Approvals

Table of Contents

Conga Approvals enables you to trigger an approval request for any object and send an email notification to the concerned stakeholders.

Approvals Release Notes

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

- Winter20.09.17 Release Notes
- Winter20.08.13 Release Notes
- Winter20.07.30 Release Notes
- Winter20.07.16 Release Notes
- Winter20.06.04 Release Notes
- Winter20.05.10 Release Notes
- Winter20.04.23 Release Notes
- Winter20.04.09 Release Notes
- Winter20.03.12 Release Notes
- Winter20.01.29 Release Notes
- Winter20.12.11 Release Notes
- Winter '20 Release Notes

Winter20.09.17 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.09.17 Release:

> For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Fixed Issues: Lists customer-reported issues that are fixed in this release or known issues fixed from previous releases
- Known Issues: Lists known issues that are applicable in this release

> This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

> **You must have Community Portal login credentials to be able to download these packages.**

Product	**Latest Certified Version** ***Version Name | Version Number***
Conga Approvals **(New)**	11.2.0190.29 | 11.190.29
Conga Grid (Required to use Approval Center) This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3

Product	Latest Certified Version *Version Name | Version Number*
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management **(New)**	11.2.0582.23 | 11.582.23
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.72 | 12.1839.72
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

ⓘ Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

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

This release does not contain any data model changes for Conga Approvals.

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
00325679	AWA-3433	Upon approving a single Approval Request, all the Approval Request with Assigned status are getting approved even though the approver has selected a single Approval Request. This happens only when the following criteria is met: • When multiple requests are in assigned status • The approver's profile is added to the APTS_AdminProfiles admin entry • No requests are assigned to the approver
00117542	AWA-3419	Approvals requests are not displaying in the correct ascending order while previewing them on the *Apttus_Approval__PreviewSubmitApprovals* visualforce page.

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

Conga Internal ID	Description
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.

Conga Internal ID	Description
AWA-3043	When you submit a large number of approval requests, the APEX CPU time limit exceeded error appears.

Known Issues

There are no known issues in this release.

DOC ID: IWAWIN20PRN20210917

Winter20.08.13 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.08.13 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

⚠ **You must have Community Portal login credentials to be able to download these packages.**

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.27 | 11.190.27
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8

Product	Latest Certified Version *Version Name | Version Number*
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.72 | 12.1839.72
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

ⓘ Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.

System Requirement	Minimum Supported Version
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

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

This release does not contain any data model changes for Conga Approvals.

Resolved Issues

The following table lists the issue resolved in this release.

Case Number	Conga Internal ID	Description
N/A	AWA-3438	Submitting for Approval displayed permissions error.
00117995	AWA-3434	After reassigning an approval request from the My Approvals page, the Return, Approve, Reject, and Save Comment buttons were visible. Only the Return button should be visible.
00115312	AWA-3363	Even if you had set Consolidate Approval as true, approval requests were not submitted at the same time. This issue was observed in approval processes containing child processes and both the approval requests were submitted to the same user.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.07.30 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.07.30 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

You must have Community Portal login credentials to be able to download these packages.

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.23 | 11.190.23 ⚠ This release note reflects the change in version number. Only internal defects were resolved in this release.
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8

Product	Latest Certified Version *Version Name | Version Number*
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.72 | 12.1839.72
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

ⓘ Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.

System Requirement	Minimum Supported Version
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

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

This release does not contain any data model changes for Conga Approvals.

Resolved Issues

There are no resolved issues in this release.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.07.16 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.07.16 Release:

> For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

> This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

> **You must have Community Portal login credentials to be able to download these packages.**

Product	**Latest Certified Version** ***Version Name | Version Number***
Conga Approvals **(New)**	11.2.0190.22 | 11.190.22
Conga Grid (Required to use Approval Center) This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3

Product	Latest Certified Version *Version Name | Version Number*
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.72 | 12.1839.72
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

ⓘ Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

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

This release does not contain any data model changes for Conga Approvals.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00117348	AWA-3405	An incorrect validation rule error message was displayed while approving or rejecting an Approval Request that satisfied the Salesforce Validation Rule.
00116257	AWA-3396	When an Approval Request was escalated, it was assigned to the Backup Admin User instead of the Queue.
00111426	AWA-3192	Even though *Skip Unresolved Assignee* flag was set to True in the rule, the submitter was not able to skip unresolved assignee from the subprocess or childprocess.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.

Conga Internal ID	Description
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.06.04 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.06.04 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

⚠ You must have Community Portal login credentials to be able to download these packages.

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.17 | 11.190.17
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.72 | 12.1839.72

Product	Latest Certified Version *Version Name | Version Number*
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.

System Requirement	Minimum Supported Version
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

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.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00115546	AWA-3333	Conga Approvals did not preview and trigger eligible approval requests for an approval rule entry containing more than 2000 entries. This issue is now resolved. You can now preview and trigger eligible approval requests for an approval rule entry containing up to **4000** entries.
00113073	AWA-3334	The attachment file got deleted if you had uploaded it from your local storage.
00113558	AWA-3321	After you click **Submit for Approvals** button on the cart page, Conga Approvals did not load the Cart Approvals UI.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.05.10 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.05.10 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

You must have Community Portal login credentials to be able to download these packages.

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.15 | 11.190.15
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.72 | 12.1839.72
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7

Product	Latest Certified Version *Version Name | Version Number*
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.

System Requirement	Minimum Supported Version
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

New Features

There are no new features in this release. Refer to the 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.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00112537	AWA-3257 AWA-3194	On My Approvals UI, you could click **Approve, Reject, and Save Comment** button even if no approval requests were selected. This issue is now resolved. Conga Approvals now displays a warning message instructing you to select an approval request.
N/A	AWA-3256	For rule-based approval flow, the approval email did not contain attachments.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.04.23 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.04.23 Release:

> For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features

- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

> This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

> **You must have Community Portal login credentials to be able to download these packages.**

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.12 | 11.190.12
Conga Grid (Required to use Approval Center) > This package is not available on Conga Install Center. > • Install in Production from here. > • Install in Sandbox from here.	2.101 | 2.101

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals Center (Required to use Approval Center) This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing **(New)** (Required if you are using CPQ Approvals)	12.2.1839.72 | 12.1839.72
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

> ⓘ Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

New Features

There are no new features in this release. Refer to the 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.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00113016	AWA-3198	Approval Requests were not auto-selected on the My Approvals UI.
00113582	AWA-3222	*My Approvals UI* did not display the count of total Approval Requests.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.

Conga Internal ID	Description
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.04.09 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.04.09 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

⚠ You must have Community Portal login credentials to be able to download these packages.

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.11 | 11.190.11
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library **(New)** (Pre-requisite for Conga CLM)	1.2.123.6 | 1.123.6
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.35 | 12.1839.35

Product	Latest Certified Version *Version Name | Version Number*
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.

System Requirement	Minimum Supported Version
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

New Features

There are no new features in this release. Refer to the 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.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00111601	AWA-3146	Conga Approvals did not assign your approval requests to backup approver if your approval request contained sequential approvals and you had set **Consolidate Approvals** as *True.*
00111037	AWA-3182	If the approval request was approved by someone other than the approver, Conga Approvals did not save approval comments.

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

Conga Internal ID	Description
AWA-2814	Conga Approvals sometimes displayed *Unable_To_Lock_Row* exception, while submitting approval requests.
AWA-2850	Agreement Term Exception did not work with Assignee Type as Rule.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-3033	If you navigate back to Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.

Conga Internal ID	Description
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.03.12 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.03.12 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

⚠ You must have Community Portal login credentials to be able to download these packages.

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.10 | 11.190.10
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library **(New)** (Pre-requisite for Conga CLM)	1.2.123.5 | 1.123.5
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839.35 | 12.1839.35

Product	Latest Certified Version *Version Name | Version Number*
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.

System Requirement	Minimum Supported Version
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

New Features

There are no new features in this release. Refer to the 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.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00109943	AWA-3090	Preview Approvals screen displayed attachments in an incorrect order.
00111760	AWA-3152	Approvals emails sent to Ad-Hoc Approvers did not contain attachments. This issue was observed for Ad-Hoc Approvers added through the **My Approvals** page.
00111601	AWA-3195	Conga Approvals did not assign your approval requests to backup approver if your approval request contained sequential approvals and you had set **Consolidate Approvals** as *True.*

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-2814	When you submit approval requests, Conga Approvals sometimes give Unable_To_Lock_Row exception.
AWA-2850	Agreement Term Exception does not work with Assignee Type as Rule.
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.01.29 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.01.29 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

You must have Community Portal login credentials to be able to download these packages.

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190.7 | 11.190.7
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123 | 1.123
Conga Contract Lifecycle Management	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing **(New)** (Required if you are using CPQ Approvals)	12.2.1839.35 | 12.1839.35
Conga Quote Configuration Integration **(New)** (Required if you are using CPQ and Proposal management)	12.2.0344.7 | 12.344.7

Product	Latest Certified Version *Version Name | Version Number*
Conga CLM Configuration Integration **(New)** (Required if you are using CLM)	12.2.0146.7 | 12.146.7
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.

System Requirement	Minimum Supported Version
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) Conga recommends the usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

New Features

There are no new features in this release. Refer to the 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.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00101589	AWA-2767/ AWA-3144	If you were logged into Salesforce Org as a Delegate Approver, you did not see all the approver action buttons on the Approval Visualforce page.

Case Number	Conga Internal ID	Description
00102799	AWA-2969	You could not access approval process records on the Opportunity object if you had enabled **Einstein Opportunity Insights** in your Salesforce Org.
00110011	AWA-3088	Before submitting an approval request, you could not see Backup or Delegate Approver names on the *PreviewSubmitApprovals* Visualforce page.
00104398	AWA-3134	Conga Approvals sent reminder emails even if you had recalled the approval request. This issue was observed if you had set Bypass Sharing as false in Approval System Properties.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-2814	When you submit approval requests, Conga Approvals sometimes give Unable_To_Lock_Row exception.
AWA-2850	Agreement Term Exception does not work with Assignee Type as Rule.
AWA-3033	If you navigate back to Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter20.12.11 Release Notes

Conga Release Notes contain the following information about Approvals Winter20.12.11 Release:

For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

You must have Community Portal login credentials to be able to download these packages.

Product	**Latest Certified Version** ***Version Name | Version Number***
Conga Approvals **(New)**	11.2.0190.4 | 11.190.4

Product	Latest Certified Version *Version Name | Version Number*
Conga Grid (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center (Required to use Approval Center) ⓘ This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3
Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library (Pre-requisite for Conga CLM)	1.2.123 | 1.123
Conga Contract Lifecycle Management **(New)**	11.2.0582.8 | 11.582.8
Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing **(New)** (Required if you are using CPQ Approvals)	12.2.1839.6 | 12.1839.6
Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344 | 12.344

Product	Latest Certified Version *Version Name | Version Number*
Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146 | 12.146
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.

System Requirement	Minimum Supported Version
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

New Features

There are no new features in this release. Refer to the 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.

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00104637	AWA-3014	Users CCed in the approval request email could incorrectly approve an approval request by replying to that email.

Case Number	Conga Internal ID	Description
00108218	AWA-3035	When you previewed an approval request, Conga Approvals created unnecessary records in the Agreement Approval History.
00108305	AWA-3041	Approval comments were not displayed properly if the comment contained special characters.
00109014	AWA-3063	If you had set Consolidate Notifications as true, all the parallel approval requests were not assigned to the backup approver. Only the first approval request was assigned to the backup approver.
00108857 / 00109540	AWA-3064/ AWA-3080	If you submitted multiline comments, Conga Approvals did not retain line breaks.
00109230	AWA-3068	If you had set Consolidate Notifications as true, all the sequential approval requests were not assigned to the backup approver. Only the last approval request was assigned to the backup approver.
00109257	AWA-3072	If you had set Consolidate Notifications as true, all the approval requests were not assigned to the delegated approver. Only one approval request was assigned to the delegated approver.
00109286	AWA-3073	Conga Approvals gave Attempt to Dereference a null error if you tried to approve an approval request from a related list. This issue was observed only for approval requests created before Summer 20 release.
00108619	AWA-3074	You could not recall an approval request if it contained Remainders from other Approvers. This issue was observed if you had set **Bypass Sharing** as *True* in Approval System Properties.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-2814	When you submit approval requests, Conga Approvals sometimes give Unable_To_Lock_Row exception.
AWA-2850	Agreement Term Exception does not work with Assignee Type as Rule.
AWA-3033	If you navigate back to the Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

Winter '20 Release Notes

Conga Release Notes contain the following information about Approvals Winter '20 Release:

> For more information on new features, enhancements, and document improvements refer to What's new in Approvals Documentation.

- Packages: Lists packages that are required to upgrade to this release of the product
- System Requirements and Supported Platforms: Lists requirements and recommendations for installing this release
- New Features: Provides high-level descriptions of new features introduced in this release, with links to more detailed information
- Enhancements: Provides high-level descriptions of enhancements to existing features
- Data Model Changes: Lists changes to the data model
- Resolved Issues: Lists customer-reported issues that are resolved in this release or known issues resolved from previous releases
- Known Issues: Lists known issues that are applicable in this release

> This documentation may contain descriptions of software features that are optional and for which you may not have purchased a license. As a result, your specific software solution and/or implementation may differ from those described in this document. Please contact your CSM or AE for information on your specific features and licensing.

Packages

The following packages and dependent packages are required to upgrade to this release to utilize all the new features of this release. These are the *minimum* required versions; later versions are also supported. Separate prerequisites for each feature can be found in the respective guides. The packages marked as **(New)** are new packages in this release.

> **You must have Community Portal login credentials to be able to download these packages.**

Product	Latest Certified Version *Version Name | Version Number*
Conga Approvals **(New)**	11.2.0190 | 11.190
Conga Grid **(New)** (Required to use Approval Center) This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	2.101 | 2.101
Conga Approvals Center **(New)** (Required to use Approval Center) This package is not available on Conga Install Center. • Install in Production from here. • Install in Sandbox from here.	11.3.0 | 11.3

Product	Latest Certified Version *Version Name | Version Number*
Conga CPQ Approvals **(New)** (Required if you are using CPQ)	11.2.0019 | 11.19
Conga Base Library **(New)** (Pre-requisite for Conga CLM)	1.2.123 | 1.123
Conga Contract Lifecycle Management **(New)**	11.2.0582 | 11.582
Conga Quote Management **(New)** (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
Conga Configuration & Pricing **(New)** (Required if you are using CPQ Approvals)	12.2.1839 | 12.1839
Conga Quote Configuration Integration **(New)** (Required if you are using CPQ and Proposal management)	12.2.0344 | 12.344
Conga CLM Configuration Integration **(New)** (Required if you are using CLM)	12.2.0146 | 12.146
Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
Conga Custom Approvals (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

ⓘ Before installing or upgrading Conga managed packages, you must review the dependency matrix for each managed package. You can refer to the package dependency matrix at Managed Packages Dependency Matrix.

System Requirements and Supported Platforms

The following table lists the minimum requirements for installing and using Approvals.

System Requirement	Minimum Supported Version
Operating System	Standard Salesforce.com requirements. See Salesforce PDF.
Browser	Conga supports the following browsers: • Microsoft Edge Chromium • Google Chrome Conga recommends the latest version of the browser for the best performance. Conga also recommends its customers to use Microsoft Edge Chromium as the browser of choice. ⚠ Internet Explorer is not supported.
Microsoft Office	• Microsoft Word 2007 with Service Pack 2 plus hotfix package 976477 (Word 2007 version number has to be greater than 12.0.6520.4999) • Microsoft Word 2010 (32 bit & 64 bit) ⓘ Conga recommends usage of the latest Microsoft Office Suite for the best performance.

Upgrade Notes

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

New Features

The following features are new to Approvals in this release.

Approval Center

Conga Approvals introduces the Approval Center, which provides a holistic view of all your approval records. Using the Approval Center, you can approve or reject approval requests, take ownership of approval requests, and create custom views.

Get Started

For more information, see Approval Center.

System Property to Submit Approval Requests in Asynchronous Mode

Conga Approvals introduces a system property, **Process Submit In Async Mode**. When you select this property, the approval requests are submitted in an asynchronous mode. This setting is useful when you have a large number of approval records to submit.

Get Started

For more information, see Approval System Properties.

Enhancements

The following enhancement is new in this release.

Deprecated: Sync Submit Request Threshold System Property

The Approval System Property, **Sync Submit Request Threshold** is deprecated from Winter '20 release. Conga recommends that you use **Process Submit in Async Mode** system property instead.

Get Started

For more information, see Approval 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/ Changes
Approval Process Instance		Represents an instance of a single, end-to-end approval process.		Changed
	Attachment Ids	List of attachment IDs to be sent with email notifications.	System	New
Approval Requests		Designates an approval request created from an approval process.		Changed
	Delegate Approvers (Names)	Names of Delegate Approvers.	System	New
Approval Request History		Designates history for an approval request created from an approval process		Changed
	Delegate Approvers (Names)	Names of Delegate Approvers.	System	New

Resolved Issues

The following table lists the issues resolved in this release.

Case Number	Conga Internal ID	Description
00101084	AWA-2378	Conga Approvals incorrectly assigned the approval request to backup admin user instead of custom queue. This issue was observed for Line Item level approval requests.
00105488	AWA-2859	For Adhoc Approval processes, you could not see Document Versions while submitting an approval request with attachments.
00103309	AWA-2967	The content on My Approvals and Approvals Rule Manage UI overflowed and did not fit into a single view. This issue was observed when you had long textual comments with multiple approvers.
00107256	AWA-2993	For rule-based approvals, the approval email did not contain the attachment file. This issue was observed if you had set **Submission Comments Enabled** as *true*.
00107428	AWA-2994	If you added special characters in comments when submitting an approval request, the special characters were displayed incorrectly when you click Preview and Submit.
00107853	AWA-3000	The content on My Approvals and Approvals Rule Manage UI overflowed and did not fit into a single view. The comment text was also replaced by '</br>'. This issue was observed when you had long textual comments with multiple approvers.
00107540	AWA-3034	If you tried to reject approval requests asynchronously, Conga Approvals gave an Exception error. This happened because cancellation approval requests were also processed asynchronously. **Resolution:** You can now reject multiple approval requests asynchronously.

Known Issues

The following table lists the known issues in this release.

Conga Internal ID	Description
AWA-2814	When you submit approval requests, Conga Approvals sometimes give Unable_To_Lock_Row exception.
AWA-2850	Agreement Term Exception does not work with Assignee Type as Rule.
AWA-3033	If you navigate back to Backup/Delegate Approver screen after performing any action, Conga Approvals displays a blank screen.
AWA-3043	When you submit a large number of approval requests, Conga Approvals sometimes give APEX CPU Time Limit Exceeded exception.

About Conga Approvals

Conga Approvals enables you to trigger an approval request for any object and send an email notification to the concerned stakeholders. Using Approvals, you can set an approval request on an object header, line item, or both. After an approval request is triggered, an approver can approve or reject a request via email or a Salesforce org. The Home page or the My Approvals page displays all the available approval requests.

