the exception.

API	Error Message	Suggested Action
Apttus_API.CPQWebService.addCustomBundle	You do not have the level of access necessary to perform the operation you requested. Contact the owner of the record or your administrator for access.	Authentication or access (system permissions) issue. Contact an administrator or check the appropriate topic for authentication instructions.
Apttus_API.CPQWebService.addMultiProducts		
Apttus_API.CPQWebService.abandonCart		
Apttus_API.CPQWebService.removeMultiBundles		
Apttus_API.CPQWebService.removeOptions		

API	Error Message	Suggested Action
Apttus_Config2.ConstraintWebService2.associateConstraintRulesForProducts		
Apttus_Config2.ConstraintWebService2.applyConstraintRulesOnAddBundle		
Apttus_Config2.CPQAdminWebService.createProducts		
Apttus_Config2.CPQAdminWebService.createCategories		
Apttus_Config2.CPQAdminWebService.associateProductToCategory		
Apttus_Config2.CPQAdminWebService.removeProductFromHierarchy		
Apttus_Config2.CPQWebService.createBundleLineItems		
Apttus_Config2.CPQWebService.createProductLineItems		
Apttus_Config2.CPQWebService.finalizeConfiguration		
Apttus_Proposal.ProposalWebService.createProposalFromAccount		

API	Error Message	Suggested Action
Apttus_Proposal.ProposalWebService.createProposalFromOpportunity		
Apttus_QPConfig.QPConfigWebService.cloneProposal		
Apttus_Config2.ConstraintWebService2.applyConstraintRulesOnAddAll	You do not have the level of access necessary to perform the operation you requested. Contact the owner of the record or you administrator for access.	Authentication or access (system permissions) issue. Contact an administrator or check the appropriate topic for authentication instructions.
	Too many products found constraint rule actionName.	You must reduce the number of products associated with constraint rule actions.
Apttus_Config2.CPQAdminWebService.buildHierarchy	You do not have the level of access necessary to perform the operation you requested. Contact the owner of the record or your administrator for access.	Authentication or access (system permissions) issue. Contact an administrator or check the appropriate topic for authentication instructions.
	The category is currently associated as an Option Group to Product(s).	This is an invalid input. You must provide only category hierarchies and not option groups.
Apttus_Config2.CPQWebService.AbandonConfiguration	You do not have the level of access necessary to perform the operation you requested. Contact the owner of the record or your administrator for access.	Authentication or access (system permissions) issue. Contact an administrator or check the appropriate topic for authentication instructions.
	Error encountered in callback processing.	Check that the appropriate callback is being processed for this call.

The following are standard Salesforce exceptions that cannot be handled or changed:

- LimitException

- UnexpectedException
- AssertException
- NoAccessException
- NoDataFoundException
- SerializationException
- VisualforceException

You can learn more about these exceptions and recommended actions to take by consulting Salesforce documentation.

REST API Guide

This section explains the REST APIs provided by Conga Configuration, Pricing, and Quoting (CPQ).

Topic	Description
What's Covered	This section walks API developers through the list of REST APIs provided by Conga.
Primary Audience	API developers
IT Environment	Refer to the latest *Conga CPQ Release Notes* for information on System Requirements and Supported Platforms.
Updates	For a comprehensive list of updates to this section for each release, see the What's New in CPQ Documentation topic.
Other Resources	Refer to *Conga CPQ Release Notes* for information on system requirements and supported platforms, new features and enhancements, resolved issues, and known issues for a specific release.

This section describes the following topic:

- CPQ Asset-Based Ordering APIs

Before using Conga CPQ, you must be familiar with the following:

- Basic knowledge of Salesforce
- Basic knowledge of REST APIs
- Salesforce and Conga terms and definitions

Select one of the following topics for more information:

CPQ Asset-Based Ordering APIs

(Open API documentation is only available to view online)

Service CPQ APIs

(Open API documentation is only available to view online)

Service Execute Job

You can use the CPQRESTServiceExecuteJob REST API to trigger maintenance or batch jobs. The following are the URI and request details in the maintenance or batch jobs supported by the CPQRESTServiceExecuteJob API with some sample data:

```
*  Supported URI patterns:
*    URI - /Apttus_Config2/ExecuteJob/ConfigDataStaticResourceUpsertJob
*    Request Body - NA
*
*    URI - /Apttus_Config2/ExecuteJob/CategoryMaintenanceBatchJob
*    Request Body - {
*                "CategoryIds": [
*                           "01t5400000CiRG3",
*                           "01t5400000CiRG4"
*                         ]
*             }
*
*    URI - /Apttus_Config2/ExecuteJob/BundleUpdateJob
*    Request Body - {
*                "ProductIds": [
*                           "01t5400000CiRG3",
*                           "01t5400000CiRG4"
*                         ]
*             }
*
*    URI - /Apttus_Config2/ExecuteJob/UpdatePriceQJob
*    Request Body - {
*                "CartId": "a5654000000cwDK"
*             }
*
*    URI - /Apttus_Config2/ExecuteJob/RepriceCartJob
*    Request Body - {
*                "CartId": "a5654000000cwDK"
*                "CartTitle": "Philips Demo Quote"
*             }
```

```
*
```

CPQ Best Practices: Governor Limits

CPQ Best Practices: Governor Limits explains Salesforce governor limits and best practices for extending Conga CPQ.