Conga Approvals allows an administrator to perform the following administrative tasks:

- Install and configure Conga Approvals
- Set up a user to approve, submit a request with or without attachments, reassigning an approval request, auto-escalate approval requests
- Setup Approval Center for your users
- Configure auto-reapprovals and consolidate approvals
- Set up backup or delegate approvers including out-of-office users
- Set up a backup administrator
- Add an Ad Hoc Approver
- Configure an approval process
- Configure Approvals for custom objects
- Configuring Approvals for Agreement and related Objects
- Change the default approvals process user interface
- Change the assignee search page results list
- Set up and enable approvals from communities
- Change approval rule step default behavior
- Set up custom assignee value expression objects
- Create approval matrices
- Adding related fields to a criteria
- Configure the cart approvals page and apex class
- Set up custom assignee (user, queue, and role)

Conga Approvals allows a user to perform the following administrative tasks:

- Install Conga Approvals
- Approve, submit a request with or without attachments
- Reassign an approval request
- Auto-escalate approval requests
- Set up an approval process

Key Terminology

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

Term	Description
Approval Matrix	Approval matrix determines the assignees based on given criteria. For example, approver authority may change based on the discount being offered on a sales contract. So different levels of approvers can be assigned based on approval authority on a given item.
Approval Process	The process definition describes approval processes within your organization. You can define approval processes for Opportunity, Agreement, or Term Exceptions objects. You have to define entry criteria, initial submission actions, step groups (steps, step filters), and final actions within an approval process.
Approval Rule (Rule)	Approval rule uniquely identifies an approver based on combinations of various logical conditions per business process and policies. For example, an agreement needs to be sent for approval to CFO if the contract value is above $1M OR (geography is emerging markets AND contract value is above $250K).
Assignee Type	The system supports the following: Approval Matrix, Auto Approval, Custom Queue, Custom Role, Custom Rule, Related User, Custom User, Queue, Role, User
Assignment Email Template	This email template is used when an item is reassigned to another user. When an item is assigned to a user for approval, he/she always has an option to reassign the item to another user.
Auto Approval	This Assignee Type can be selected for Approval Rule Entry Criteria when you want the item in an approval process to be automatically progressed. For this to work, in an Approval Process step that references the Approval Rule using Auto Approval, the Assigned Approver > Assignee Type must be Subprocess.

<table>
<tr><th>Term</th><th>Description</th></tr>
<tr><td>Backup Approvers</td><td>When a user is out of the office, all approvals can be automatically assigned to backup approvers. The user may choose to reassign in-process approval requests to backup approvers as well.</td></tr>
<tr><td>Consolidated Approvals</td><td>If some approval process results in a situation where the same user is required to approve the same object record/line items, these multiple approvals can be consolidated in one approval. In other words, the approver has to approve or reject it only once to approve or reject all of their approval requests for a single approval process. Consider the following approval process with three steps:
<table>
<tr><th>Approval Step</th><th>Approver 1</th><th>Approver 2</th><th>Approver 3</th></tr>
<tr><td>Sales Level 1</td><td>John</td><td>Mary</td><td>Dave</td></tr>
<tr><td>Sales Level 2 (Dependent on 1)</td><td>James</td><td>John</td><td>Suze</td></tr>
<tr><td>Executive Sign-off</td><td>Art</td><td>-</td><td>-</td></tr>
</table>
When the object record is submitted for approval and the approvers are selected based on the criteria in the approval process, the approver John is included in two steps, including one dependent step. After Mary and Dave have approved their requests for the Sales Level 1 step, John's approval request in the Sales Level 2 step will be Assigned to him, while his approval request in Sales Level 1 is on Hold. This enables him to approve both of his approval requests via a single action.</td></tr>
<tr><td>Custom Assignee</td><td>The custom assignee option allows you to configure custom code to be called to evaluate an assignee. If none of the other options are meeting the assignment requirements, a custom code can be developed to evaluate the approver based on the given criteria.</td></tr>
</table>

Term	Description
Custom Queue	This assignee type points to a field that contains a queue name in any custom object with a filter to narrow it down to a single row. There are two ways to specify a filter: 1. Select the row where the 'Region__c' field has a value of 'Asia'. 2. Select the row where the 'Region__c' field has a value that equals the value of the 'MyRegion__c' field in the associated business object.
Custom Role	This points to a field that contains a role name in any custom object with a filter to narrow down to a single row. For example: Custom_Object__c.Approver_RoleName__c
Custom User	This points to a user lookup reference field in any custom object with a filter to narrow down to a single row. The custom object may have filter criteria enclosed in parenthesis like the Region filter in this example: Custom_Object__c(Region__c = 'Asia').Approver_UserId__c
Entry Criteria	You can define certain logical conditions as part of entry criteria for an approval process. If these conditions evaluate true, the approval process is evaluated. This criteria allows you to make sure that all agreements (or opportunities) do not process through the given approval process.
Reassign	When an item is assigned to a user for approval, he/she always has an option to reassign the item to another user.
Step	A step is a business activity of approval that is performed by a user to approve or reject the record.
Step Assignee Type	The type of assignee to which the record will be assigned for approval.
User	This assignee type refers to a named user of the system. Use this assignee type when you want to dynamically assign an approval request to a user.

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

What's New in Approvals Documentation

The following section lists changes in the documentation to support each release.

Winter '20

Document	Topic	Description
Winter '20	Approval System Properties	Modified topic. Added information about the following system properties: • Process Submit in Async Mode • Approval Center Objects
	Updating App Name and App Logo	New topic for this release.
	Permission Sets for Accessing Approval Center	New topic. New feature for this release.
	Adding Approval Status to Objects	Modified topic. Added information about creating a lookup relationship of the custom object with the approval request object.
	Approval Center	New topic. New feature for this release.
	Submitting for Approvals with Attachments	New topic. New API introduced in this release.

Summer 2020

Document	Topic	Description
Summer 2020 Rev A	Approval System Properties	Modified topic. Added information about Queueable Apex Batch Size system property.

Document	Topic	Description
Summer 2020	Submitting Version Specific Agreement Attachments	New topic. New feature for this release.
	Approval System Properties	Modified topic. Added the following new system properties: • Sync Submit Request Threshold • Sync Approve Request Threshold • Email Sender Org Wide Display Name Used • Display Name
	Submitting a Request with Attachments	Modified topic. Added a note explaining attachment file size limits for the download link.
	Approvals Custom Config Settings	Modified topic. Added the new custom config setting, Show All Document Version Details.
	Approvals and My Approval Pages	Modified topic. Updated the screenshots to show the sequence column in the latest UI.
	Asynchronous Processing of Approvals Records	New topic. New feature for this release.
	Upgrading Approvals	Modified topic. Added information about upgrading to the Summer 2020 release.
	Submitting Bulk Approval Requests	New topic. New API introduced in this release.
	Recalling Bulk Approval Requests	New topic. New API introduced in this release.
	Submitting Approvals Asynchronously	New topic. New API introduced in this release.

Document	Topic	Description
	Create a Runtime Adhoc Approval Process	Updated topic to include API signature.
	Update a Runtime Adhoc Approval Process	Updated topic to include API signature.
	Retrieve a Runtime Adhoc Approval Process	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Delete a Runtime Adhoc Approval Process	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Add Comment to a Runtime Adhoc Approval Process	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Add Attachments to a Runtime Adhoc Approval Process	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Delete Attachments to a Runtime Adhoc Approval Process	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Use Case and Sample Code for Adhoc Approval Runtime Process APIs	Modified topic. Added use-case and API flow for ad-hoc approval process.
	Delete An Adhoc Approval Step	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Submit For Approvals With Comments Using An Adhoc Approval Specification	Updated topic to include API signature and SOAP Response/ Request XML for API integration.

Document	Topic	Description
	Submit For Approvals Using An Adhoc Approval Specification	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Previewing Adhoc Approvals	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Submitting for Approvals	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Submit For Approvals With Comments Using An Adhoc Approval Specification	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Submitting For Approvals With Comments JSON	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Retrieving Reassign Page URL	Updated topic to include API signature.
	Retrieving Approve or Reject Page URL	Updated topic to include API signature.
	Retrieving Approval Page URL	Updated topic to include API signature.
	Retrieving Approval History	Updated topic to include API signature.
	Retrieving Add Comment Page URL	Updated topic to include API signature.
	Rejecting Approval Requests	Updated topic to include API signature and SOAP Response/ Request XML for API integration.

Document	Topic	Description
	Reassigning Approval Requests	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Previewing Approvals	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Escalating Approval Requests	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Enabling Auto Re-approvals on Proposal and Proposal Line Items	Updated topic to include API signature.
	Creating Ad-hoc Approvals	Updated topic to include API signature.
	Checking if User is Authorised to Approve or Reject	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Canceling Approvals	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Approving Approval Requests	Updated topic to include API signature and SOAP Response/ Request XML for API integration.
	Adding Comments to Approval Requests	Updated topic to include API signature and SOAP Response/ Request XML for API integration.

Spring 2020

Document	Topic	Description
Spring 2020	Submitting a Request with Attachments	Modified topic. Added information for grouping multiple documents across multiple agreement records.
	Creating Search Filters (Approvals)	Modified topic. Added information for character limit increase for Approval Search Filters.
	• Setting Up An Adhoc Approval Process in Runtime • Configuring My Approvals and Approvals Pages • Approvals and My Approval Pages • Submitting a Request with Attachments • Approving a Request	Modified topic. Added information for support added after scalability improvement.
	Upgrading Approvals	New topic. Added information about how to upgrade Intelligent Workflow Approvals.
	Submitting For Approvals With Comments JSON	Added new API - Submit For Approval With Comments API for MAX using JSON parameter to pass comments.
	Submitting For Approvals With Comments	Modified topic. Updated code sample.

Winter 2019

Document	Topic	Description
Winter 2019 Rev A	Setting up Backup or Delegate Approvers	Modified topic. Added information about email notifications for delegate approver.

Document	Topic	Description
Winter 2019	Creating An Approval Process	Modified topic. Added information for creating an Adhoc approval process.
	Setting Up An Adhoc Approval Process in Runtime	New topic. Added info for setting up an Adhoc approval process in runtime.
	Creating Approval Steps	Modified topic. Added information for selecting Carbon Copy assignees. Added reminded jobs information.
	Setting Up Backup and Delegate Approvers	Modified topic. Added info to create multiple entries for Backup/Delegate records.
	Approvals and My Approval Pages	Modified topic. Added information for removing an Adhoc approver from the My Approvals page.
	Previewing Adhoc Approvals	Added API info.
	Submit For Approvals Using An Adhoc Approval Specification	Added API info.
	Submit For Approvals With Comments Using An Adhoc Approval Specification	Added API info.
	Delete An Adhoc Approval Step	Added API info.
	Create a Runtime Adhoc Approval Process	Added API info.
	AdhocApprovalProcessDTO	Added DTO info for Adhoc approval process API.
	Update a Runtime Adhoc Approval Process	Added API info.

Document	Topic	Description
	Retrieve a Runtime Adhoc Approval Process	Added API info.
	Delete a Runtime Adhoc Approval Process	Added API info.
	Add Comment to a Runtime Adhoc Approval Process	Added API info.
	Add Attachments to a Runtime Adhoc Approval Process	Added API info.
	Delete Attachments to a Runtime Adhoc Approval Process	Added API info.
	Use Case and Sample Code for Adhoc Approval Runtime Process APIs	Added sample code for Adhoc approval process runtime APIs.

Summer 2019

Document	Topic	Description
Summer 2019	Custom Query Callback for Reassigning and Adding Approver	Modified topic. Added information for reassigning approver.
	Setting Up a Backup or Delegate for Out-Of-Office	New topic.
	Creating Approval Steps	Modified topic.
	Creating An Approval Process	Modified topic.
	Setting Up Backup and Delegate Approvers	Modified topic.
	Approval Email Notification Templates	Modified topic.

Document	Topic	Description
	Approval and My Approval Pages	Modified topic.

Spring 2019

Document	Topic	Description
Spring 2019	N/A	No new topics were added.

Winter 2018

Document	Topic	Description
Winter 2018	Configuring CPQ Approval Workflow	Modified topic. Added information for Standard Approval Step and Approval Rule Entry Step.
	Adding Comments to Approval Requests	Added API info.
	Approving Approval Requests	Added API info.
	Canceling Approvals	Added API info.
	Checking If an Approval is Required	Added API info.
	Checking if User is Authorized to Approve/Reject	Added API info.
	Creating Ad-hoc Approvals	Added API info.
	Escalating Approval Requests	Added API info.
	Previewing Approvals	Added API info.
	Reassigning Approval Requests	Added API info.
	Rejecting Approval Requests	Added API info.

Document	Topic	Description
	Retrieving Add Comment Page URL	Added API info.
	Retrieving Approval History	Added API info.
	Retrieving Approval Page URL	Added API info.
	Retrieving Approve or Reject Page URL	Added API info.
	Retrieving Reassign Page URL	Added API info.
	Submitting an Approval Request	Added API info.
	Taking Ownership of an Approval Request	Added API info.

Approvals for Administrators

This section describes how Conga Approvals works and how to set up approvals for your customers.

Topic	Description
What's Covered	This section provides administrators with information on setting up approvals within Conga Approvals. It also covers the most common use cases for administration and assumes a level of familiarity with basic Salesforce.
Primary Audience	Admin users responsible for setting up approvals and users for Conga Approvals.
IT Environment	For information pertaining to the requirements and recommendations, see System Requirements and Supported Platforms Matrix.
Updates	For a comprehensive list of updates to this guide for each release, see the What's New in Approvals Documentation topic.
Other Resources	• See Approvals for Users for information on Approvals workflow and use cases. • See Approvals for SOAP API Developers for SOAP APIs provided to work with Approvals objects.

This section describes the following tasks:

- Approvals Functional End User Workflow
- Configuring CPQ Approval Workflow
- Configuring Approvals for Agreement and related Objects
- Configuring Approvals for Custom Objects
- Optional Configuration

Before using Approvals, 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 Conga Approvals
- Configuring Post-Installation Settings

Getting Started with Conga Approvals

Installing Conga Approvals

Multiple packages must be installed to implement the complete Approvals solution. Packages for Approvals 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.

Caution

Conga recommends downloading and installing Conga packages in a Salesforce sandbox org 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.

Order	Package	Install Center tab to access the package	Required?
1	Conga Base Library	CPQ	Y
2	Conga Contract Lifecycle Management	Contract Management	Y
3	Conga Approvals	Advance Approval	Y
4	Conga CPQ Approvals (Required if you are using CPQ)	Advance Approval	Y
5	Conga Quote Management (Required if you are using CPQ Approvals)	CPQ	Y
6	Conga Configuration & Pricing (Required if you are using CPQ Approvals)	CPQ	Y
7	Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	Integrations	Y
8	Conga CLM Configuration Integration (Required if you are using CLM)	Integrations	Y
9	Conga Quote Approvals (Required if you are using Proposal Management)	Advance Approval	Y

Order	Package	Install Center tab to access the package	Required?
10	Conga Custom Approvals Management (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	Advance Approval	Y
11	Conga Grid (Required if you are using Approval Center)	Note: This package is not available on Conga Install Center. Visit Salesforce AppExchange to download it.	Y
12	Conga Approval Center (Required if you are using Approval Center)	Note: This package is not available on Conga Install Center. Visit Salesforce AppExchange to download it.	Y

⚠ You must have Conga-provided login credentials to the Conga Community Portal to be able to download packages.

To install the Approvals managed packages

1. Go to the **Resources** > **Install Center** tab on the Conga Community Portal.
2. In **My Packages** navigation link, click **Advanced Approvals**. From the **VERSION** drop-down, select the version that you want to install.
3. Click **Install Version**.
4. Select the environment in which you want to install the packages.
   - To install the packages in your production environment, click **Install in Production.**
     - OR -
     To install the packages in your sandbox, click **Install in 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 Approvals package.
- Apply other access levels to assign users permissions to Read, Create, Edit, Delete, View All, and Modify All for specific objects in the Approvals package.

> If permission is not enabled for a profile, you can apply Conga-provided permission sets such as the **Conga Approvals Admin** Permission Set and **Conga Approvals 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 Access Control and Permissions.

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 Push Upgrade to auto-upgrade Conga managed packages. Conga Push Upgrade 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 Push Upgrade for auto-upgrade

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

1. Launch the Conga Push Upgrade 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 Push Upgrade.
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 Push Upgrade, see Conga Push Upgrade documentation.

Approval Objects

This section lists out the objects inside the **Conga Approvals Management** package.

Objects	Purpose of the object
Adhoc Approval Process	Defines adhoc approval process information for a specific business object.
Adhoc Approval Group	Defines groups of approvers for an adhoc approval process.
Adhoc Approver	Defines approvers for an adhoc approval group.

Objects	Purpose of the object
Backup/Delegate Approver	Allows approvers and administrators to specify backup approvers. There is an option to transfer all pending request from and to the approver and the backup approver upon activation or cancellation of backups. The approvals engine automatically routes requests to backup approvers when this is in effect.
Approval Request	Designates an approval request created from an approval process.
Term Exception Approval Condition	Required conditions for a given Term Exception Approval.
Term Exception Approval	Required approvals for a given Term Exception.
Global Discount Policy	Designates global discount policies for six levels of approvals.
Search Filter (Approvals)	Represents a search filter.
Approval Process	Designates an approval process defined in the system.
Approval Request History	Designates history for an approval request created from an approval process.
Approval Process Instance	Represents an instance of a single, end-to-end approval process.
Approval Rule Assignee	Represents a single assignee in an approval rule entry.
Approval Rule	Represents a single rule in an approval ruleset.
Approval Matrix	Specifies user approval levels and authorized discount percentages.
Approval Rule Dimension	Represents an approval rule dimension.
Approval Rule Entry	Represents a single entry in an approval rule.

Objects	Purpose of the object
Reminder	Specifies design time information for process step and subprocess reminders.
Reminder Instance	Specifies runtime information for process step and subprocess reminders.
Formula Field (Approvals)	Represents a formula field used in approval rules.

Conga Custom Approvals is a sophisticated workflow application for standard or custom objects that allows you to create and manage complex approval processes beyond the capability built into Salesforce. You can route approval requests to various approvers in series or parallel along with the ability to process notifications and actions at each step of the process.

This section lists out the object inside the Conga **Custom Approvals Management** package.

Objects	Purpose of the object
License	Represents a license to enable custom approval feature.

Upgrading Approvals

This section covers all the required upgrade tasks for upgrading Approvals to the latest version from the previous two releases.

If you do not have Approvals installed, you can contact Conga Support to request an installation link, then perform the standard installation as described in Installing Conga Approvals.

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.

Preparing for Upgrade

Before you upgrade to Approvals Winter '20, you must ensure the following:

- You go through Approvals Feature By Release to know about the new features, enhancements, and deprecated features in Approvals since your existing release. After you upgrade Approvals to Winter '20, 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 for Approvals managed package and dependent packages.
- You must 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.

Upgrading to Approvals on Salesforce Winter '20

This section describes step-by-step instructions to upgrade from Summer 2020 to Winter '20

Upgrading Approvals Summer 2020 to Approvals Winter '20

1. Go to **Setup** > **Installed Packages** and ensure that your Salesforce org has the following Summer 2020 packages already installed.

Order	Product	*Version Name / Version Number*
1	Apttus Approvals Management	11.1.0161 | 11.161
2	Apttus CPQ Approvals Management (Required if you are using CPQ)	11.0.0018 | 11.18
5	Apttus Base Library (Pre-requisite for Apttus Contract Manangement)	1.0.93 | 1.93
6	Apttus Contract Management	11.0.0547 | 11.547
7	Apttus Proposal Management (Required if you are using CPQ Approvals)	10.1.0209 | 10.209
8	Apttus Configuration & Pricing (Required if you are using CPQ Approvals)	12.1.1787 | 12.1787

Order	Product	*Version Name / Version Number*
9	Apttus Quote/Proposal Configuration Integration (Required if you are using CPQ and Proposal management)	12.1.0332 | 12.332
10	Apttus Contract-Configuration Integration (Required if you are using CLM)	10.1.0103 | 10.103
11	Apttus Quote/Proposal Approvals Management (Required if you are using Proposal Management)	6.5.0004 | 6.4
12	Apttus Custom Approvals Management (Required if you are using Apttus Objects or other Salesforce Standard/Custom Objects)	1.0

2. Ensure that you have the following packages and dependent packages to upgrade to Winter '20. These packages are required to utilize the new features and enhancements of Winter '20.

Order	Product	*Version Name / Version Number*
1	Conga Approvals	11.2.0190 | 11.190
2	Conga Grid (Required to use Approval Center)	2.101 | 2.101
3	Conga Approvals Center (Required to use Approval Center)	11.3.0 | 11.3
4	Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
5	Conga Base Library (Pre-requisite for Conga Contract Management)	1.2.123 | 1.123

Order	Product	*Version Name / Version Number*
6	Conga Contract Lifecycle Management	11.2.0582 | 11.582
7	Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
8	Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839 | 12.1839
9	Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344 | 12.344
10	Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146 | 12.146
11	Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4
12	Conga Custom Approvals Management (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

3. Perform the upgrade. The upgrade procedure is the same as the installation procedure. Install the required managed package in the same order as mentioned in the table below. For detailed information on installing managed packages, see Installing Conga Approvals.
4. Update the app name and app logo to reflect the latest Conga branding. For more information, refer to Updating App Name and App Logo.
5. After the upgrade is complete, perform the post-upgrade tasks.

Upgrading Approvals Spring 2020 to Approvals Winter '20

1. Go to **Setup** > **Installed Packages** and ensure that your Salesforce org has the following Spring 2020 packages already installed.

Order	Product	*Version Name / Version Number*
1	Apttus Approvals Management	11.0.0115 | 11.115
2	Apttus CPQ Approvals Management (Required if you are using CPQ)	11.0.0016 | 11.16
3	Apttus Base Library (Pre-requisite for Apttus Contract Management)	1.0.38 | 1.38
4	Apttus Contract Management	11.0.0518 | 11.518
5	Apttus Proposal Management (Required if you are using CPQ Approvals)	8.5.0177 | 8.177
6	Apttus Configuration & Pricing (Required if you are using CPQ Approvals)	11.0.1552 | 11.1552
7	Apttus Quote/Proposal Configuration Integration (Required if you are using CPQ and Proposal management)	10.2.0263 | 10.263
8	Apttus Contract-Configuration Integration (Required if you are using CLM)	10.1.0103 | 10.103
9	Apttus Quote/Proposal Approvals Management (Required if you are using Proposal Management)	6.5.0004 | 6.4
10	Apttus Custom Approvals Management (Required if you are using Apttus Objects or other Salesforce Standard/Custom Objects)	1.0

2. Ensure that you have the following packages and dependent packages to upgrade to Winter '20. These packages are required to utilize the new features and enhancements of Winter '20.

Order	Product	*Version Name | Version Number*
1	Conga Approvals	11.2.0190 | 11.190
2	Conga Grid (Required to use Approval Center)	2.101 | 2.101
3	Conga Approvals Center (Required to use Approval Center)	11.3.0 | 11.3
4	Conga CPQ Approvals (Required if you are using CPQ)	11.2.0019 | 11.19
5	Conga Base Library (Pre-requisite for Apttus Contract Management)	1.2.123 | 1.123
6	Conga Contract Lifecycle Management	11.2.0582 | 11.582
7	Conga Quote Management (Required if you are using CPQ Approvals)	10.2.0227 | 10.227
8	Conga Configuration & Pricing (Required if you are using CPQ Approvals)	12.2.1839 | 12.1839
9	Conga Quote Configuration Integration (Required if you are using CPQ and Proposal management)	12.2.0344 | 12.344
10	Conga CLM Configuration Integration (Required if you are using CLM)	12.2.0146 | 12.146
11	Conga Quote Approvals (Required if you are using Proposal Management)	6.5.0004 | 6.4

Order	Product	*Version Name / Version Number*
12	Conga Custom Approvals Management (Required if you are using Conga Objects or other Salesforce Standard/Custom Objects)	1.0

3. Perform the upgrade. The upgrade procedure is the same as the installation procedure. Install the required managed package in the same order as mentioned in the table below. For detailed information on installing managed packages, see Installing Conga Approvals.
4. Update the app name and app logo to reflect the latest Conga branding. For more information, refer to Updating App Name and App Logo.
5. After the upgrade is complete, perform the post-upgrade tasks.

Performing the Post Upgrade Tasks

After upgrading the Approvals package and dependent packages, you must ensure the following:

- All custom settings are correct as described in Approval System Properties and Approvals Custom Config Settings.
- All upgrade scenarios must work the same way as the previous release.
  - Callbacks: Call back class must not change.
  - Metadata changes: Validate all pick-list type fields.
  - Layout changes: Validate layout changes.
  - RemoteAPIcalls: All API calls must be functional.
- All approval process data must work after upgrade including submit/approve/reject flows.

⚠ The upgrade procedure above is tested against basic setup and not custom code or trigger.

Updating App Name and App Logo

After you upgrade to the Winter '20 version of Conga Approvals, you need to upgrade the app name and app logo to reflect the latest Conga branding. It is required only if you are upgrading from a previous version of Apttus Intelligent Workflow Approvals to Conga

Approvals Winter '20 or later version. If you are installing Conga Approvals for the first time, you do not need to update the app name and app logo.

Updating the App Name and Logo

For Salesforce Classic:

1. Navigate to **Setup → Build → Apps.**
2. Click **Edit** next to Apttus Approval Management.
3. Update the App Label to *Conga Approvals.*
4. Click Insert an Image.
5. Select Conga Documents from File Location drop-down menu.
6. Select Conga Logo. Click **Save.**
7. Repeat steps 2-6 for Apttus Approvals Setup.

For Salesforce Lightning:

1. Navigate to **Setup → Platform Tools → Apps.**
2. Click **Edit** next to Apttus Intelligent Workflow Approvals.
3. Update the App Label to *Conga Approvals.*
4. Click Insert an Image.
5. Select Conga Documents from File Location drop-down menu.
6. Select Conga Logo. Click **Save.**
7. Repeat steps 2-6 for Apttus Approvals Setup.

Optional Configuration

The following configurations are optional and common for CPQ Approvals and Intelligent Approvals for Custom Objects.

- Changing the Default Approvals Process User Interface
- Changing the Assignee Search page results list
- Opportunity record fields and Advanced Currency Management
- Support for Conga Approvals from Communities
- Reporting on Approval Request Channels
- Changing Approval Rule Step Behavior
- Setting up Custom Assignee Value Expression Objects
- Creating Approval Matrices
- Adding Related fields to a Criteria
- Child Object Filters for Step Entry Criteria
- Configuring the Cart Approvals Page and Apex Class

- Setting up Custom Queues

Changing the Default Approvals Process User Interface

Conga Approvals provides a default wizard-like set of pages that guide you through the process of creating approval processes and term exception approvals. You can change this behavior and use the standard Salesforce page instead. Changing the setting does not affect the object values, but the ability to be guided through the creation and editing process is lost when you change to the standard Salesforce page.

To change the default Intelligent Approvals Process UI

1. Go to **Setup > Objects** and click **Approval Process**.
2. From the **Buttons, Links, and Actions** related list, click **Edit** next to **Approval Processes Tab**. By default, *ApprovalProcessDefnList (Visualforce Page, Package: Conga Approvals)* is selected.
3. From the **Salesforce Classic Override** field, select the **Standard page** and click **Save**.
4. From the **Buttons, Links, and Actions** related list, click **Edit** next to **View**. By default, *ApprovalProcessDefn (Visualforce Page, Package: Conga Approvals)* is selected.
5. From the **Salesforce Classic Override** field, select the **Standard page** and click **Save**.
6. Repeat step 4 and step 5 for **Edit** and **New**.

Now when you access the **Approval Processes** tab and create an item, view, or edit an existing item, the standard Salesforce page appears instead of the step-by-step wizard pages. To revert back to the default Approval Processes UI, follow the steps above and change the settings back to *ApprovalProcessDefnList (Visualforce Page, Package: Conga Approvals)* and *ApprovalProcessDefn (Visualforce Page, Package: Conga Approvals)*.

To change the default Term Exception Approvals UI

1. Go to Setup > Objects and click Approval Process.
2. From the **Buttons, Links, and Actions** related list, click **Edit** next to **Edit**. By default, *TermExApproval (Visualforce Page, Package: Conga Approvals)* is selected.
3. From the **Salesforce Classic Override** field, select the **Standard page** and click **Save**.
4. From the **Buttons, Links, and Actions** related list, click **Edit** next to **New**.
5. From the **Salesforce Classic Override** field, select the **Standard page** and click **Save**.

Now when you access the **Term Exception Approvals** tab and create a New item or Edit an existing item, the standard Salesforce page appears instead of the step-by-step wizard pages. To revert back to the default Term Exception Approvals UI, follow the steps above

and change the settings back to *TermExApproval (Visualforce Page, Package: Conga Approvals).*

Changing the Assignee Search page results list

By default, when searching for a step assignee type of *Queue*, *Role*, *Rule*, *Subprocess* or *User*, the results are displayed twenty per page.

The number can be changed; however, it is important to understand how changing the value can impact performance. If the number is too high, it can take additional time for the search page to load the results. When no value is designated, the default twenty is used.

To set the assignees search page

Build > Develop > Custom Settings

1. Click **Manage** beside **Approvals System Properties**.
2. Click **Edit** beside **System Properties**.
3. In the **UI Assignee Search Page Size** field, enter the maximum number of rows to display on each search page.
4. Click **Save**.

The next time someone is configuring an approval step and selects an Assignee Type of *Queue*, *Role*, *Rule*, *Subprocess* or *User*, the resulting search page will reflect this setting.

Opportunity record fields and Advanced Currency Management

Advanced currency management (ACM) is optional Salesforce functionality that must be enabled directly by Salesforce, to allow you to manage dated exchange rates on opportunity records.

When ACM is enabled in your org and being used on opportunities, you cannot use any Opportunity object fields that are of a Data Type of Currency in your intelligent approval emails or on the approval summary page. This includes the standard Amount field and any custom fields using the currency type.

If you want to have currency type field information in your emails, you cannot have the values within the content of the email. You would need to have links back to the

opportunity record itself in the email and users would have to view the date within Salesforce.

Support for Conga Approvals from Communities

Approvals are now supported for Partner Managed Quotes and Contracts using Salesforce Communities. Your channel partners can not only create quotes and contracts, but can also access all the approval functions that are available to your internal employees. For a community page setup, enable the following functions as community user:

- Preview Overall from Cart
- Return to Cart from Cart Preview Page
- Preview Overall from Quote Header
- Line Item Preview from Cart
- Submit for Approval from Cart Preview
- Return to Quote from Cart Submit for Approval confirmation page
- Submit for Approval from Quote Preview Page
- Return to Quote from Submit for Approval from Quote Header Preview page
- Approvals History page from Quote Header for Submitted Quote
- My Approvals Page (When no approvals needed)
- My Approvals Page (When approvals needed)
- Approve from My Approvals Page
- Reject from My Approvals Page
- Take Queue Ownership from My Approvals Page
- Cancel Submitted Approval as community user

Do the following to provide access for a community user:

1. Assign all the Approval packages to the Community User.
2. Create a permission set for the community users.
3. Assign Apex Pages/Classes access to the permission set for community users.
4. Manage User Assignments for the Community User.
5. Provide fields and object level permissions.
6. For Objects, set the Field Level accessibility by profile - Hidden/Visible/Editable
7. Navigate to Object > Page layouts > Page Layout assignment. Associate profiles to the layout.
8. Go to Permission set > Object Setting, provide Create, Read, Edit, and Delete permissions for the community user.
9. Navigate to Permission Set > Object Settings > Proposals > Provide permission to the Configure Products (Approvals) link or button .

> If any error is appears pertaining the Read permissions of account locations, navigate to Object Settings > Account Locations > and provide all 4 Create, Read, Edit, and Delete permissions.

10. Go to Setup > All Communities > Administration > Settings and make the tabs visible to your community user.

Errors you may face during approvals community setup :

- After clicking the Configure Products (Approvals) button, Error: SObject row was retrieved via SOQL without querying the requested field: Apttus_Proposal_Proposalc.Apttus_QPConfigLocationId_c.
  Resolution : > You might have missed point 9.
- After clicking the Configure Products (Approvals) button, Error : Assigned to : Assigned To ID : Owner
  Resolution : > Go To Custom Settings -> Go To Config System Properties > Assign Admin User
- When you click Submit For Approvals, error : Backup Admin Not Assigned error appears, navigate to Setup-> Custom Settings -> Approvals Custom Config > Click ProductConfiguration__c > Backup Admin User and provide a Backup Admin user name.

To configure buttons and fields for approvals in communities

Ensure that you have set the requisite permissions for community users for apex classes, objects, and VF pages.

1. Navigate to the **Approval** object and create a custom field for approval actions with datatype as formula. The formula field for the custom field remains the same as that of the Approval Action field except for the hyperlink.
2. Add the site prefix specific to the community setting, to the hyperlink. For example, HYPERLINK("/developer/Apttus_Approval__ApprovalRequestEscalate?id="&Id,"Escalate","_top"), here developer is the site prefix for the community org.
3. Ensure that you add the custom fields to the appropriate page layout of the pages the community user has access to.

After you add the button to the community page layout, the button is visible to the community user having access to the page.

For each community page which has a specific hyperlink, configure separate custom buttons and fields for the community and associate the site prefix in the hyperlink of the particular custom button or field.

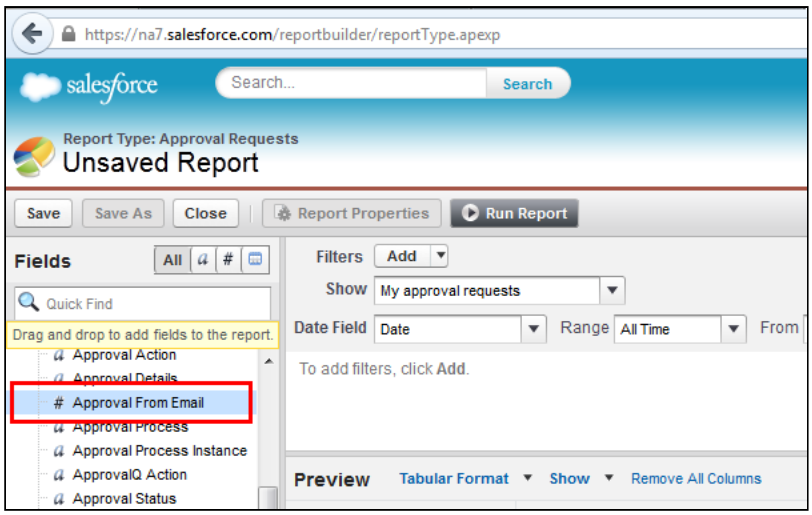
Reporting on Approval Request Channels

As approval requests can be completed within Salesforce or outside of the system via email, this feature provides a way to identify which of the two is used for each approval request.

By adding the new Approval From Email field to a report, based on the Approval Request custom object, you are able to see whether someone actioned an approval request via email. This means they replied to an approval request email by entering the word *APPROVE*, *APPROVED*, *YES*, *REJECT*, *REJECTED*, or *NO* in the first line of their email response. Any other method of deciding on an approval request is considered as approving it via Salesforce.

To create a custom report

1. Click ☐ to display all tabs and then select **Reports**.
2. Click **New Report** and from the **Select Report Type** list, expand the **Other Reports** folder.
3. Select **Approval Requests** and click **Create**.
4. From the **Fields** list, drag and drop **Approval From Email** and any other fields you want to include in the report.



5. Click **Save** to display the Save Report dialog, enter the required details, and click **Save**.

The report is saved in Salesforce.

After a number of approval requests have been decided on, you can run the report and see which requests were approved via email.

You must also ensure that the Enable Email Approval Response custom setting for Approvals System Properties is enabled. This is required for approvers to be able to approve approval requests directly in the email, outside of Salesforce.

Changing Approval Rule Step Behavior

By default, if an approval rule step cannot find an assignee to send an approval to, it will get routed to the backup approver. The *APTS_EnableRuleCriteriaForStep* admin property setting can override the default behavior, and skip the step, instead of sending it to the backup approver.

If this property is set to *True*, when a rule has no assignees the step is skipped. This has the same effect as a step filter that evaluates to false.

To set the admin property setting

1. Select the **Admin** tab - or - select  and select **Admin**.
2. Click **Go** to display all of the admin settings.
3. If **APTS_EnableRuleCriteriaForStep** is listed, click **Edit**, enter **True** or **False** in the **Value** field, and click **Save**. - or - if the setting is not displayed, click **New Admin**.
4. Enter the name **APTS_EnableRuleCriteriaForStep**, enter **True** or **False** in the **Value** field, and click **Save**.

Subsequent approval processes that are routed to a step where an assignee cannot be found will be impacted by this setting.

Setting up Custom Assignee Value Expression Objects

To have values available in the Object picklist, when creating a Custom Value Expression for determining assignees, those custom objects must be added to Approval System Properties. For more details on where these objects are used in Intelligent Approvals, see Creating Expressions.

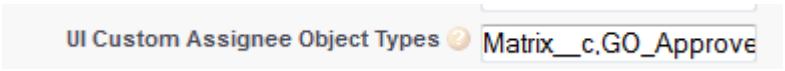
> **Note**
>
> The custom Salesforce objects are configured as normal.

To set up Custom Value Expression Objects

The custom objects you want to add must have been created already.