Topic	Description
What's Covered	This section explains the best practices to configure and extend CPQ functionalities. The section covers error scenarios and recommendations to handle such errors.
Primary Audience	This document is intended for: • Implementation teams that customize and extend the standard functionality provided by CPQ on Salesforce • CPQ Administrators
IT Environment	Refer to the latest *Conga CPQ Release Notes* for information on System Requirements and Supported Platforms.
Other Resources	• Refer to *Conga CPQ* Release Notes for information on system requirements and supported platforms, new features and enhancements, resolved issues, and known issues for a specific release.

This section provides the following information:

- Established governor limits of Salesforce
- Best Practices to avoid governor limits
- Handling governor limits when encountered
- Guidelines for CPQ
- Recommendations to improve the performance in different areas of CPQ
- Parameters contributing to the performance of CPQ

Before using CPQ, you must be familiar with the following:

- Advanced Salesforce administration
- Salesforce and Conga terms and definitions

Select one of the following topics for more information:

- About Governor Limits
- Guardrail for Using CPQ

About Governor Limits

Salesforce governor execution limits ensure the efficient use of resources on the Force.com multi-tenant platform, which is a single resource shared by many customers and organizations. To ensure that no customer monopolizes resources, Salesforce.com has created a set of limits that governs the code execution. Whenever a governor limit is exceeded, the platform halts the execution of the program and displays an error.

Apex code is executed on Salesforce.com servers as part of atomic transactions. Apex transactions ensure the integrity of data. The Apex run-time engine strictly enforces limits to ensure that the Apex code or processes do not monopolize shared resources.

A single trigger execution is one transaction. Most of the governor limits are per-transaction based and some limits are not transaction based. For example, the number of API calls in an org or email notifications that are reset every 24 hours. Salesforce has established the following governor limits.

- Per-Transaction Apex Limits
- Per-Transaction Certified Managed Package Limits
- Lightning Platform Apex Limits
- Static Apex Limits
- Size-Specific Apex Limits
- Miscellaneous Apex Limits

For example, Per-Transaction Apex Limits count for each Apex transaction. For Batch Apex, these limits are reset for each execution of a batch of records in the execute method.

> When CPQ features are implemented in a specific way, they can reach the Per-Transaction Apex Limits. Therefore, this section describes Per-Transaction Apex Limits. For other Salesforce governor limits, refer to https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm.

Established Salesforce Governor Limits

Salesforce governor limits are implemented in the following key areas.

Key Area	Limit
Apex CPU Timeout	Salesforce has a timeout limit for transactions based on the CPU usage. If a transaction consumes too much CPU time, Salesforce ends it as a long-running transaction. The timeout is set to a maximum of 10 to 15 seconds for each transaction.
Heap Size	The A*pex heap size too large* error occurs when too much data is being stored in memory during processing. This is the amount of memory allocated to objects defined in the Apex Code. The limit depends on the type of execution (for example, synchronous and asynchronous calls). Salesforce has set a limit of 6 MB for synchronous transactions and 12 MB for asynchronous transactions.
View State	The view state of a web page is composed of the data that is necessary to maintain the state during a server request. Salesforce has set this limit to 170 KB.
Salesforce Object Query Language (SOQL)	There is a limit to the number of records that can be retrieved or updated by a SOQL or data manipulation language (DML) query. Salesforce displays an error when this limit is exceeded. Salesforce has set the maximum number of records that can be returned in a single query to 50000.

The following table lists the limits for synchronous Apex and asynchronous Apex (Batch Apex and @future methods) when they are different. Otherwise, this table lists only one limit that applies to both synchronous and asynchronous Apex.

Description	Synchronous Limit	Asynchronous Limit
Total number of SOQL queries issued	100	200
Total number of records retrieved by SOQL queries	50000	
Total number of records retrieved by *Database.getQueryLocator*	10000	
Total number of SOSL queries issued	20	

Description	Synchronous Limit	Asynchronous Limit
Total number of records retrieved by a single SOSL query	2000	
Total number of DML statements issued	150	
Total number of records processed as a result of DML statements, *Approval.process*, or *database.emptyRecycleBin*	10000	
Total stack depth for any Apex invocation that recursively initiates triggers due to insert, update, or delete statements	16	
Total number of callouts (HTTP requests or web services calls) in a transaction	100	
Maximum cumulative timeout for all callouts (HTTP requests or web services calls) in a transaction	120 seconds	
Maximum number of methods with the @future annotation allowed for each Apex invocation	50	0 in batch and future contexts; 1 in queueable context
Maximum number of Apex jobs added to the queue with *System.enqueueJob*	50	1
Total number of send email methods allowed	10	
Total heap size	6 MB	12 MB
Maximum CPU time on the Salesforce servers	10000 milliseconds	60000 milliseconds
Maximum execution time for each Apex transaction	10 minutes	
Maximum number of push notification method calls allowed for each Apex transaction	10	
Maximum number of push notifications that can be sent in each push notification method call	2000	

For more information on Execution Governors and Limits, refer to https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm.