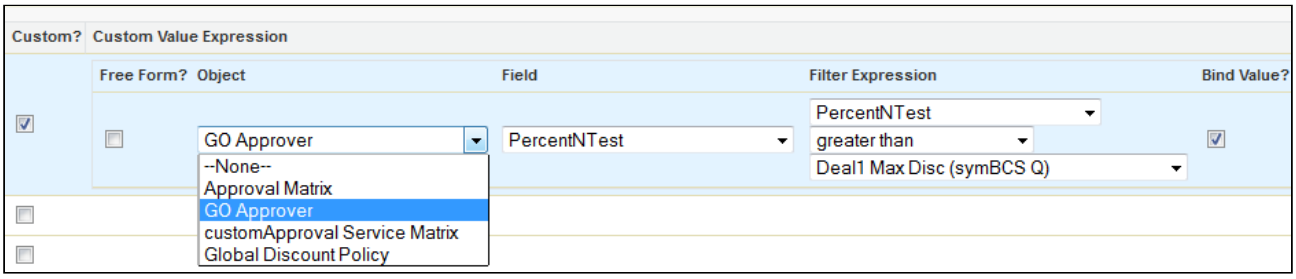
Build > Develop > Custom Settings

1. Click **Manage** for **Approvals System Properties**.
2. Click **Edit** for **System Properties**.
3. In the **UI Custom Assignee Objects Types** field, enter API names for the custom objects you want to use, separated by commas. The format should be `CustomObjectname1__c, CustomObjectname2___c, CustomObjectname3__c`.



4. Click **Save** when complete.

The custom objects are now available for selection in the *Object* picklist in the *Custom Value Expression* section of Entry Step Criteria, when you are selecting approval request assignees. If you select Free Form, you can enter your custom formula in the Value field.

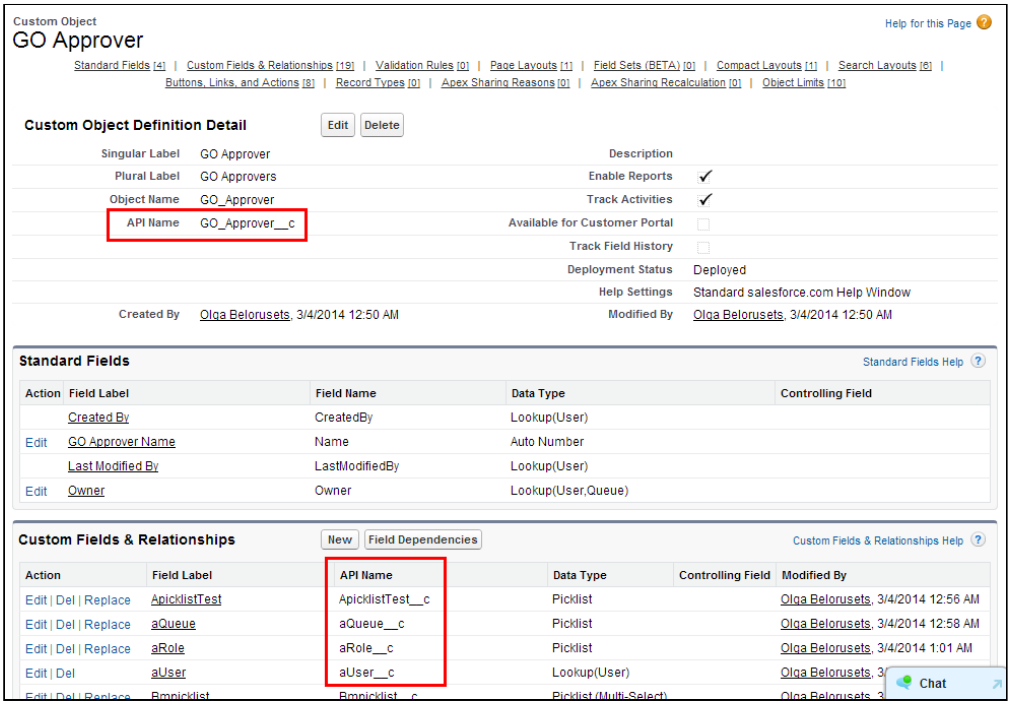


Custom Assignee Example

You can create custom objects to be used with custom assignee types in approval rules. This example will take a custom object called *GO Approver* and show how it is used to provide assignees when creating an approval rule.

Step 1: Create your custom object.

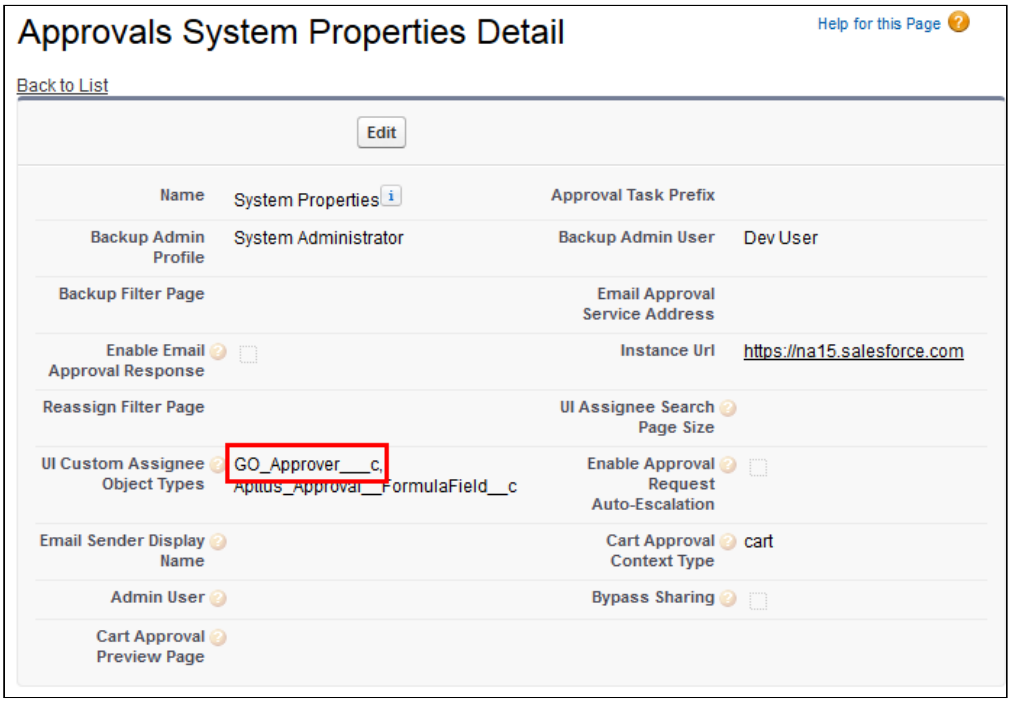
From **Setup**, go to **Create > Objects** and click **New Custom Object**. Create your custom object, adding the necessary fields to enable you to create a formula for selecting assignees. The API name for the custom object and its fields are what are used to create the custom formula in the approval rule.



Step 2: Make the custom object available to Intelligent Approvals via a custom setting.

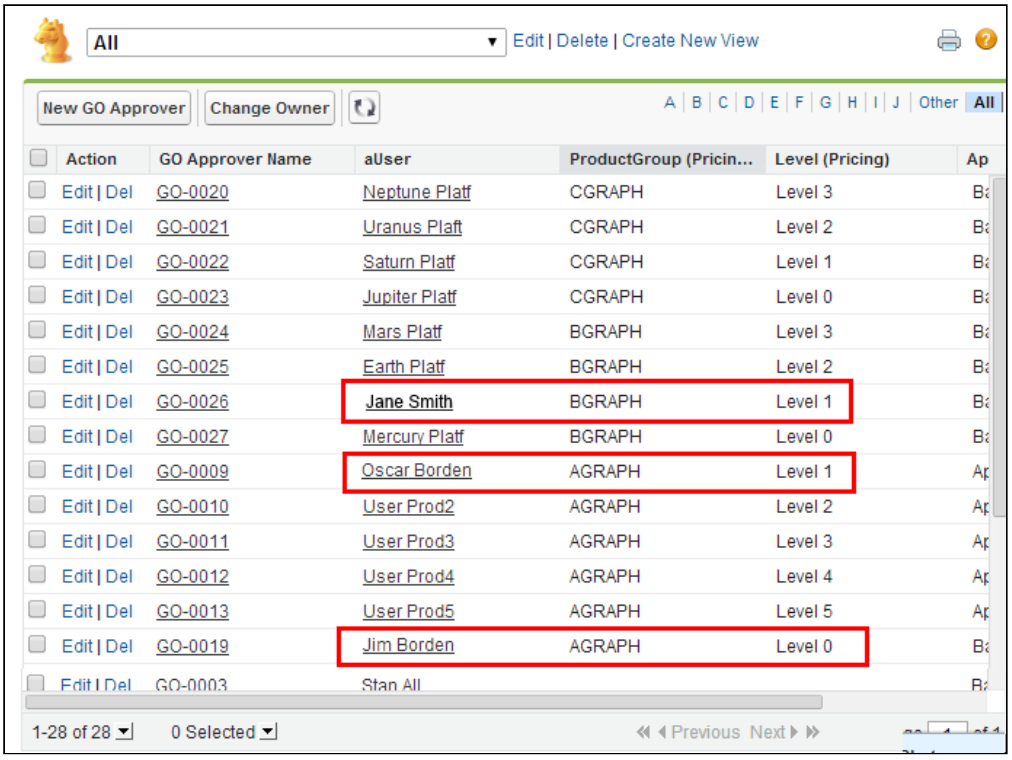
From **Setup**, go to **Develop > Custom Settings** and click **Manage** for **Approval Systems Properties**. Click **Edit** and for **UI Custom Assignee Object Types** enter the name of the custom objects you want to use with custom assignees.

The Approval Systems Property must include the API name of the custom object:



Step 3: Create users for the custom object.

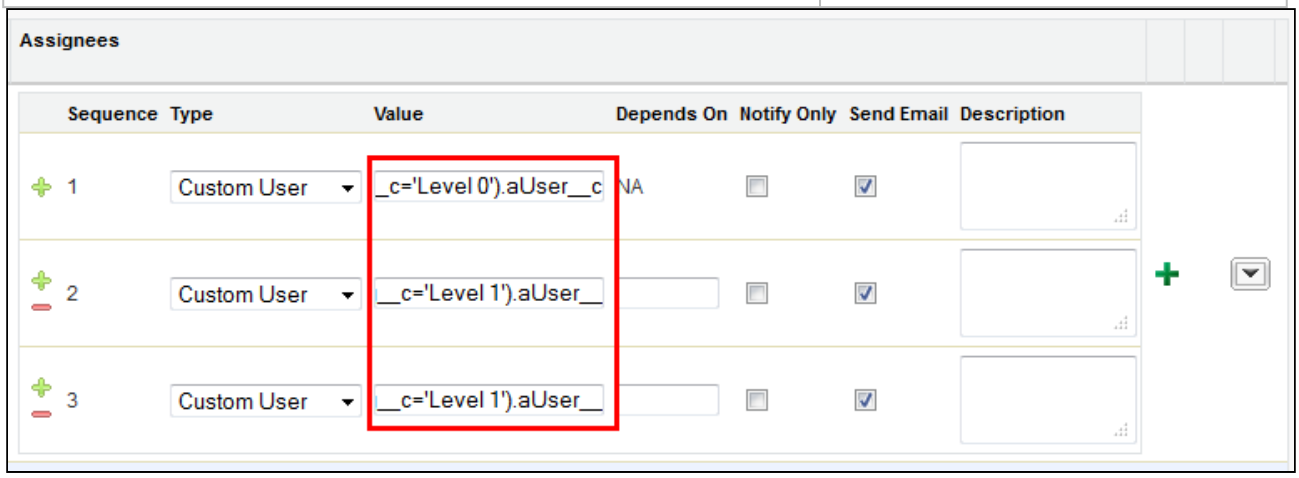
Go to the *GO Approver* tab to display the page and create all the users you want to approve requests. The values you enter in the fields for each user is what is used when you create formulas for selecting assignees.



Step 4: Create your approval rules, using formula expressions to select assignees.

Go to an Approval Rule of Type *Condition* and create assignees by selecting *Custom User* and then enter a formula in the *Value* field. For instance the following formulas would route an approval to the following assignees, based on the users you created for the custom object.

`GO_Approver__c(ProductGroup_Pricing__c='AGRAPH' AND Level_Pricing__c='Level 0').aUser__c`	Jim Borden
`GO_Approver__c(ProductGroup_Pricing__c='AGRAPH' AND Level_Pricing__c='Level 1').aUser__c`	Oscar Borden
`GO_Approver__c(ProductGroup_Pricing__c='BGRAPH' AND Level_Pricing__c='Level 1').aUser__c`	Jane Smith



Creating Approval Matrices

Approval matrices enable you to dynamically assign an approval request to someone selected from a list of a maximum of six potential approvers. An approval matrix can be selected as an assignee type when creating an approval step.

The potential approvers are assigned an approval level, which is associated with a percentage discount. Those discount values are based on the *Authorized Discount Percents* set up in Global Discount Policies. You must have these policies set up for an approval matrix to route approvals to the correct level.

The Global Discount Policy values set the default discount percentage levels that will typically require approval and also designate who the approver should be for each level. Matrices enable you to have approvals start at any of the policy levels. For instance, if the

policy's *Authorized Discount Percent* for *Approval Level 1* is 15% and no approval is required until the discount on certain items reaches 35%, an approval matrix can be created that does not start until the approval level 2.

To create Global Discount Policies

1. Select the **Global Discount Policies** tab or click **All tabs** and then select **Global Discount Policies**.
2. Click **New** and enter a whole number for the **Authorized Discount Percent**. This represents the percentage discount at which an approval will be routed to the associated approval level.
3. Enter a description and then select the **Approval Level** for the discount policy entry. This value will be used in the approval matrix to associate a user with the approval level.
4. Click **Save** to return to the Global Discount Policy list - or - click **Save & New** and create another policy entry.

Once you have created the required policies, you can create an Approval Matrix that will make use of the Global Discount Policy levels.

To create an approval matrix

There must be Global Discount Policy levels for each level you want to use in the matrix.

1. Click the **Approval Matrix** tab or click **All tabs** and then select **Approval Matrix**.
2. Click **New** to display the Approval Matrix Edit page.
3. Enter the required values for the following:

Option	Description
Matrix Owner	This is the Salesforce.com user account associated with the matrix. This means the matrix can also be accessed from the *Custom Links* section on that user's account page. The owner and the user do not need to be the same user account.
User	This is used to find the correct approval matrix, for when an object enters the intelligent approval process. For instance, if an opportunity enters the approval process and an approval step assignee type is an approval matrix

Option	Description
Is Active	Select this option to have the matrix available for inclusion in an approval process. **Note** **Is Valid** is an internal Salesforce.com setting that can be ignored.
Approval Level	This is the level the approval matrix will start at, which corresponds to the same level for global discount policies.
Next Level Approver	This approver is typically the same as *Approver 2*.

4. Select the Approver, Authorized Discount, and Approver Level values for each approver you want to include in the matrix. The Authorized Discount value should match those of the Global Discount Policies.
5. Click **Save** to return to the Approval Matrix list - or - click **Save & New** and create another matrix.

If it is active, the approval matrix can now be selected when configuring approval process steps.

After you create an Approval Matrix record, you can see the following links when you edit the Approval Matrix record:

- Generate Approval Matrix: When you create a new approval matrix for a single user, you can simply click the Generate Approval Matrix link. This creates the matrix for a new user and adjusts any other matrices that have dependencies on the created batch job.
- Generate Approval Matrix for All: Click the Generate Approval Matrix for All if you have changed values in the Global Approver Settings (the threshold percentages) which requires that all matrices be regenerated and use a batch job mechanism as well.
- Set Next Level Approver: Enables you to change the next level approver specified in the matrix. For example, you can change Approver 2 to Approver 5.

Expected Behavior

The suggested way to use these matrices is as follows:

- For a single user, create a new approval matrix record and enter the minimum information such as User, isActive, isValid, (both true) and Approval Level. Click Generate Approvals Matrix to fill in the rest of the values of the matrix automatically.
- For multiple users, create a new approval matrix record for each user and enter the minimum information such as User, isActive, isValid, (both true) and Approval Level. Click Generate Approvals Matrix to fill in the rest of the values of the matrix automatically. Click Generate Matrix for All to do the same (this runs more jobs and updates to matrices for all users).
- For a bunch of users or whenever global discounts have been adjusted, pick any matrix record and click Generate Approval Matrices for All. Ensure that you specify the first and last matrix records so every single record gets updated.

Adding Related fields to a Criteria

Related fields can be used to add custom formula fields to Intelligent Approval criteria.

Specifically you can use related fields with:

- Approval Process Entry Criteria
- Approval Step Entry Criteria
- Re-Approval Criteria
- Child and Search Filters
- Approval Rules Criteria - Both Dimension and Condition
- Approval Rule Entry Criteria - Approval conditions and Auto Reapproval conditions

When you configure a formula field to be used as a related field, it is automatically available with each of the criterias listed above.

To add business objects to Formula Field picklist

1. From Setup, select **Create > Objects > Formula Field (Approvals)**.
2. From **Custom Fields & Relationships**, click **Business Object**.
3. From **Picklist Values**, click **New**.
4. Enter the API Name for the objects you want to include, such as `Apttus_Config2__ProductConfiguration__c` and `Apttus_Config2__LineItem__c`.
5. Click **Save**.

The API names are now added to the picklist values.

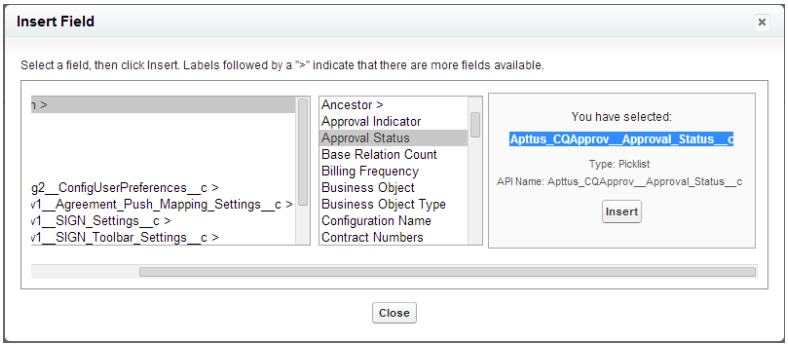
The next time you create or edit a Formula Field (Approvals), based on the values in Step 4, *Product Configuration* and *Line Item* are available from the *Business Object* picklist.

To set up Formula Fields (Approvals)

If you want additional Business Object options other than the default values, they must be added to the *Formula Field* (Approvals) business object picklist.

1. Click All tabs and select **Formula Fields (Approvals)**.
2. Click **New** and enter the following:

Option	Description
Field Name	Name displayed in the Field list when setting entry criteria for a process, step filters, conditions, or re-approval criteria.
Type	Currently this value can only be *Relationship*.
Business Object	This is the business context object that relates to the context object of the approval process or approval step. The field will only be available in a process or step, when this value matches that of the process or step. The available default values are *Standard Agreement*, *Agreement Line Item*, and *Opportunity*.

Option	Description
Formula	Contains fields from the business object, such as `Apttus_Config2__PriceListId__c` for the Price List field on the Configuration object. It can also be used for fields that are associated with the business object via a lookup field. For this you must use `__r` to indicate there is a relationship between the field and the object. For instance, `Apttus_Config2__ConfigurationId__r.Apttus_QPConfig__Proposald__r.Payment_Term__c` enables the Payment Term field to be used when Product Configuration is the business object. You can use Salesforce Insert Field dialog to build your formula and then copy and paste it into the Formula field. You can access the formula by creating a new field for the appropriate business object. In this example the *Approval Status* for the *Product Configuration* object. 
Description	Enter information to state the purpose of the formula field is.

3. Click **Save**.

The new field is added to the Formula Field (Approvals) list and can be included in an approval process.

When you go to add criteria in approvals, the Field list will display the related field, with the format *Related(Field Name): Formula Field Label*. For example *Related(Price List): Price List*.

Child Object Filters for Step Entry Criteria

When you create step entry criteria for your approval process, you can now use custom child object fields with the expressions. Previously you were limited to the fields of the main context object.

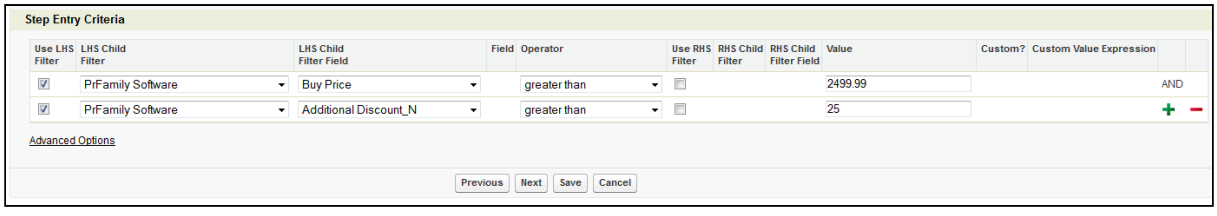
When creating an approval step you can choose to use a child filter. Here you can select from a list of search filters that are related to child objects. This enables you to then select any of that child object's fields. That field value can be compared with another child object field, a static value, or a custom expression.

As with any expression, you must ensure that the operators you choose are applicable to the data type of the fields you are comparing. For instance, when using a percentage field you can use operators such as *less than* and *greater than*, but not *contains* or *starts with*. As well the value you are comparing against would need to be another percentage field or a static value such as the number *75*.

Configuring the Search Filters

You must have created any custom child objects you want to use the filters.

1. Go to **Build > Installed Packages > Conga Approvals** and under **Package Components** select **Business Object** for the **Search Filter** parent object.
2. Under **Picklist Values** click **New** and enter the API name for the child object you want to use with child filters. Repeat the step for each child object.
3. Go to the **Search Filters** tab or click **All tabs** and then **Search Filters**.
4. Click **New**, select the **Business Object** you want to use the filters with and click **Next**. This can include child objects of objects associated with an approval process.
5. Enter a **Filter Name** and **Description**.



6. Enter the required **Filter Criteria**, which will be used in the step entry criteria for the approval process. It is this expression that forms the basis of the child filter expression in the step entry criteria, for determining if an approval step is used.
7. Click **Save**.

The child object will be available when you select *Use LHS Filter* or *USE RHS Filter*, when creating an approval process for its parent object.

Walkthrough

Filters must have been created already via the Search Filters tab.

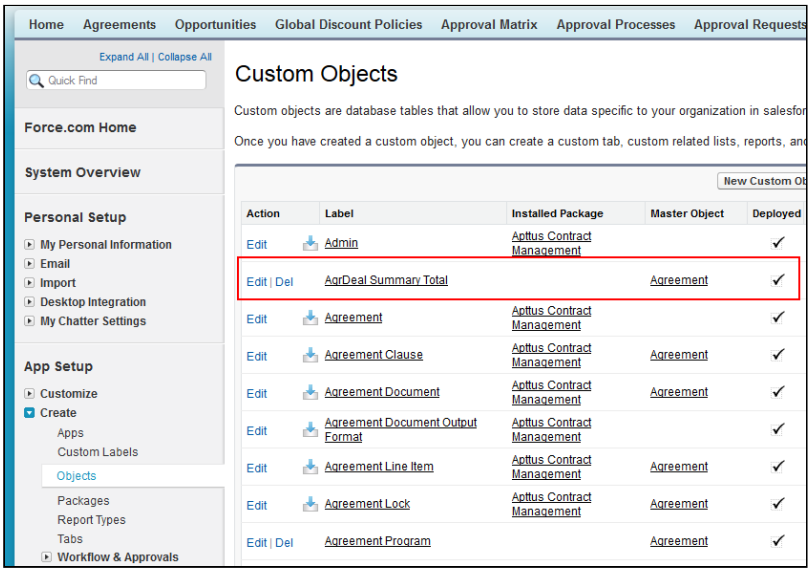
1. For an approval process, click **New Approval Step**.
2. Enter a name for the step and click **Next**.
3. Click **Use LHS Filter** and the **LHS Child Filter** and **LHS Child Filter Field** drop-down lists are displayed. All search filters created for the child object of the approval process are available from the **LHS Child Filter** list.
4. Select the child object from the **LHS Child Filter** and select one of its fields from **LHS Child Filter Field**.
5. Select a valid operator for the field type and then select another filter or fixed value to compare it to.
6. Add further filters if necessary and then click **Next** to finish creating the step.

When an object record submitted for approval matches the step entry criteria, the approval process will be routed to that step's assignee.

Example: Child Object Filters

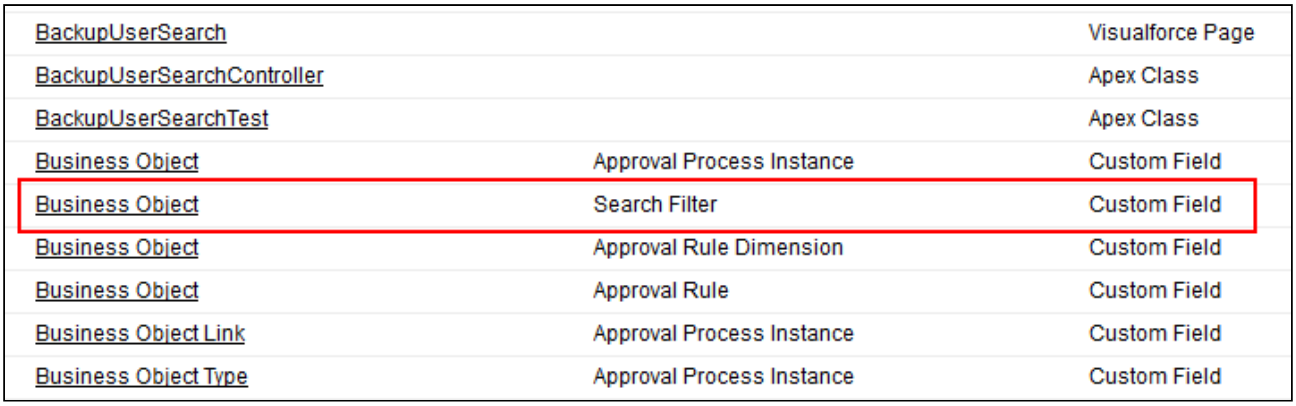
Using Agreements as an example, previously you could only use Agreement fields, such as Total Agreement Value, Category, and Status for entry criteria. With this feature you could create a new custom object and associate it with the Agreement object, enabling it to be used for step entry criteria.

Step 1: Create the custom child object for the Agreement object. In this instance that will be *AgrDeal Summary Total*. This will include fields used for indicating additional discounts for products included in an agreement.

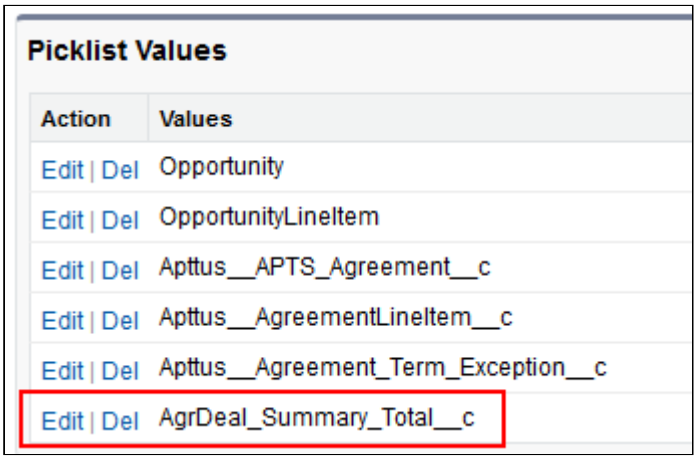


Step 2: After the new custom object has been created for Agreements, the *AgrDeal Summary Total* related list needs to be added to the *Agreements* page layout.

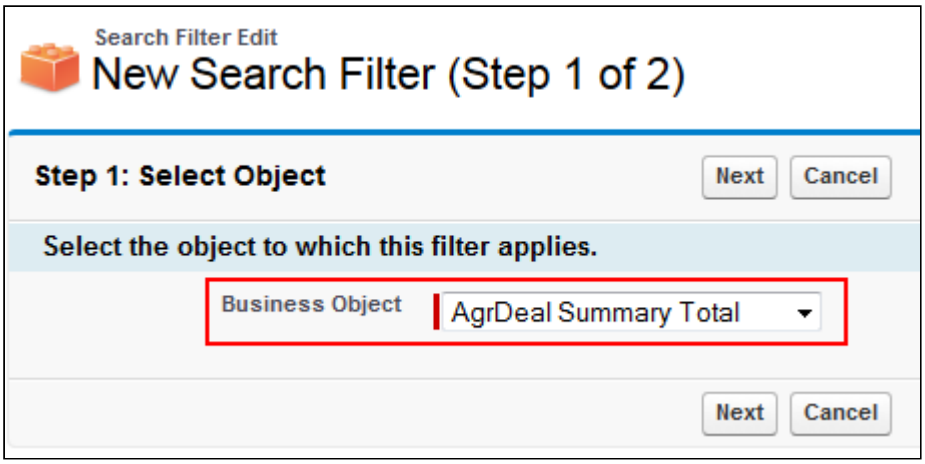
Step 3: Add *AgrDeal Summary Total* to the picklist for the Search Filter Object. This is required to have it available when you set up a Search Filter in the next step. You got to *Build > Installed Packages > Conga Approvals > View Components* and select *Business Object* (Search Filter).



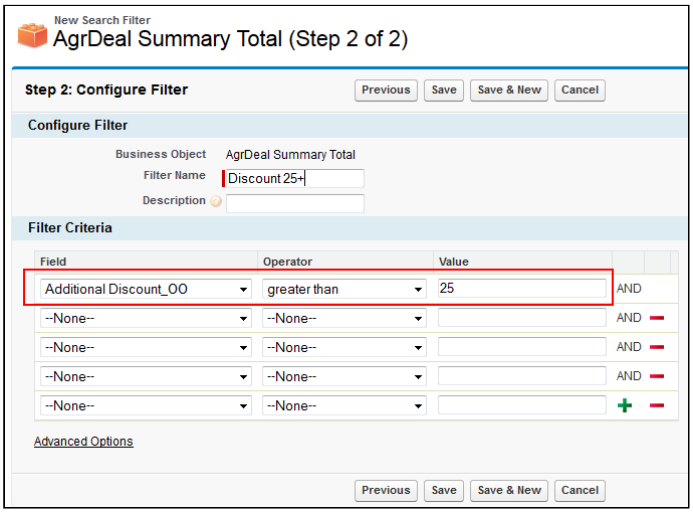
The *AgrDeal Summary Total* API name needs to be entered into the picklist.



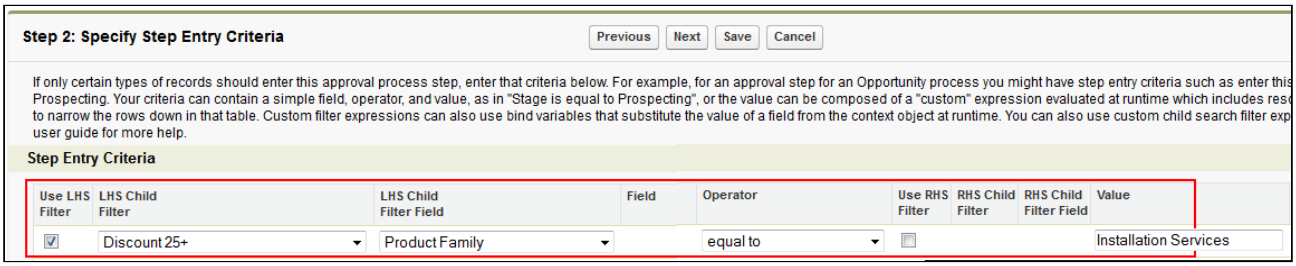
Step 4: Now that AgrDeal Summary Total had been added to the picklist values, a search filter can be created that references it. This is then referenced in the approval step entry criteria. Click **All tabs** and then *Search Filters*, to begin creating the new filter called *Discount 25+*.



The expression you create for the search filter will be used to build the step entry criteria expression. Then when the *Additional Discount* value for an *AgrDeal Summary Total* record is greater than *25%*, the step entry criteria expression will be checked.

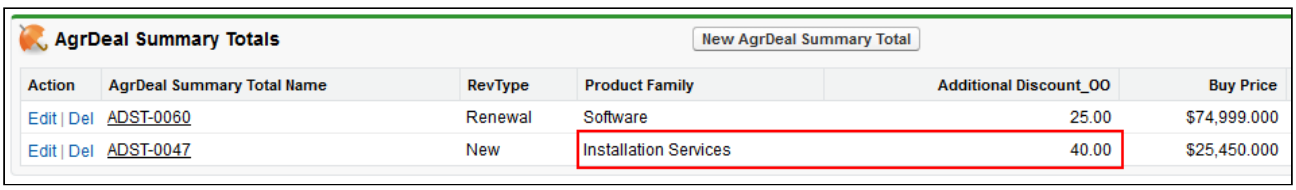


Step 5: Now an approval process step needs to be created which references that search filter. Go to the approval process for an Agreement and create a new step. In *Step 2*, click *Use LHS Filter* and from the *LHS Child Filter* list, select *Discount 25+*. This means only agreement records with an *AgrDeal Summary Total* record that has an *Additional Discount* greater than 25% may enter that step.



Because this step is to route approvals to users who are responsible for the Acme Data product, as well as having a discount of greater than 25%, the AgrDeal Summary Total record's *Product Family* field, must have a value of *Acme Data*. Select an assignee for the step and then save and activate the approval process.

Step 6: Now that the approval process and approval step have been activated, submit an agreement for approval that evaluates as true when compared with the step entry criteria. Here *ADST-0047* has a *40%* discount for the *Installation Services* product family.



This meets the entry criteria as configured above and an approval request email is sent to the assignee for this step.

Configuring the Cart Approvals Page and Apex Class

1. Go to **Setup** > **Develop** > **Apex Classes** and click **New**.
2. Copy and paste the code below and ensure the name of the new class is *CartViewController*.

```
/**
 *  Apttus CPQ Approval
 *  CartViewController
 *
 *  @2012-2013 Apttus Inc. All rights reserved.
 */
public with sharing class CartViewController {
    // context id
    private ID contextId = null;
    // approval type
    private String approvalType = null;
    /**
    * Class Constructor
    * @param stdController the standard controller
    */
    public CartViewController(ApexPages.StandardController stdController) {
        // get context id
        contextId = stdController.getId();

        // get default approval context type
        //approvalType = SystemUtil.getDefaultApprovalCtxType();
        approvalType = 'cart';
    }
    /**
    * Gets context object Id
    */
    public ID getCtxObjectId() {
        return contextId;
    }
    /**
    * Gets cart approval context type
    */
    public String getCtxApprovalType() {
        return approvalType;
```

```
    }
}
```

3. Save the class and then go to **Pages** and click **New**.
4. Copy and paste the following code into the Visualforce Markup window

```
<apex:page standardController= "Apttus_Proposal__Proposal__c"
    extensions= "CartViewController" >
    <apex:form >
        <apex:pageMessages id= "errorPage"  />

        <apex:outputPanel id= "idApprovalContextPanel" >
            <Apttus_CQApprov:ApprovalContextView
contexSOId= "{!ctxObjectId}"
                contextApprovalType= "{!ctxApprovalType}" />
        </apex:outputPanel>
    </apex:form>
</apex:page>
```

5. Ensure that the **Name** is *CartView* and click **Save**.
6. Go to **Create > Objects** and click **Quote/Proposal**.
7. Scroll to **Page Layouts** and click **Edit** for the page layout you want to add Cart Approvals to.
8. From the **Visualforce Pages** window, drag and drop **CartView** onto the page.
9. Display the VisualForce Page Properties for CartView and select **Show scrollbars**.
10. Save the changes to the page.

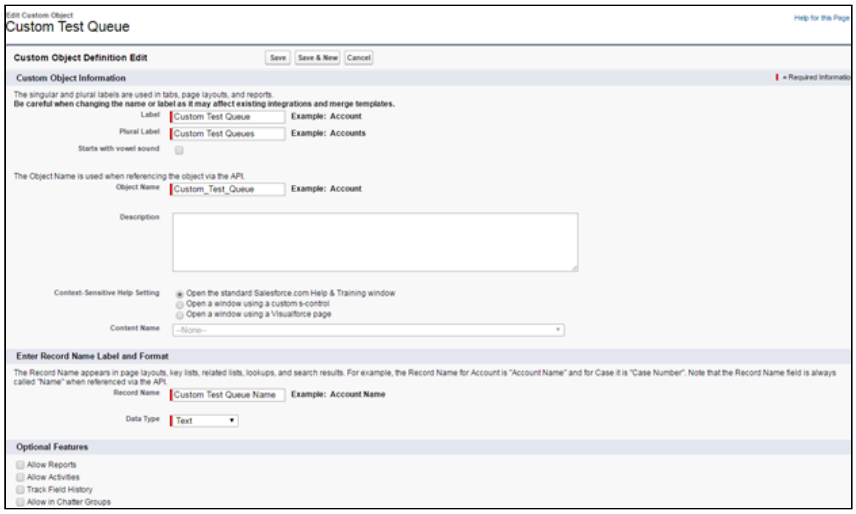
The CartView page acts like a related list for displaying the Cart Approvals, so the next time you go to a Quote/Proposal record that has been submitted for approval from the Cart, the approval steps and associated approvers will be displayed.