For more information on Salesforce Developer Limits and Allocations Quick Reference, refer to https://resources.docs.salesforce.com/220/latest/en-us/sfdc/pdf/salesforce_app_limits_cheatsheet.pdf.

Best Practices to Avoid Governor Limits

As a Salesforce Code Developer, you must follow some best practices to ensure that your code is scalable and does not exceed the governor limit. The following are such best practices:

- Bulkify your code.
- Avoid SOQL queries or DML statements inside For loops.
- Bulkify your helper methods.
- Use collections, streamlining queries, and efficient For loops.
- Streamline multiple triggers on the same object.
- Query large data sets.
- Use the limits of Apex methods to avoid exceeding governor limits.
- Use @future appropriately.
- Use batch Apex if you are working for more than 50000 records.

For more information, refer to Apex best Practices.

Best Practices for Customization

While implementing the following customization, you might encounter problems.

- **Roll up Summary Fields:** When you define Roll up summary fields on the Quotation object and clone a quote with line items, you may encounter issues like Apex CPU Time Out, because the calculation happens each time a new line item is created. Avoid defining Roll up summary fields if possible.
- **Formula Fields:** Because formula fields are calculated at runtime, they impact performance. For example, if you have 50 formula fields on a line item and there are 100 line items in the cart, 5000 formulas are calculated in a single SOQL. Use numeric expressions instead.
- **SOQL:** Any Query that takes 200 milliseconds or more is considered long-running. Optimize the query by requesting relevant fields and indexing fields in the *where*

clause. Enable debug, keep database logs, keep the value of **Apex Profiling** to *Finish* in Debug Level Details, and turn off everything in log filter.

The following table lists the CPQ objects and the customization you can perform on those objects.

CPQ Objects	Customization
• Proposal • Proposal Line Item • Product Configuration • Line Item • Product Attribute Value • Quote • Quote Line Item • Opportunity	• Custom Fields • Formula Fields • Roll up Summary Fields • Workflows • Process Builder • Validation rules • Triggers • SOQL • Callbacks

Follow the recommendations when you customize CPQ objects:

- Execute triggers on appropriate events, not on all events. For example, execute triggers before insert, after an update, and so on.
- Avoid defining a trigger on Line Item objects.
- Invoke any Apex record trigger, class, or extension for more than 200 records.
- Review any DML statements in a trigger, because these statements can trigger another set of triggers.
- Avoid using rollup summary fields if you have a trigger in place on the same object. Perform the required calculation in the trigger. If it cannot be avoided, the recommended maximum limit is 20.
- Avoid using more than 10 formula fields on one object, such as Line Item object and Proposal Line Item object, because it impacts CPQ performance. If you are creating a formula field on a line item only to access it in a Pricing Callback (PCB), you can use CPQ numeric expressions, so that you can avoid creating metadata fields. For more information, see CPQ for Administrators.
- Merge any workflow on a line item object into Pricing Callback if you have one.
- Ensure that the Product Attribute Value (PAV) fields are in sync on all PAV objects such as Proposal PAV, Asset PAV, Agreement PAV, and Order PAV.
- Consider creating skinny tables because they speed up the execution of SOQL queries.
- Avoid calling managed methods or APIs in a loop.
- Limit customization to the objects mentioned.
- Do not interpret the functional purpose of any standard field based on the value stored. When in doubt, file a support case for clarification.

- Creating triggers on common objects such as task, attachment, and so on should have checks to bypass in case of parent objects not relevant to that function. Check for this in preexisting code written before Conga is implemented.
- Only use Option Filter Callback when needed. Measure code and optimize custom code performance.
- Avoid using Process Builder and workflows on CPQ objects.
- Avoid the combination of Process Builder, workflows, and trigger within the same object. Instead, move the logic from Process builder and workflows to triggers. You must not define more than five triggers on one object.

Handling Governor Limits

This section describes some of the key exceptions observed in CPQ and describes how to deal with governor limits if the code encounters them after configuring CPQ. In general, you need to analyze logs and follow the best coding practices to resolve the errors. For more information, refer to Salesforce App Limits Cheatsheet.

Exception 1: System.LimitException: Apttus_Config2: Too many DML rows: 10001

This exception occurs when you try to perform DML operations on more than 10000 records at a time.

DML is different than SOQL. DML is the process typically used to insert, update, upsert, or delete records; SOQL is the query language used to return rows. Per the governor limits, the maximum number of records retrieved by SOQL queries is 50000. The maximum number of records processed as a result of DML statements, *Approval.process*, or *database.emptyRecycleBin* is 10000.

Sample Business Use Cases

- The Hierarchy View update job displays the *Too many queries* error, which has impacted the price book and display of products (products are not displayed under the correct categories on the catalog).
- When adding items to the option group *Destination Sets* in CPQ Admin Console in Production, an error is received for the Bundle LOGIQ S8 R3 (ULS_PCL_GI_0025).