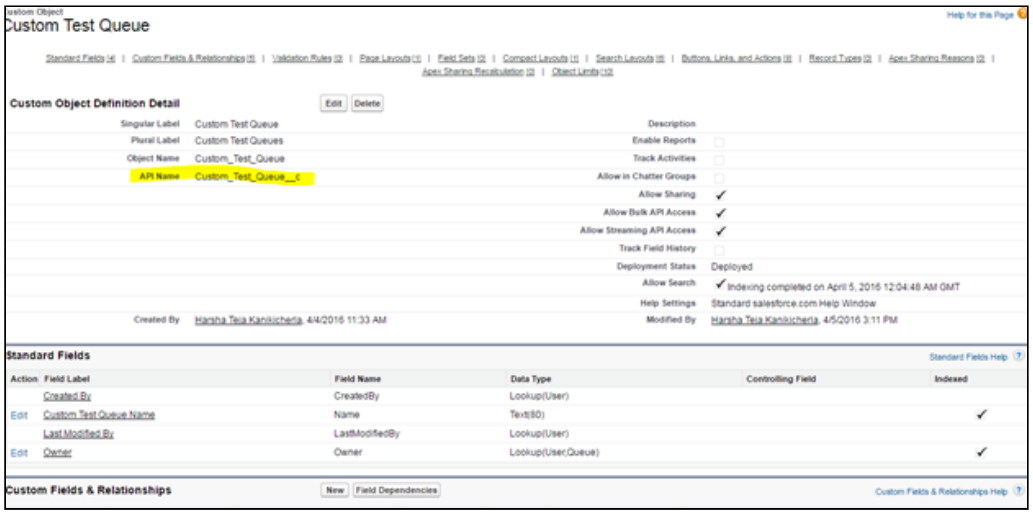
You must finish configuring the other objects before the feature will work as expected.

Setting up Custom Queues

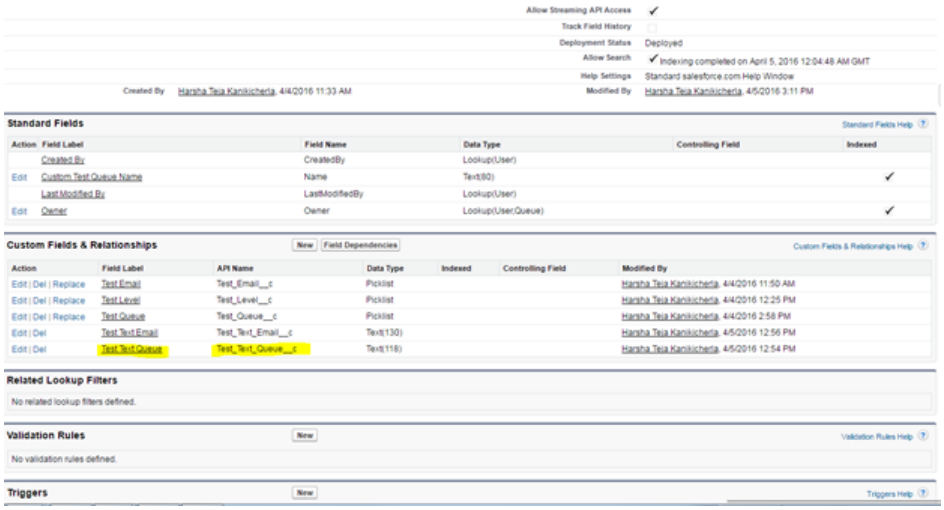
To create Custom Queues:

1. Create a custom object.

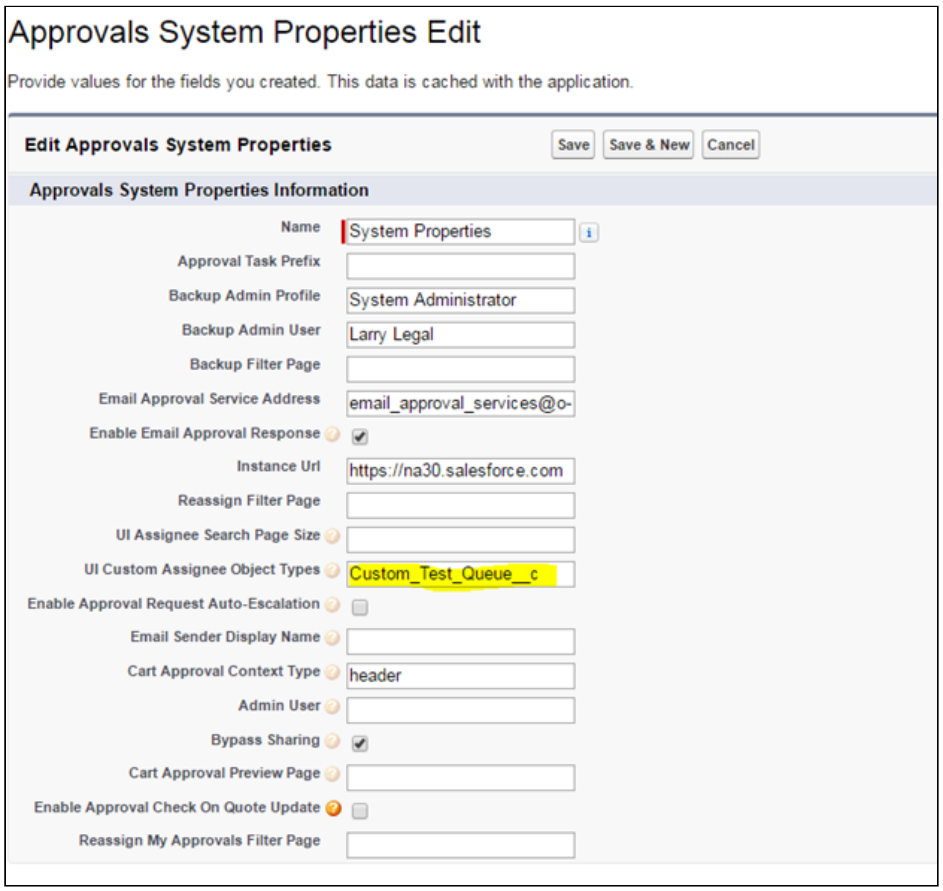




2. Create Custom fields (text fields) in the object to have the queue label (which is used to point out the queue) and other fields to have any other possible details about the queue (If required):
   In the use case a field “Test Level” is created and a value “Level 1” is specified.

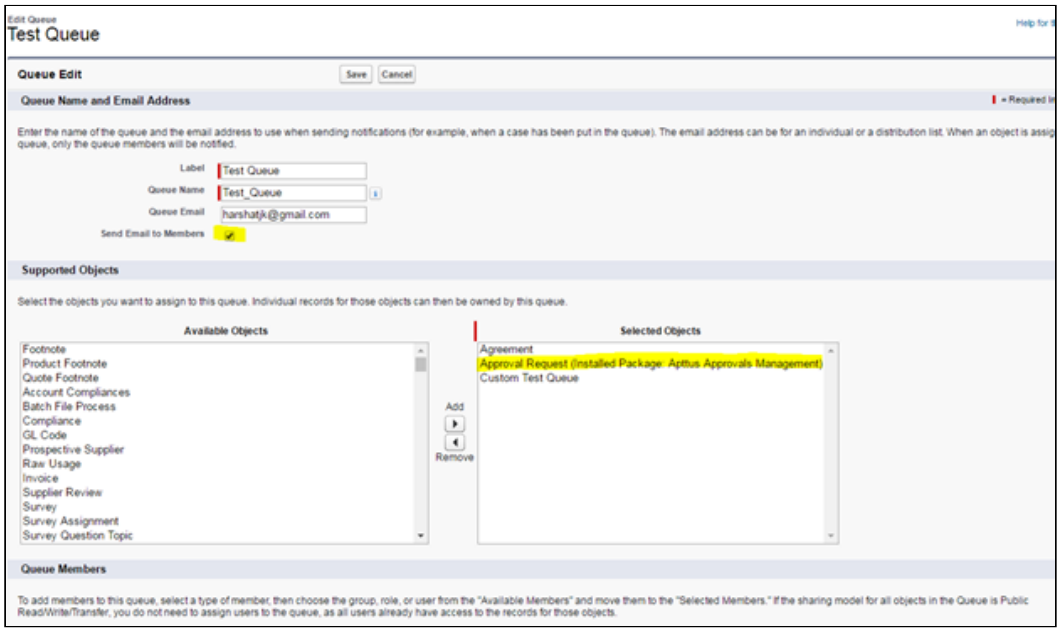


3. Make the custom object available to Intelligent Approvals via a custom setting.
4. From Setup, go to Develop > Custom Settings and click Manage for Approval Systems Properties. Click Edit and for UI Custom Assignee Object Types enter the name of the custom objects you want to use with custom assignees.

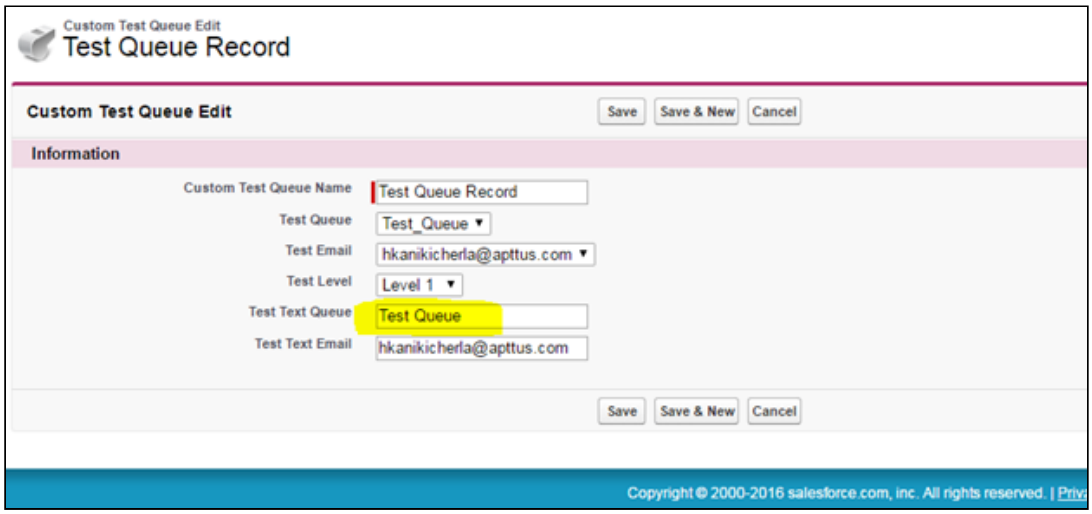


5. The Approval Systems Property must include the API name of the custom object.

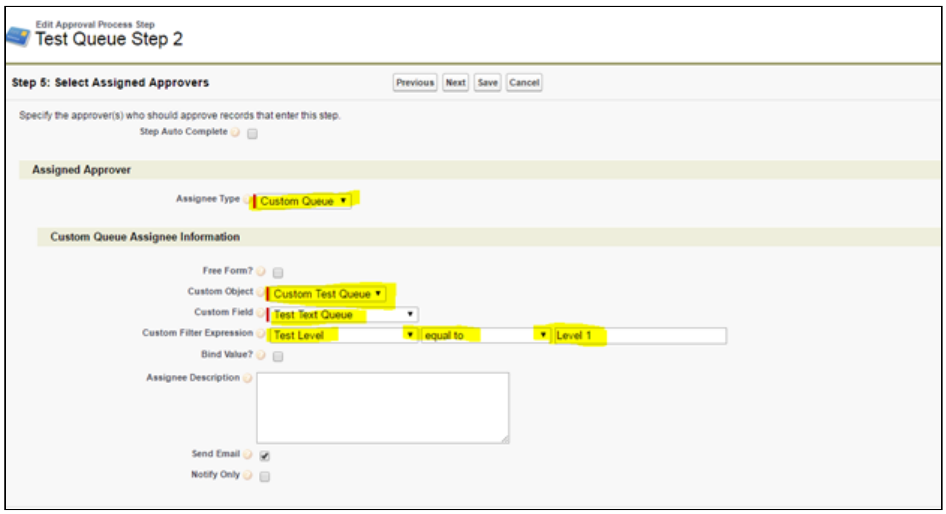
6. Create Test Queue.
   The highlighted ones are the necessary configurations.



7. Send Email to Member's must always be selected in order to send mails to the people in the queue.
8. Approval Request (Installed Package: Conga Approvals) > Object must be selected in order for the approvals to be redirected to the queue.
9. Create records for the custom object:
   Populate the custom text field with Queue label.



10. Create Approval Process and steps.



11. During the Approval Step, while assigning the approvers select the object created as "Custom object", text field where queue label is defined as Custom Field and Custom filter expression has the formula which picks up the queue. You get the formula below after configuring:

```
Custom_Test_Queue__c(Test_Level__c= 'Level 1' ).Test_Text_Queue__c
```

Configuring Post-Installation Settings

This topic contains the following post-installation settings:

- Approval System Properties
- Approvals Custom Config Settings
- Approval Custom Classes
- Setting up the Admin Profiles
- Configuring Global Settings for the Object
- Configuring Email Templates
- Setting the Config Custom Classes
- Configuring Status Picklist values
- Adding Values to Context Object Picklists
- Setting Approval Flag to True
- Creating Search Filters Approvals
- Permission Sets for Accessing Approval Center

Approval System Properties

This section provides a brief description of all the Approval System Properties. Navigate to Custom Settings > (Manage) Approval System Properties > (Edit) System Properties.

To set the Approval & Config System Properties

1. Go to **Setup > Build > Develop > Custom Settings** and click **Manage** for **Approvals System Properties**.
2. Click **Edit** beside **System Properties**.
3. For Cart Approval Context Type, enter one of the following:
    - **cart** - There will be one approval process for all the items in the cart.
    - **header**- The approval process is triggered only for the header items of the cart.
4. Click **Save**.

Setting	Description
Name	System Properties. The default name for the system property settings.
Approval Task Prefix	The prefix for the approval history subject.
Backup Admin Profile	Specify the profile name of the backup administrator who can control the settings and approval profile.
Backup Admin User	Specify the full name or email address of the user who will serve as a backup administrator. ⚠ This is mandatory in the event you deactivate a user from the org. If there is no BackUp Admin User available, the system displays an error. This is also true where the email address being used is active but not the same address for a user in this org.
Backup Filter Page	The custom page to filter and select a backup user.
Bypass Sharing	Indicates whether apex code can bypass record sharing.

Setting	Description
Enable Email Approval Response	Enables the approvers to approve or reject an approval request through e-mail. The approver can write APPROVE or REJECT in the first line of the email. The approval comments can be added from the second line onwards.
UI Assignee Search Page Size	Specify the number of rows the search result shows when you search for user roles, rules, queues, or users. The default is 20 rows.
UI Custom Assignee Object Types	Specify the API names of the standard or custom objects, separated with comma, you want to use when defining custom assignee expressions.
Enable Approval Request Auto Escalation	Selecting this checkbox enables the system to escalate approval requests to users or queues if an approval action is pending for a specific period of time. The users to whom the request is escalated to and the period after which a request is escalated is specified in the approval process.
Email Sender Display Name	Indicates the datasource for the email sender display name. If not specified, defaults to the name of the requester for agreements and defaults to the name of the submitter for all other cases. The valid values are: • Approver • Requester • Submitter
Enable Auto-Reapproval On Finalized Cart	Setting this flag to true allows previous approvals from the finalized cart to be included in the check for auto-reapproval, otherwise re-approvals start from scratch on the new cart object.

Setting	Description
Cart Approval Context Type	Specify the context for which an approval process is applicable. The applicable values are: • Cart - triggers an approval process for the entire cart • Header - triggers an approval process for the header items of the cart only • LineItem - triggers an approval process for the line items of the cart.
Admin User	Specify the name of the Admin User who is the default owner of activities created by a user who is not allowed to be the owner (for example: customer portal user). Format = first name, last name.
Cart Approval Preview Page	Specify the name of the custom preview cart page that you want to use if you want to define and use a custom page to preview approvals. Leave it blank to use the default page.
Email Approval Service Address	The email service address that can receive messages for processing.
Enable Approval Check On Quote Update	Is run by default for CPQ Approvals. Approval required check is run, whenever the Quote is updated. This flag enables the system to evaluate the approval process and if the entry criteria is satisfied, changes the status to Approval Required.
Enable Resubmit	Enables you to define the behavior of the Submit for Approvals button in the Approvals related list of your object. This system property replaces the CancelPendingProcess parameter on the Preview and Submit formula button on your object. When selected, and user clicks Submit for Approvals, current approval requests are cancelled and new approval requests are resubmitted. When deselected, and the and user clicks Submit for Approvals, the user cannot cancel and resubmit and ongoing approval request.

Setting	Description
Enable Upload Attachment From Desktop	Enables you to browse and attach external documents from your local machine for **Submit with Attachments** action on Preview approvals page. This eliminates your effort to attach the required documents in Notes & Attachments related list of your object record before submitting the record for approval.
Inbound Attachment Callback	Inbound attachment callback class name. The class should implement the CustomClass.IInboundAttachmentCallback interface.
Instance Url	The Salesforce instance url for redirecting to custom pages.
Reassign Filter Page	The custom page to filter and select a reassign user.
Reassign My Approvals Filter Page	
Cancellation Email to Approved/ Notified	If this flag is set to true, cancellation email is sent for approved/notified requests when approval requests are recalled.
Queueable Apex Batch Size	Specifies the number of records to be processed by a single queueable apex batch job. ⚠ This setting is deprecated. Conga recommends that you do not use it. This property is not superseded or replaced with any other system property.
Sync Submit Request Threshold	Specifies the number of records to submit the approvals in synchronous mode. If the number of approval records cross this threshold, an asynchronous job is submitted in the background to submit the approvals. The default value is 100. If the number of records to submit is less than 100, the submission happens in a synchronous mode. Auto-Complete and Notify feature works in a synchronous mode. ⚠ This setting is deprecated. Conga recommends that you use **Process Submit in Aysnc Mode** instead.

Setting	Description
Sync Approve Request Threshold	Specifies the number of approval records to approve or reject in a synchronous mode. If the number of approval records cross this threshold, an asynchronous job is submitted in the background to process the approvals. The default value is 100. If the number of records to approve or reject is less than 100, the processing happens in a synchronous mode.
Email Sender Org Wide Display Name Used	Enable this setting to use an org-wide email address to send approval emails. If you disable it, the email address of the user who submitted the approval request is used.
Email Org Wide Display Name	Enter the display name of the org-wide email address to use when sending approval emails.
Process Submit in Aysnc Mode	Enable this setting to submit the approval records in asynchronous mode. If you select it, an asynchronous batch job is submitted in the background to process the approval requests.
Approval Center	Enter comma-separated values of context object API names to view them on Approval Center. For example, *Apttus__APTS_Agreement__c, Apttus__Agreement_Clause__c, Apttus_Config2__LineItem__c, Apttus_Config2__ProductConfiguration__c* The first object name is displayed as the default tab if there are approval requests associated with it.

Approvals Custom Config Settings

These custom settings are used to set the default pages, templates, and backup admin user for the system. The same settings are used for each business context object (Agreement, Opportunity, Quote, etc...) that you are using with Approvals.