Best Practices

- A separate class implementing the *Database.Batchable* interface allows CPQ to handle DML in batches of records.
- You can use <apex:actionPoller> in a Visualforce page.
- If the trigger has DML functionalities for many rows (more than 10000), it must be in Batch Apex.

Exception 2: Visualforce Remoting Exception: Apex heap size too large

Salesforce enforces an Apex Heap Size Limit of 6 MB for synchronous transactions and 12 MB for asynchronous transactions.

The *Apex heap size too large* exception occurs when too much data is being stored in memory during processing. The limit depends on the type of execution (for example, synchronous or asynchronous calls).

Sample Business Use Cases

When the constraint rule maintenance batch job is executed from the Maintenance tab, it runs erroneously. Especially for constraint rules with *Product Field Set* in **Product Scope** in constrain rule condition and criteria, a lot of memory is consumed to create Product Constraint View records. Eventually the batch crosses Salesforce governor limit of 12 MB memory usage per asynchronous transaction and skips creating views for some rules. The following exception occurs: *Apex heap size too large: 13808228*.

The Apex governor limits for heap size are as follows:

Total Heap Size	
Synchronous Limit	6 MB
Asynchronous Limit	12 MB
Email Services	32 MB

Best Practices

- Do not use class-level variables to store a large amount of data.

- Utilize SOQL For loops to iterate and process data from large queries.
- Construct methods and loops that allow variables to go out of scope as soon as they are no longer needed.
- Reduce the limits to the SOQLs and use transient keyword.

Exception 3: System.LimitException: Apex CPU time limit exceeded

Salesforce has a timeout limit for transactions based on CPU usage. If transactions consume too much CPU time, Salesforce terminates the execution of long-running transactions. The maximum CPU time on the Salesforce servers is 10000 milliseconds for synchronous transactions or 60000 milliseconds for asynchronous transactions.

Sample Business Use Cases

- Users could not see products in the cart after executing the Category Maintenance job.
- Users cannot execute the Update Product Constraints View and they receive an error - *Apex CPU Time Limit Exceeded*.
- Users encountered a "system.LimitException" error when they had more than 100000 products, defined a constraint rule with 98 condition products, and configured product scope as FieldSet. If there are a large number of fields defined in that Product Field Set, user encountered the error.

Best Practices

- Remove unnecessary code and loops.
- Use collections (Set, Hashmap, or List) efficiently.
- Avoid using a loop within another loop.
- SOQL query should be indexable.
- Avoid unnecessary re-initialization of variables.
- Use Aggregated SOQL (database operations are not counted in this limit, so avoid arithmetic operations in Apex).
- Check how much time workflow rules and process builders take.

In CPQ

- Define the following settings before running the Category Maintenance job.
  APTS_UpdateViewUseDmlLimit = true
  APTS_UpdateViewProductBatchSize = 50
- Optimize the constraint rule of *ProductScope = Product Field Set* to a lower number (this always breaks because of Salesforce limit).
- Avoid using many Custom Formula fields.
- To avoid "system.LimitException" error, create multiple constraint rules to reduce the number of condition products in a single constraint rule. Also, lower the number of fields in the Product Field Set.

Exception 4: System.LimitException: Too many query rows: 50001

There is a limit to the number of records that that can be retrieved by the SOQL query, which is 50000 records. The 50000 limit is an overall per-transaction limit and not a per-query limit. If you attempt to query the more than 50000 records, you will get an exception.

Sample Business Use Cases

- When running Category Maintenance, the following error occurs: *Apttus_Config2:Too many query rows: 50001*
- When trying to add more than 100 products from Catalog, CPQ displays a message: *Apttus_Config2:Too many query rows: 50001*
- When there are more than 400 lines added to the quote, the performance of CPQ is impacted.

Best Practices

- Use Batch Apex, in which the 50000 limit counts for each batch execution.
- These limits count for each Apex transaction. For Batch Apex, these limits are reset for each execution of a batch of records in the execute method.
- Limit the results by adding more criteria or using a LIMIT 50000 statement in SOQL.

In CPQ

- All constraint conditions or actions must have a narrow search.
- Avoid using *Product field set*.

Exception 5: System.LimitException: Too many SOQL queries: 101

This exception occurs when you exceed the SOQL queries governor limit. You can run up to a total of 100 SOQL queries in a single call or context.

Sample Business Use Cases

- If you finalize a cart that has 50 to 60 ramp lines, the cart stops responding.
- Each product added to the cart has a constraint rule. When you add 90 line items to the cart, constraint rules fail to work.

Best Practices

- Change the code by following the Apex Code best practices so that the number of SOQL queries triggered is less than 100.
- Change the context; use the @future annotation, which runs the code asynchronously.
- Ensure that you do not have a recursive loop calling a SOQL.
- Follow Apex Code key principals while writing triggers and bulk requests.
- Do not perform any DML or CRUD operation inside a For loop.

In CPQ

- Enable the **Run Misc Finalization Task in Async Mode** setting in Config System Properties (Developer Setting).

Guardrail for Using CPQ

CPQ is a complex application with configurability and extension capability. There are multiple objects interconnected that provide this capability. The relationship between objects is used to perform the compute-intensive configuration and pricing. This permutation-combination of various objects, their relationship, and data in each object contribute to the overall computation. It is difficult to arrive at a limit for each object that can be marked as a guardrail.

This section provides information on product guardrail parameters that contribute to CPQ performance.

Guidelines for CPQ

This section describes guidelines for CPQ configuration. CPQ performance is impacted when the guidelines are exceeded.

> These are general guidelines recommended by Conga. You must test in your org to determine an optimal limit for your implementation.

Description	Guideline
Maintenance Batch Job	
Number of **Product Hierarchy View** records for a single category, to execute Category Maintenance Batch job	30000
Cart	
Maximum number of line items to be selected for mass actions on the cart (using the **Max Allowed Lines For Mass Actions** setting) > The default value for selecting the line items for mass action is 10. When you enable the **Perform Mass Actions in Parallel** custom setting for selecting a large number of line items on the cart for mass action, the **Max Allowed Lines For Mass Actions** custom setting is overridden.	10
Number of columns in the Cart page	10
Rules	

Description	Guideline
Number of actions in constraint rules	10
Number of rules associated with products in any leaf category	1000
Number of conditions per rule	10
Number of rules for each category	1000
Number of rules for each bundle or options	10
Category	
Number of levels in a category hierarchy	3
Number of subcategories	10
Product Groups	
Product Group Members for each Price List	10000
Number of products for each product group	200
Number of Product Groups a product can be associated with	200
Option Groups/Bundle Structure	
Number of options for each option group	1000
Nested bundle structure ⓘ You can have a bundle > subbundle > option.	3

Description	Guideline
Attributes	
Number of attributes ⓘ CPQ has been enhanced to create and configure an unlimited number of attributes, overcoming the 800 field limit of Salesforce. The additional attributes can be used for Product Definitions, Product Configurations, and Attribute-based Rules. These attributes are created as an extension to the Product Attribute Value (PAV) object through a combination of master-detail and look-up relationships. The PAV object can be used for pricing whereas the PAV extension object cannot be used for pricing.	800
Number of attribute value matrix entries	10000
Pricing	
Number of price dimensions	4
Number of matrix entries for each price list item	1000
Number of price rules entries	500
Number of Price List setups in an org ⓘ When customer-specific pricing is required, use contractual price list instead of standard price list.	100
Approvals	

Description	Guideline
Number of approval steps ⓘ In Salesforce Admin setup, you can configure 100 approval steps (with a combination of standard steps and approval rules). In Salesforce runtime, you can submit, approve, or reject 350 approval requests simultaneously.	100

Recommendations to Improve CPQ Performance

This section describes the recommendations to improve the performance in different areas of CPQ.

Catalog Page

- If there is only one category, enable the **Hide top level category** setting to display subcategories as categories.
- Remove subbundles from parent bundles and treat subbundles as separate products.
- Use product structure (such as bundles), not rules (such as constraint, validation) wherever possible.
  For example, there is a requirement to include Product B if the user selects Product A. If you model A and B as standard items and a constraint rule to include B for A, this is not an optimal way of modeling. Instead, create bundles and auto-include Product B when the user selects bundle Product A.
- Minimize the number of options
  - Remove inactive options (end of life products) from bundles periodically.
  - Add products (options) to the price list in the region where they are sold.
  - Remove non-priced options from the bundle structure. Use attributes instead.
  - Avoid creating options for non-tangible products. Use attributes instead. For example, a country should be an attribute and not an option.
  - Set frequently used options as default.
- Avoid SKU proliferation (creating multiple instances of the options to handle quantity-based pricing).
- Leverage the **Price included bundle** option wherever applicable.
- Avoid creating duplicate data to maintain quality.
- Standardize naming conventions for rules and attributes.

- Use multiple option groups instead of maintaining all options under one option group.
- Create and manage *Public* favorite configurations effectively on the Catalog page. Use *Private* favorite configurations instead.
- Reuse and share configurations through favorites.

Cart Page

- Filtered lookup columns on the cart may impact CPQ performance.
- Manage the pricing callback code efficiency.
- Avoid defining multiple deal guidance setup for a single criteria.
- Avoid customizing a large cart.
- Verify if Option Groups with Max Cardinality = 1 can be used as picklist values.

Rules

- Keep the use of Product Visibility Rules to minimal to optimize catalog load performance. Create a separate price list or constraint rule.
- Create option specific rules. You can configure rules for common options without being specific to bundles.
- Define the criteria in a constraint rule condition wherever applicable, instead of constraint rule action criteria.
- Reduce the number of constraint rules by merging the constraint actions with similar conditions.
- Write your criteria on one condition. Do not repeat criteria on other conditions.
- Use client-side constraint rules to get better CPQ performance, that is, avoid server round-trip.
- Avoid using Formula fields in Constraint Rule Action criteria especially when you are using client-side constraint rules. Instead, use Conga numeric expressions.
- Avoid using formula fields on constraint rule conditions
- Avoid nested rule conditions.
- Avoid rules where the scope includes both bundles and options.
- Avoid creating redundant rules.
- Avoid creating multiple conditions with the same product when using AND association. The following table illustrates an example.