Setting	Description
Approval Context Type	The approval context type associated with the process for the object. The valid values are: • Single • Multiple
Approval Group Status Field	The API name of the approval group status field. Only applicable to custom context objects. If not specified, defaults to "Approval_Group_Status__c".
Backup Admin User	As well as the global backup admin user that is designated via the Approvals System Properties, you can also use this setting to designate a backup admin user for each context object. If you do not select a user here, the global backup admin user will be used. For more details, see Setting up the Backup Administrator.
My Approvals Page	Name of the My Approvals page used for showing consolidated approval requests.
Approval Summary Page	This is the default summary page for the context object and overrides the global default page. Approval summary pages contain the information and action items that enable a record to be reviewed and approval actions to be taken. The default summary pages provide sufficient information for agreements, opportunities, quote/proposals, proposal line items, and term exceptions; however, you can create a custom page and reference it here.
Preview Page	Name of the Preview page used for showing consolidated approval requests preview.

Setting	Description
Assignment Email Template	This is the default email template used to send notifications when an approval request is submitted. This is the global default if not overridden for the approval process. For more details on templates, see Setting up Email Templates and Alerts. ⚠ This is a mandatory field. This field cannot be left blank.
Reassignment Email Template	This is the default email template used to send notifications when an approval request is reassigned. This is the global default if not overridden for the approval process. For more details on templates, see Setting up Email Templates and Alerts. ⚠ This is a mandatory field. This field cannot be left blank.
Escalation Email Template	This is the default email template used to send notifications when an approval request is escalated. This is the global default if not overridden for the approval process. For more details on templates, see Setting up Email Templates and Alerts. ⚠ This is a mandatory field. This field cannot be left blank.
NotifyOnly Email Template	This is the default email template used to send notifications when a notify only process step is executed. This is the global default if not overridden for the approval process. For more details on templates, see Setting up Email Templates and Alerts. ⚠ This is a mandatory field. This field cannot be left blank.

Setting	Description
Cancellation Email Template	This is the default email template used to send notifications when an approval request is cancelled. This is the global default if not overridden for the approval process. For more details on templates, see Setting up Email Templates and Alerts. ⚠ This is a mandatory field. This field cannot be left blank.
Reminder Email Template	This is the default email template used to send notifications when an approval request needs a reminder.
Enable Adhoc Approval	Indicates whether Adhoc Approval is enabled for this object.
Approval Status Field	The API name of the approval status field. Only applicable to custom context objects. This is for using custom objects with Approvals where the *Approval Status* field needs to be displayed on the custom object records. If not specified, defaults to "Approval_Status__c". It is not used for standard supported objects.
Use Files Instead of Notes & Attachments	When selected, the approvals engine queries all attachments from Files instead of Notes & Attachments. For example, you create a quote and attach a file at the Quote level. You must select this flag on the Proposal object.
Parent Reference Field	Parent object reference.
Child Object Types	The list of child object types associated with the approval process for the parent object. This list is automatically maintained by the application.
Child Object Types2	The list of child object types associated with the approval process for the parent object. This list is automatically maintained by the application.

Setting	Description
Include Child Obj Attachments For Submit	Indicates whether or not to include attachments associated with the current context object's children in the attachment selection screen when submitting for approval.
Show All Document Version Details	Select this setting to view all the agreement versions from the document version related list. If you do not select it, only the latest version of the agreement is available. By default, the setting is set to false.

Approval Custom Classes

Approval custom classes used to register custom classes that Implement Global interfaces exposed by Conga for customization.

Classes	Purpose of the class
Inbound Attachment Callback Class	Inbound attachment callback class name. The class should implement the CustomClass.IInboundAttachmentCallback interface.
Query Callback Class	Query callback class name. The class should implement the CustomClass.IQueryCallback interface.

Custom Query Callback for Reassigning and Adding Approver

Previously, the default behavior for reassigning and adding an approver queries from the My Approvals page was as follows:

- Reassigning Approver default behavior - Selects all users
- Add Approver default behavior - Selects all users, queues, and roles

From Summer 2018 release, as an Approvals admin, you can determine certain predefined subset of users, groups, queues or roles to be added as adhoc or additional users to the list of approvals for in-flight approval requests. Support has also been added for custom callback code to filter query for users, roles and queues for reassigning and adding an Approver on the My Approvals page.

In the Summer 2019 release, we have added the ability to reassign to queues and roles as well as users. When you select re-assignment to a Queue or a Role, a list of queues or roles is dynamically populated via a drop-down. The new default behavior available is:

- Reassigning Approver default behavior - Selects all users, queues, or roles
- Add Approver default behavior - Selects all users, queues, or roles

You can reassign an Approval Request to an User, Queue or Role. You can further customize the out-of-the-box behavior by overriding the Approval System Properties to filter list of users, queues, and roles from the reassign user page accessed by the Reassign link from the Approval Requests related list, and by overriding the DefaultQueryCallback class used to filter users, queues and roles from the Reassign and Add Approver icons on the My Approvals page.

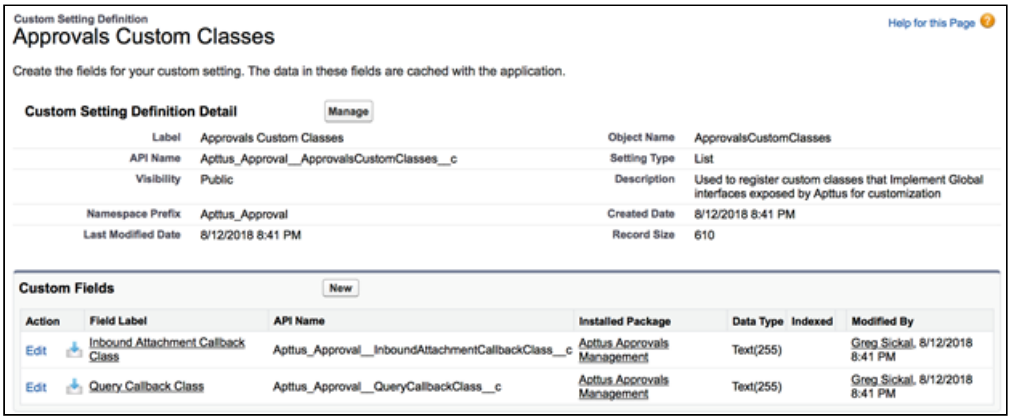
If you want to change the default behavior from selecting all users, queues, and roles to only selecting a subset of them, you can create a custom apex callback class that contains custom code to query users, queues, and roles when reassigning users or adding adhoc approvers from the My Approvals page. Your code must implement the IQueryCallback and IQueryCallback2 interfaces and define the following six methods if you want to override the default behavior which is to select all available users, queues and roles. Note that since there are individual methods to select users, queues, or roles for reassigning or adding new approvers, you can display a different set of records for reassign approvers than adding them if you wanted to. Or your code can call a common method to return the same set of records for both.

```
/**
     *  Apttus Approvals Management
     *  IQueryCallback
     *
     *  @2018-2019 Apttus Inc. All rights reserved.
     */
    global  interface  IQueryCallback {

         /**
         * Callback invoked to perform a filtered query when reassigning
user approvers
         * @param request the approval request context object
         * @param searchText the prefilter search text
         * @return list of sobjects returned from query
         */
        List<sObject> doReassignUserQuery(Approval_Request__c request,
String searchText);
```

```
        /**
         * Callback invoked to perform a filtered query when adding user
approvers
         * @param request the approval request context object
         * @param searchText the prefilter search text
         * @return list of sobjects returned from query
         */
        List<sObject> doAddUserQuery(Approval_Request__c request, String
searchText);

        /**
         * Callback invoked to perform a filtered query when adding queue
approvers
         * @param request the approval request context object
         * @param searchText the prefilter search text
         * @return list of sobjects returned from query
         */
        List<sObject> doAddQueueQuery(Approval_Request__c request, String
searchText);

        /**
         * Callback invoked to perform a filtered query when adding role
approvers
         * @param request the approval request context object
         * @param searchText the prefilter search text
         * @return list of sobjects returned from query
         */
        List<sObject> doAddRoleQuery(Approval_Request__c request, String
searchText);

    }

    /**
     *  Apttus Approvals Management
     *  IQueryCallback2
     *
     *  @2018-2019 Apttus Inc. All rights reserved.
     */
    global interface IQueryCallback2 extends IQueryCallback {

        /**
         * Callback invoked to perform a filtered query when reassigning
queue approvers
         * @param request the approval request context object
```

```
         * @param searchText the prefilter search text
         * @return list of sobjects returned from query
         */
        List<sObject> doReassignQueueQuery(Approval_Request__c request,
String searchText);

         /**
         * Callback invoked to perform a filtered query when reassigning
role approvers
         * @param request the approval request context object
         * @param searchText the prefilter search text
         * @return list of sobjects returned from query
         */
        List<sObject> doReassignRoleQuery(Approval_Request__c request,
String searchText);

    }
```

Save the custom class name in the “Query Callback Class” custom field of the “Approvals Custom Classes” custom setting:



To override the default behavior

- From the Approval System Properties, in the **Reassign Filter Page** custom setting specify the name of the visual force page that lets users select from a filtered list of users, queues, and roles.

To specify the custom reassign user search, do the following:

1. Go to **Setup > Develop > Custom Settings** and click **Manage** for Approval System Properties.

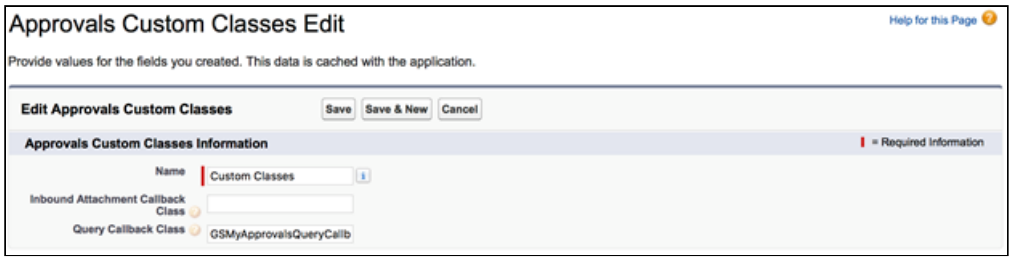
2. In **Reassign Filter Page** specify the name of the visual force page.
3. Click **Save**.

To override the Default Query Callback class

1. Go to **Setup > Develop > Custom Settings** and click **Manage** for *Approvals Custom Classes*.
2. Click **Edit** for Custom Classes.

   > ⚠ If there is no existing entry, you should create one. Click **New** and in **Name**, type Custom Classes.

3. In Query Callback Class, type the name of your callback class. For example, using the custom class in the Use Case below, enter "GSMyApprovalsQueryCallback" as the query callback class name:



4. Click **Save**.

Use Case

For example, here is an implementation of a custom callback that implements custom filters for reassigning and adding approver users, adding approver users, queues and roles. This is just a sample implementation. Your custom class can be simpler or much more complex. Each method takes two parameters: the approval request context object the user has selected to reassign or add approvers for and search text entered in the search box used to perform additional record filtering. Each of the methods in your callback class needs to return a list of Users, QueueSObjects, or UserRoles in each of the three methods.

> ⚠ You must define your class as global for the callback mechanism to work.

```
/**
 *  Apttus Approvals Management
 *  GSMyApprovalsQueryCallback
 *
 *  @2018-2019 Apttus Inc. All rights reserved.
```

```
 */
global with sharing class GSMyApprovalsQueryCallback
    implements Apttus_Approval.CustomClass.IQueryCallback,
               Apttus_Approval.CustomClass.IQueryCallback2 {

    // limit rows
    public static final Integer LIMIT_ROWS = 1000;

    /**
    * Callback invoked to perform a filtered query when reassigning user
approvers
    * @param request the approval request context object
    * @param searchText the prefilter search text
    * @return list of sobjects returned from query
    */
    public List<sObject>
doReassignUserQuery(Apttus_Approval__Approval_Request__c request, String
searchText) {
        try {
            String sq = '\'';
            String qryStr = 'SELECT Id, Name, Username, LastName,
FirstName, Email, UserRoleId, UserRole.Name, IsActive, Profile.Name ';
            qryStr += ', (select Id, ';
            qryStr += 'Apttus_Approval__Approval_Level__c, ';
            qryStr += 'Apttus_Approval__Description__c';
            qryStr += ' from ' + 'Apttus_Approval__Approval_Matrices__r)';
            qryStr += ' FROM User ';
            qryStr += ' WHERE IsActive = true ';

            if (!nullOrEmpty(searchText)) {
                // build the search term
                String searchTerm = (searchText != null ? searchText : '')
.trim();

                // append wildcards to the search term
                searchTerm = (!searchTerm.startsWith('%') ? '%' +
searchTerm : searchTerm);
                searchTerm = (!searchTerm.endsWith('%') ? searchTerm +
'%' : searchTerm);

                qryStr += 'AND ';
                qryStr += 'Name LIKE ' + sq +
String.escapeSingleQuotes(searchTerm) + sq;
```

```
            }

            // EXAMPLE: filter out users with names not containing 'Borden'
and system admin profiles
            qryStr += ' AND (NOT Name LIKE ' + sq + 'Borden%' + sq +
')' ;
            //qryStr += ' AND Profile.Name = ' + sq + 'System
Administrator' + sq;
            qryStr += ' ORDER BY Name' ;

            // limit rows
            qryStr += ' LIMIT ' + LIMIT_ROWS;

            // execute the query
            return Database.query(qryStr);

        } catch (Exception ex) {
            system.debug(LoggingLevel.ERROR, ex.getMessage());
            return new List<sObject>();

        }

    }

    /**
     * Callback invoked to perform a filtered query when reassigning queue
approvers
     * @param request the approval request context object
     * @param searchText the prefilter search text
     * @return list of sobjects returned from query
     */
    public List<sObject>
doReassignQueueQuery(Apttus_Approval__Approval_Request__c request, String
searchText) {
        try {
            String sq = '\'' ;
            String qryStr = 'SELECT Id, QueueId, Queue.Id, Queue.Name,
Queue.DeveloperName, Queue.Type, Queue.Email' ;
            qryStr += ' FROM QueueSobject ' ;
            qryStr += ' WHERE ' ;

            if (!nullOrEmpty(searchText)) {
                // build the search term
                String searchTerm = (searchText != null ? searchText : '' )
.trim();
```

```
                // append wildcards to the search term
                searchTerm = (!searchTerm.startsWith( '%' ) ?  '%'  +
searchTerm : searchTerm);
                searchTerm = (!searchTerm.endsWith( '%' ) ? searchTerm +
'%'  : searchTerm);

                qryStr +=  'Queue.Name LIKE '  + sq +
String.escapeSingleQuotes(searchTerm) + sq;
                qryStr +=  ' AND SObjectType = '  + sq +
'Apttus_Approval__Approval_Request__c'  + sq;
                qryStr +=  ' AND' ;
            }

             // EXAMPLE: filter out queues with names containing 'Approvals
BQA'
            qryStr +=  ' Queue.Name LIKE '  + sq +  'Approvals BQA%'  + sq;
            qryStr +=  ' AND SObjectType = '  + sq +
'Apttus_Approval__Approval_Request__c'  + sq;
            qryStr +=  ' ORDER BY Queue.Name' ;

             // limit rows
            qryStr +=  ' LIMIT '  + LIMIT_ROWS;

             // execute the query
             return  Database.query(qryStr);

        }  catch  (Exception ex) {
            system.debug(LoggingLevel.ERROR, ex.getMessage());
             return   new  List<sObject>();

        }

    }

     /**
     * Callback invoked to perform a filtered query when reassigning role
approvers
     * @param request the approval request context object
     * @param searchText the prefilter search text
     * @return list of sobjects returned from query
     */
```

```
    public List<sObject>
doReassignRoleQuery(Apttus_Approval__Approval_Request__c request, String
searchText) {
        try {
            String sq = '\'';
            String qryStr = 'SELECT Id, Name, DeveloperName,
RollupDescription';
            qryStr += ' FROM UserRole ';
            qryStr += ' WHERE ';

            if (!nullOrEmpty(searchText)) {
                // build the search term
                String searchTerm = (searchText != null ? searchText : '')
.trim();

                // append wildcards to the search term
                searchTerm = (!searchTerm.startsWith('%') ? '%' +
searchTerm : searchTerm);
                searchTerm = (!searchTerm.endsWith('%') ? searchTerm +
'%' : searchTerm);

                qryStr += 'UserRole.Name LIKE ' + sq +
String.escapeSingleQuotes(searchTerm) + sq;
                qryStr += ' AND';
            }

            // EXAMPLE: filter out roles with names containing Sales
            qryStr += ' UserRole.Name LIKE ' + sq + 'Sales%' + sq;
            qryStr += ' ORDER BY UserRole.Name';

            // limit rows
            qryStr += ' LIMIT ' + LIMIT_ROWS;

            // execute the query
            return Database.query(qryStr);

        } catch (Exception ex) {
            system.debug(LoggingLevel.ERROR, ex.getMessage());
            return new List<sObject>();

        }

    }

    /**
```

```
     * Callback invoked to perform a filtered query when adding user
approvers
     * @param request the approval request context object
     * @param searchText the prefilter search text
     * @return list of sobjects returned from query
     */
     public  List<sObject>
doAddUserQuery(Apttus_Approval__Approval_Request__c request, String
searchText) {
         try  {
            String sq =  '\'' ;
            String qryStr =  'SELECT Id, Name, Username, LastName,
FirstName, Email, UserRoleId, UserRole.Name, IsActive, Profile.Name ' ;
            qryStr +=  ', (select Id, ' ;
            qryStr +=  'Apttus_Approval__Approval_Level__c, ' ;
            qryStr +=  'Apttus_Approval__Description__c' ;
            qryStr +=  ' from '  +  'Apttus_Approval__Approval_Matrices__r)' ;
            qryStr +=  ' FROM User ' ;
            qryStr +=  ' WHERE IsActive = true ' ;

             if  (!nullOrEmpty(searchText)) {
                 // build the search term
                String searchTerm = (searchText !=  null  ? searchText :  '' )
.trim();

                 // append wildcards to the search term
                searchTerm = (!searchTerm.startsWith( '%' ) ?  '%'  +
searchTerm : searchTerm);
                searchTerm = (!searchTerm.endsWith( '%' ) ? searchTerm +
 '%'  : searchTerm);

                qryStr +=  'AND ' ;
                qryStr +=  'Name LIKE '  + sq +
String.escapeSingleQuotes(searchTerm) + sq;
            }

             // EXAMPLE: filter out users with names not containing 'Borden'
and system admin profiles
            qryStr +=  ' AND (NOT Name LIKE '  + sq +  'Borden%'  + sq +
 ')' ;
             //qryStr += ' AND Profile.Name = ' + sq + 'System
Administrator' + sq;
            qryStr +=  ' ORDER BY Name' ;
```

```
            // limit rows
            qryStr += ' LIMIT ' + LIMIT_ROWS;

            // execute the query
            return Database.query(qryStr);

        } catch (Exception ex) {
            system.debug(LoggingLevel.ERROR, ex.getMessage());
            return new List<sObject>();

        }

    }

    /**
     * Callback invoked to perform a filtered query when adding queue
approvers
     * @param request the approval request context object
     * @param searchText the prefilter search text
     * @return list of sobjects returned from query
     */
    public List<sObject>
doAddQueueQuery(Apttus_Approval__Approval_Request__c request, String
searchText) {
        try {
            String sq = '\'';
            String qryStr = 'SELECT Id, QueueId, Queue.Id, Queue.Name,
Queue.DeveloperName, Queue.Type, Queue.Email';
            qryStr += ' FROM QueueSobject ';
            qryStr += ' WHERE ';

            if (!nullOrEmpty(searchText)) {
                // build the search term
                String searchTerm = (searchText != null ? searchText : '' )
.trim();

                // append wildcards to the search term
                searchTerm = (!searchTerm.startsWith( '%' ) ? '%' +
searchTerm : searchTerm);
                searchTerm = (!searchTerm.endsWith( '%' ) ? searchTerm +
'%' : searchTerm);

                qryStr += 'Queue.Name LIKE ' + sq +
String.escapeSingleQuotes(searchTerm) + sq;
```

```
                qryStr +=  ' AND SObjectType = '  + sq +
'Apttus_Approval__Approval_Request__c'  + sq;
                qryStr +=  ' AND' ;
            }

             // EXAMPLE: filter out queues with names containing 'BO'
            qryStr +=  ' Queue.Name LIKE '  + sq +  'BO%'  + sq;
            qryStr +=  ' AND SObjectType = '  + sq +
'Apttus_Approval__Approval_Request__c'  + sq;
            qryStr +=  ' ORDER BY Queue.Name' ;

             // limit rows
            qryStr +=  ' LIMIT '  + LIMIT_ROWS;

             // execute the query
             return  Database.query(qryStr);

        }  catch  (Exception ex) {
            system.debug(LoggingLevel.ERROR, ex.getMessage());
             return   new  List<sObject>();

        }

    }

     /**
     * Callback invoked to perform a filtered query when adding role
approvers
     * @param request the approval request context object
     * @param searchText the prefilter search text
     * @return list of sobjects returned from query
     */
     public  List<sObject>
doAddRoleQuery(Apttus_Approval__Approval_Request__c request, String
searchText) {
         try  {
            String sq =  '\'' ;
            String qryStr =  'SELECT Id, Name, DeveloperName,
RollupDescription' ;
            qryStr +=  ' FROM UserRole ' ;
            qryStr +=  ' WHERE ' ;

             if  (!nullOrEmpty(searchText)) {
```