Constraint Rule Field	**Value**
Condition 1	*Product A*
Condition 2	*Product A with Attribute A*

Constraint Rule Field	Value
Association	*1 AND 2*

In the above-mentioned example, the condition is only fulfilled when both instances of the product are added to the cart.

- Do not create a server-side constraint rule with constraint condition using a formula field attribute from Product Attribute Rule Extension object.

Installed Product Page

- The number of renewals per trip is 20 by default. To avoid CPQ time out, reduce the number in the **Max Renews Per Trip** setting in Installed Products Settings.

Pricing on Cart

- Use Conga expressions wherever possible.
- Price Batch Size
    - The Price Batch Size must be high (higher the value faster the performance).
    - Define Price Batch Size correctly.
    - Measure and evaluate the highest possible value. Do not set up a value that can result in governance limits.
- Enable defer pricing only if the user experience is impacted (performance latency). Enabling defer pricing can increase the latency in the step when pricing is executed.
- Use the **Compute Totals In Separate Step** setting to avoid the CPU time limit issue to a large extent. This setting indicates whether the totaling should be performed in a separate step or if it should be combined with the base pricing step. This setting ensures that the totaling, which is dependent on the number of lines, is done as a separate remoting call.
- Manage deal guidance rules.
    - Consolidate rules wherever possible.
    - Simplify cross object fields.
    - Deal guidance limit varies based on the pricing complexity, number of lines, and deal guidance complexity.
- Manage the pricing callback code efficiency. For best practices for defining pricing callback, see Pricing Callback Classes.

Recommendations to Import Legacy Products into Conga CPQ

This topic describes best practices for migrating purchased product information from legacy CRM and ERP systems into Conga CPQ and for tracking subsequent renewals.

CPQ Implementation without Billing

To import assets in a pure CPQ implementation without Billing, it is best to create the asset line-item data directly in the Conga environment. This section describes best practices for importing legacy data, the asset types that may be imported, and those asset types' characteristics.

Maintaining Uniqueness within the Imported Data

- When importing data into the Asset Line Item object, the combination of line number (or primary line number) and business object ID must be unique. Business object ID can be quote ID, order ID, account ID, or agreement ID (if applicable). If the combination is not unique, the Installed Products page does not complete loading data.
- Maintaining line number uniqueness can be challenging when the process is batched for huge volumes. Implementation teams must ensure that the line number is sequenced programmatically to maintain uniqueness across multiple batches.

Standalone Assets

A standalone asset line item needs the following basic data elements to be transferred successfully to Conga CPQ. Implementation teams must ensure that this data is available within the source of the purchased products.

- The Sold To account associated with the standalone asset governs its display on the Installed Products page.
- The unit sale price of the asset with its corresponding selling frequency becomes the asset's base price for any future asset-based orders.
- For a recurring entity, the asset line item's term is important for prorating future asset-based orders and for determining the renewal start dates for manual and automated renewals.
- Set the **Inactive** flag on the asset line item to false. Any asset with Inactive=True is not visible on the Installed Products page.
- Set **Has Attributes** to True if the asset has attributes.

- Set Asset Status to Activated. The Installed Products page does not recognize statuses such transactional statuses as Amended or Renewed. An asset's status is either Activated or Cancelled.

Bundled Assets

In addition to the points specified for standalone assets, you must specify a bundle's relationships to its child assets (options). Your legacy systems may not operate with the concept of a bundled asset. Entities are related either using a contract or an order. Implementation teams must ensure that all the data is available and structured appropriately for the bundled asset to be used with asset-based ordering.

- Set **Has Options** to True on the asset.
- Set **Has Attributes** to True if the asset has attributes.
- Set **Has Attributes** to True at the bundle level even if the bundle has no attributes: options within the bundle may have attributes that drive quantity or cumulative range pricing. If **Has Attributes** is set to True, the reprice action does not reflect the change.
- Populate the parent bundle number (representing the primary line number of the immediate parent line in the hierarchy).
- Populate the parent asset ID (representing the primary line number of the immediate parent asset). This makes a difference when user has cloned options within the bundle structure.

Examples

The following examples show the differences in the population of the parent bundle ID and parent asset ID.

A simple bundle with one option

Asset Name	Charge Type	Selling Term	Start Date	End Date	Is Primary Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
Supernet Transportation Service	Subscription Fee	12	1/1/2021	12/31/2021	Yes	1			

Asset Name	Charge Type	Selling Term	Start Date	End Date	Is Primary Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
AAP Options	Subscription Fee	12	1/1/2021	12/31/2021	Yes	2	Supernet Transportation Service	1	Supernet Transportation Service

A bundle with two options where one of the options is a bundle

Asset Name	Charge Type	Selling Term	Start Date	End Date	Is Primary Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
Supernet Transportation Service	Subscription Fee	12	1/1/2021	12/31/2021	Yes	4			
AAP Options	Subscription Fee	12	1/1/2021	12/31/2021	Yes	7	Supernet Transportation Service	4	Supernet Transportation Service
VPN	Subscription Fee	12	1/1/2021	12/31/2021	Yes	5	Supernet Transportation Service	4	Supernet Transportation Service
Level 2 VPN	Subscription Fee	12	1/1/2021	12/31/2021	Yes	6	Supernet Transportation Service	5	VPN

A bundle with a nested bundled option that is cloned

Asset Name	Charge Type	Selling Term	Start Date	End Date	Is Primary Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
Supernet Transportation Service	Subscription Fee	12	1/1/2021	12/31/2021	Yes	1			
AAP Options	Subscription Fee	12	1/1/2021	12/31/2021	Yes	2	Supernet Transportation Service	1	Supernet Transportation Service
VPN	Subscription Fee	12	1/1/2021	12/31/2021	Yes	3	Supernet Transportation Service	1	Supernet Transportation Service
Level 2 VPN	Subscription Fee	12	1/1/2021	12/31/2021	Yes	4	Supernet Transportation Service	3	VPN
Level 2 VPN	Subscription Fee	12	1/1/2021	12/31/2021	Yes	5	Supernet Transportation Service	5	VPN
VPN	Subscription Fee	12	1/1/2021	12/31/2021	Yes	6	Supernet Transportation Service	1	Supernet Transportation Service

Ramped Assets

To persist discrete data points associated with each period for billing and revenue recognition in a Quote-to-Cash (Q2C) environment, CPQ creates one asset for each period

while processing a ramped (multi-year) deal. The following examples show the structure of such assets:

- Ramped assets are all connected using a primary ramp asset, acting as a loosely-coupled bundle that holds the assets for all periods.
- CPQ populates the bundle asset line item.
- You must set the **Is Primary Ramp Line** flag to True for the first period.

Examples

The following examples show the differences in the population of the parent bundle ID and parent asset ID.

A standalone ramped asset

Asset Name	Charge Type	Start Date	End Date	Is Primary Line	Is Primary Ramp Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
CPQ	Subscription Fee	1/1/2017	12/31/2017	Yes	Yes	1			
CPQ	Subscription Fee	1/1/2018	12/31/2018			1	CPQ		
CPQ	Subscription Fee	1/1/2019	12/31/2019			1	CPQ		

A ramped asset with bundle level ramps that cascade to the options

Asset Name	Charge Type	Start Date	End Date	Is Primary Line	Is Primary Ramp Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
Apttus	Subscription Fee	1/1/2017	12/31/2017	Yes	Yes	1			

Asset Name	Charge Type	Start Date	End Date	Is Primary Line	Is Primary Ramp Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
Apttus	Subscription Fee	1/1/2018	12/31/2018			1	Apttus		
Apttus	Subscription Fee	1/1/2019	12/31/2019			1	Apttus		
CPQ	Subscription Fee	1/1/2017	12/31/2017	Yes	Yes	2	Apttus	1	Apttus
CPQ	Subscription Fee	1/1/2018	12/31/2018			2	Apttus	1	Apttus
CPQ	Subscription Fee	1/1/2019	12/31/2019			2	Apttus	1	Apttus
ABO	Subscription Fee	1/1/2017	12/31/2017	Yes	Yes	3	Apttus	1	Apttus
ABO	Subscription Fee	1/1/2018	12/31/2018			3	Apttus	1	Apttus
ABO	Subscription Fee	1/1/2019	12/31/2019			3	Apttus	1	Apttus
Billing	Subscription Fee	1/1/2017	12/31/2017	Yes	Yes	4	Apttus	1	Apttus

Asset Name	Charge Type	Start Date	End Date	Is Primary Line	Is Primary Ramp Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
Billing	Subscription Fee	1/1/2018	12/31/2018			4	Apttus	1	Apttus
Billing	Subscription Fee	1/1/2019	12/31/2019			4	Apttus	1	Apttus

A ramped asset with option-level ramps

Asset Name	Charge Type	Start Date	End Date	Is Primary Line	Is Primary Ramp Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
Apttus-Service Included	Standard Price	1/1/2017	12/31/2017	Yes	Yes	1			
Apttus-Service Included	Subscription Fee	1/1/2017	12/31/2019			1	Apttus-Service Included		
CPQ	Subscription Fee	1/1/2017	12/31/2017	Yes	Yes	2	Apttus-Service Included	1	Apttus-Service Included
CPQ	Subscription Fee	1/1/2018	12/31/2018			2	Apttus-Service Included	1	Apttus-Service Included
CPQ	Subscription Fee	1/1/2019	12/31/2019			2	Apttus-Service Included	1	Apttus-Service Included

Asset Name	Charge Type	Start Date	End Date	Is Primary Line	Is Primary Ramp Line	Primary Line Number	Bundle Asset Line Item	Parent Bundle Number	Parent Asset Line Item
ABO	Subscription Fee	1/1/2017	12/31/2017	Yes	Yes	3	Apttus-Service Included	1	Apttus-Service Included
ABO	Subscription Fee	1/1/2018	12/31/2018			3	Apttus-Service Included	1	Apttus-Service Included
ABO	Subscription Fee	1/1/2019	12/31/2019			3	Apttus-Service Included	1	Apttus-Service Included

Linkage to Child Objects

You can link an imported asset line item to its attribute detail using the asset attribute value object. You can link it to its usage detail using the asset usage tier values if applicable.

Quote-to-Cash Implementation with Standard Billing

For a Q2C implementation where there is a need to generate billing for assets, the recommended strategy is to create the order and order lines. Activation of the order and order lines will then generate the assets and the billing. The following are the best practices in those scenarios:

- Specify the billing frequency and billing method for the price list item associated with the order line item. If you do not specify, order activation will fail.
- Set the status of the order header and order lines to Pending Activation.
- Set the Line Status of the order line items to New.

Quote-to-Cash Implementation with Custom Billing

For a Q2C implementation to generate custom billing (using a billing plan) for assets, create the proposal and proposal line items, attach a billing plan to the proposal header, accept the quote, and activate the corresponding order. In such a scenario:

- Follow the standard process to create or import a proposal.
- Auto-activate orders by passing either the **Ready for Activation Date** from the proposal (if the billing is aligned to this date) or by setting the **Auto Activate Order** flag to True.
- Do not create the renewal suite on order activation while going through this process.

Renewal Pipeline Generation for Imported Assets

To generate the renewal pipeline for legacy assets, use the OnDemand mode in conjunction with the lead time.

- To group the generated renewal quotes by quote ID, use the *Apttus_QPConfig ProposalId r.Name* grouping field.
- Do not generate renewal suites for legacy assets on order activation in a batch process. This overwhelms CPQ resources, violates the governor limits, and causes unpredictable behavior.

Using a Batch Process to Create Imported Assets

- Asset import through an order involves calling several data manipulation languages (DMLs) and APIs. Executing too many orders in one batch can result in a *Too many SOQL queries: 201 error*.
- You must ensure that the batch size is optimized to prevent errors based on the size of your order and the size of each line item.

Factors of Master Data that Contribute to Performance

The CPQ Master data model represents the setup and administration of configurations, rules, and pricing of the system. This topic lists the factors of master data that contribute to the performance of CPQ. A combination of these factors, when implemented in a specific way, can impact CPQ performance.

- Number of Products
- Number of Price Lists
- Number of Price List Item
- Number of Price List Item per Price List
- Related Price Lists
- Charge Types per Price List
- Number of Price Matrices
- Number of Price Matrices per
- Price List Item
- Number of dimensions per Price Matrix
- Number of Price Rules
- Rule sets
- Rule entries
- Number of Catalog levels (Hierarchy)
- Number of Items in a catalog
- Number of Options in a Option Group
- Number of Option Groups in a bundle
- Number of Options in a Bundle
- Number of level of bundle or subbundle
- Number of Attributes
- Number of Pricing Attributes
- Product Attribute Rules
- Number of Numeric Expressions (for each product)
- Number of Constraint Rules
- Number of Constraint Rule Conditions per
- Constraint Rule
- Number of Constraint Rule Actions per Constraint Rule
- Deal Guidance
- Deal Guidance Entries
- Attribute Value Matrix
- Number of CPQ Search Filters
- Line Level Approvals

Factors of Transactional Data that Contribute to Performance

The CPQ Transactional data model represents the configuration, rules, and pricing that is applied for each transaction and how that impacts the performance at runtime. The data

setup of the master data drives the behavior for each transaction. This topic lists the factors of transactional data that contribute to the performance of CPQ. A combination of these factors, when implemented in a specific way, can impact CPQ performance.

- Number of Product Configuration per Quote
- Number of line items per Product Configuration
- Number of charge types applied per Product
- Number of Attributes per bundle/Option
- Attribute Rules per Bundle/Option
- Number of constraint rules applied per cart
- Number of Product Groups one Option/Bundle is associated with
- Number of columns in a cart
- Number of Approval Processes (line Level)
- Number of Summary Groups in cart
- Related Price Lists
- Promotions related to line items
- Deal guidance
- Adjustment Line Item bucketing

CPQ Features by Release

Review the latest CPQ Features by Release document.

- Features by Release

Features by Release

This document contains an overview of features introduced in each major release of Conga CPQ. For more information, see Conga CPQ Features by Release.

Conga Customer Community & Learning Center Resources

The Conga Customer Community is your one-stop shop for success!

After registering as a new member, you'll gain access to a wide variety of resources, from a personalized onboarding checklist to free expert-led webinars, to our thought-leadership blog! It's also your portal to manage your Conga account, access the Install Center, and submit support tickets.

You can also access the Conga Learning Center. All customers get access to a limited catalog of getting started courses. Consider upgrading to the Conga Learning Pass to unlock the premium training subscription.

Ready to get started?

Log into the Conga Customer Community with your credentials.

Not yet registered? No problem. Set up an account to receive your login credentials via our registration page.

After you log in, there are 2 ways to access the Conga Learning Center:

- On the main page, click the Learning Center tile.
- Navigate to the "Resources" dropdown menu at the top, then click the "Learning Center" option.