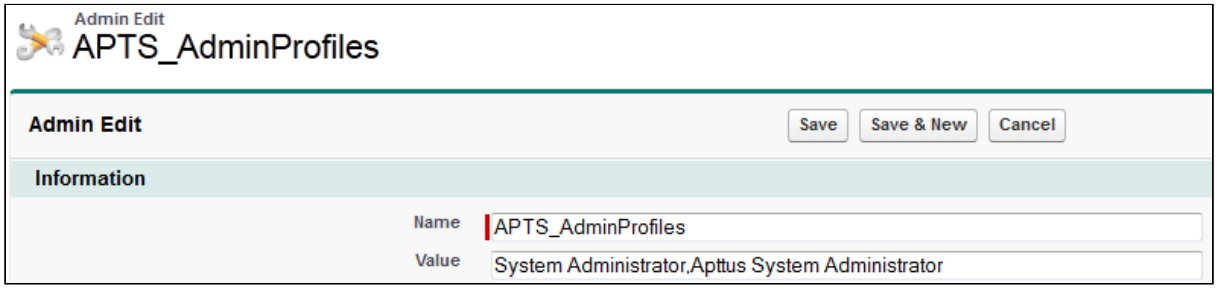
```
                // build the search term
                String searchTerm = (searchText != null ? searchText : '' )
.trim();

                // append wildcards to the search term
                searchTerm = (!searchTerm.startsWith( '%' ) ? '%' +
searchTerm : searchTerm);
                searchTerm = (!searchTerm.endsWith( '%' ) ? searchTerm +
'%' : searchTerm);

                qryStr += 'UserRole.Name LIKE ' + sq +
String.escapeSingleQuotes(searchTerm) + sq;
                qryStr += ' AND' ;
            }

            // EXAMPLE: filter out roles with names containing Role
            qryStr += ' UserRole.Name LIKE ' + sq + 'Role%' + sq;
            qryStr += ' ORDER BY UserRole.Name' ;

            // limit rows
            qryStr += ' LIMIT ' + LIMIT_ROWS;

            // execute the query
            return Database.query(qryStr);

        } catch (Exception ex) {
            system.debug(LoggingLevel.ERROR, ex.getMessage());
            return new List<sObject>();

        }

    }

    /**
    * 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 );

    }
```

Setting up the Admin Profiles

Admin profiles enable you to associate users – via their roles – with admin level access to Approvals.

To set up Admin Profiles in Approvals,

1. Select the **Admin** tab.
   - or -
   Select ⊞ (All tabs) and select **Admin**.
2. Click **Go** to display all of the admin settings.
3. If **APTS_AdminProfiles** is listed, click **Edit**, enter the required profile names, separated by a comma, in the **Value** field and click **Save**.
   - or -
   If the setting is not displayed, click **New Admin**.
4. Enter the name **APTS_AdminProfiles**, enter the required profile names, separated by a comma, in the **Value** field, and click **Save**.



Users associated with the profiles can now complete administrative tasks for Approvals.

Configuring Global Settings for the Object

Approvals custom configuration settings provide the global default values for using Approvals with different objects.

The backup admin user, summary page, and email templates are designated here so that they do not need to be selected each time you create an approval process. You can choose to override them for specific processes, but this ensures approval requests will work correctly regardless.

To configure global settings

You must have created the required custom pages and templates.

1. Go to **Build > Develop > Custom Settings** and click **Approvals Custom Config**.
2. From the **Custom Setting Definition Detail** section, click **Manage** to display the current objects associated with Approvals.
3. If it does not already exist, click **New** and enter `API name of the Object` in the **Name** field, which is the API name for the object to be used with approvals.
4. In the **Approval Status Field**, enter `Approval_Status__c` or whatever the API name is for the picklist created for the custom object.
5. The custom summary page and templates you created should be entered in the **Approval Summary Page** and various **Email Template** fields. If you do not enter values, the default system values will be used.
6. Specify the Approval Context Type as Single or Multiple. If you want an approval process to be triggered for Header and the Line items, enter Multiple. If you want to trigger an approval process for the header items only enter type as Single.
7. Specify the Child Object Types if child object types are associated in the approval process.
8. Click **Save**.

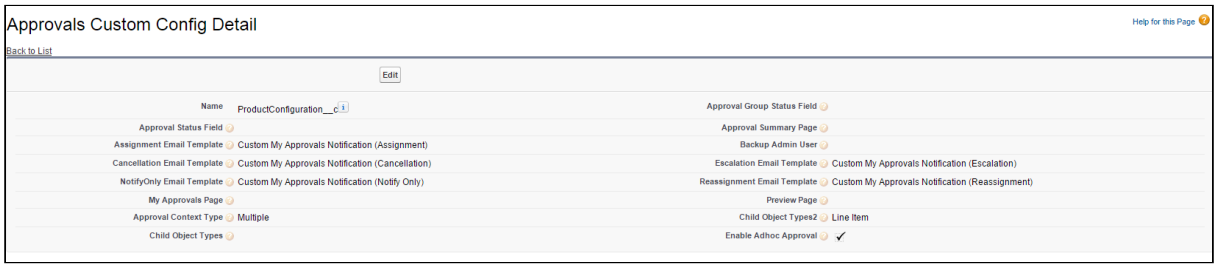
The page and templates will be used by default for the next approval process created for the context object.

When creating an approval process you can override these templates if necessary.

To configure object settings

You must have created the required custom pages and templates.

1. Go to **Build > Develop > Custom Settings** and click **Approvals Custom Config**.
2. From the **Custom Setting Definition Detail** section, click **Manage** to display the current objects associated with Approvals.
3. If it does not already exist, click **New** and enter `ProductConfiguration__c` in the **Name** field, which is the API name for the object to be used with approvals.
4. In the **Approval Status Field**, enter `Approval_Status__c` or whatever the API name is for the picklist created for the custom object.
5. The custom summary page and templates you created should be entered in the **Approval Summary Page** and various **Email Template** fields. If you do not enter values, the default system values will be used.



6. Specify the Approval Context Type as **Single** or **Multiple**. If you want to use Header and line item approvals, set the approval context type to **Multiple**. For header only approvals, set the context type to **Single**.
7. Specify the Child Object Type. The Child object type is associated with the approval process for a parent object.
8. You can enable ad hoc approvals for a custom object by selecting Enable Ad Hoc Approvals.
9. Click **Save**.

The page and templates will be used by default for the next approval process created for the custom context object.

When creating an approval process you can override these templates if necessary.

Configuring Email Templates

Typically the easiest way to create your own custom template is to clone one of the email templates provided in the unmanaged Conga samples package and then edit it. After that has been done you can then override the default email templates for all of your approval processes using APTS_ApprovalConfig or on an ad-hoc basis for specific approval requests.

The Conga Approvals Management (Customization Samples) package provides email template examples for currently supported objects, as well as for Salesforce Cases, which can subsequently be used as the basis for creating email templates for custom objects you've configured yourself. Also note that Opportunity Product is also used to provide templates for further customization.

You have to specify the email templates in the Approval Process and in the Approvals Custom Config custom setting.

As well as providing samples for templates and the summary page, the pages that are used to provide approval functionality on the custom object record page are also included.

As this package is unmanaged, it cannot be upgraded. If you want to move to the newest version, you must uninstall the existing package and then install the new package. It is recommended you uninstall the samples package after you have created your custom templates and pages.
Queue email address cannot be used for notifications. If Queue has "Send Email to Members" box un-checked - email of current user will be used for notifications (not queue email address). This is due to Salesforce limitations which we cannot bypass.

Installing Customization Samples package

The Conga Approvals Management (Customization Samples) package contains email templates, components, Visualforce pages and Apex controllers which can help you build you templates and approval summary pages for custom context objects.

The samples contained in the package should not be used 'as is' and are only to be used in the development of your own bespoke templates and pages, and should not be installed in production.

Configuring Email Templates

For a process with header and child process steps, use the Conga Approval Email Templates. Notification templates are used to create the notifications sent to approvers to alert them to pending approvals and to allow them to approve or decline a request without logging in to Salesforce.

The following approval email notification templates are available as a part of the standard Approvals Management package. These are generic email notification templates that can be used for approvals on any standard or custom objects without any change. However, these notification templates can be customized as needed. Clone these templates if you want to customize your own template.

To view the Quote Header name in the My Approvals page, you must change your existing process definitions or custom config settings to use the new templates provided with this release. If you have already customized an email template to use with CPQ Cart Approvals, you need to include the MyCartApprovalsEmail class inside your custom Visualforce template to correctly launch the My Approvals page that shows the quote id and name fields.

Sample	Component Type	Description
Apttus My Approvals Notification (Assignment)	Visualforce Email Template	This template is used to notify users they must decide on an approval request.
Apttus My Approvals Notification (Reassignment)	Visualforce Email Template	This template is used to notify users that an approval request task initially assigned to them has been reassigned to someone else.
Apttus My Approvals Notification (Escalation)	Visualforce Email Template	This template is used for auto-escalation. Configuration is required to activate this feature.
Apttus My Approvals Notification (Cancellation)	Visualforce Email Template	This template is used to notify users that an approval request has been cancelled.
Apttus My Approvals Notification (Notify Only)	Visualforce Email Template	This template is used to notify users only.

Approval Email Notification Templates

The following approval email notification templates are available as a part of the standard Approvals Management package. These are generic email notification templates that can be used for approvals on any standard or custom objects without any change. However, these notification templates can be customized as needed. Clone these templates if you want to customize your own template.

Sample	Component Type	Description
Apttus My Approvals Notification (Assignment)	Visualforce Email Template	This template is used to notify users they must decide on an approval request.

Sample	Component Type	Description
Apttus My Approvals Notification (Reassignment)	Visualforce Email Template	This template is used to notify users that an approval request task initially assigned to them has been reassigned to someone else.
Apttus My Approvals Notification (Escalation)	Visualforce Email Template	This template is used for auto-escalation. Configuration is required to activate this feature.
Apttus My Approvals Notification (Cancellation)	Visualforce Email Template	This template is used to notify users that an approval request has been cancelled.
Apttus My Approvals Notification (Notify Only)	Visualforce Email Template	This template is used to notify users only.
Apttus My Approvals Notification (Reminder)	Visualforce Email Template	This template is used to send reminder emails.

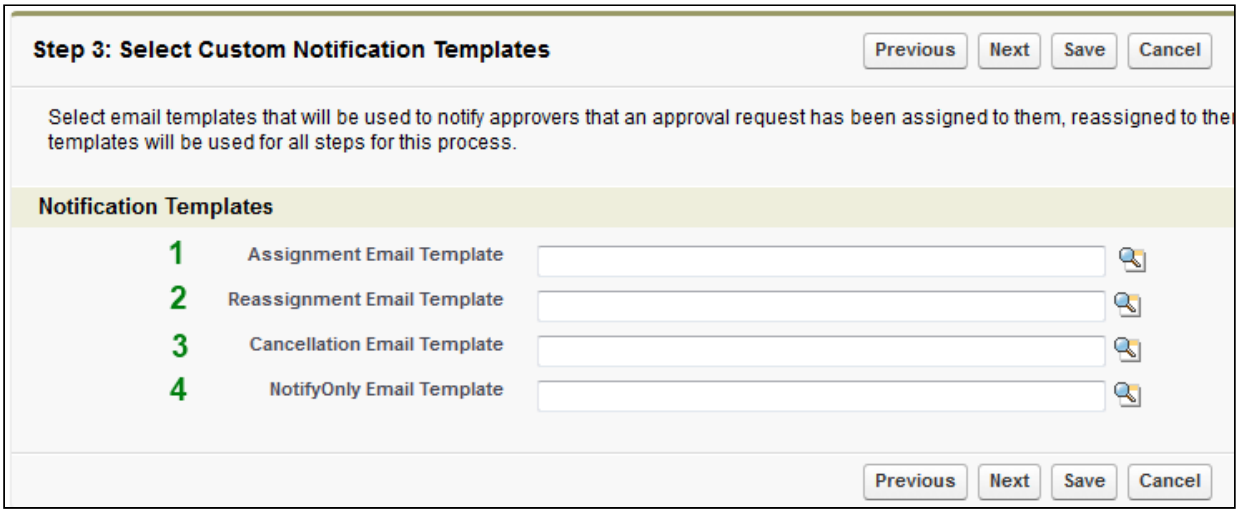
APTS_ApprovalConfig

The following example shows *APTS_ApprovalConfig*, with template overrides for each template type and for some of the objects which can be associated with Intelligent Approvals: Opportunities, Agreements Term Exceptions, Quote/Proposals, and Quote Line Items.

The numbers indicate how the templates in *APTS_ApprovalConfig* map to the fields used to select the templates when creating your intelligent approval process.

Name	APTS_ApprovalConfig
Value	XML
Code	<ApprovalConfig>

```
<OpportunitySpec>
<EmailTemplates>
1 <AssignmentEmailTemplate>C1d_OpptyApprovalAssignEmail</AssignmentEmailTemplate>
2 <ReassignmentEmailTemplate>C1d_OpptyApprovalReassignEmail</ReassignmentEmailTemplate>
4 <NotifyOnlyEmailTemplate>C1d_OpptyApprovalNotifyOnlyEmail</NotifyOnlyEmailTemplate>
3 <CancellationEmailTemplate>C1d_OpptyApprovalCancelEmail</CancellationEmailTemplate>
</EmailTemplates>
</OpportunitySpec>

<AgreementSpec>
<EmailTemplates>
<AssignmentEmailTemplate>C1d_AgrAppr_NotificationAssignmentEmail</AssignmentEmailTemplate>
<ReassignmentEmailTemplate>C1d_AgrApprReassignEmail</ReassignmentEmailTemplate>
<NotifyOnlyEmailTemplate>C1d_AgrApprNotifyOnlyEmail</NotifyOnlyEmailTemplate>
<CancellationEmailTemplate>C1d_AgrApprCancelEmail</CancellationEmailTemplate>
</EmailTemplates>
</AgreementSpec>

<QuoteSpec>
<EmailTemplates>
<AssignmentEmailTemplate>C1d_QuoteApprovalAssignEmail</AssignmentEmailTemplate>
<ReassignmentEmailTemplate>C1d_QuoteApprovalReassignEmail</ReassignmentEmailTemplate>
<NotifyOnlyEmailTemplate>C1d_QuoteApprovalNotifyOnlyEmail</NotifyOnlyEmailTemplate>
<CancellationEmailTemplate>C1d_QuoteApprovalCancelEmail</CancellationEmailTemplate>
</EmailTemplates>
</QuoteSpec>

<QuoteLineItemSpec>
<EmailTemplates>
<AssignmentEmailTemplate>C1d_QuoteLItemApprovalAssignEmail</AssignmentEmailTemplate>
<ReassignmentEmailTemplate>C1d_QuoteLItemApprovalReassignEmail</ReassignmentEmailTemplate>
<NotifyOnlyEmailTemplate>C1d_QuoteLItemApprovalNotifyOnlyEmail</NotifyOnlyEmailTemplate>
<CancellationEmailTemplate>C1d_QuoteLItemApprovalCancelEmail</CancellationEmailTemplate>
</EmailTemplates>
</QuoteLineItemSpec>

</ApprovalConfig>
```



Configurable Email Sender Name

You can select who you want to be displayed in the *From* field, when Approvals emails are sent.

You can select to have emails indicate they were sent by an *Approver*, *Requester*, or *Submitter* in the *From* field.

Approver	This value comes from the last person in the chain of approval requests to approve the request. For the initial approval request email - before anyone has actually approved it - the submitter is used.
Requester	This value comes from the Agreement record that is submitted for approval. For example: Requestor Ben Easter Requester is only valid for agreements and agreement term exceptions.
Submitter	This value comes from the user who clicks *Submit* in the *Approvals* related list for the object record that is being approved.

If the setting is not configured, it defaults to the name of the requester for agreements, otherwise it defaults to the name of the submitter.

Email Approval System Properties Settings

To enable approval requests to be decided via email, an email service and Approval System Properties settings for emails must be configured.

Typically an approval request will be sent to the designated assignee, from where they can access Salesforce to see more details and approve the request; however, by ensuring these settings are configured correctly, an assignee can indicate their decision and progress the approval by simply replying to the request email.

To configure the email service

1. Go to **Build > Develop > Email Services** and click **New Email Service**.

2. For **Email Service Name**, enter `ApprovalReplyService` and for Apex Class, **select ApprovalEmailHandler**.
3. Select **Active** and for **Route Error Emails to This Email Address** enter the email address of a user with admin level access. The remaining settings are not required, but may be used if necessary.
4. Click **Save** and **New Email Address** and then enter `ApprovalReplyService` for email address.
5. Select **Active**, select a **Context User** and click **Save**. The email address is automatically generated for your Salesforce org, apart from the text before@which you entered in Step 4. For more details, see Salesforce Help.

This email service can now be used to ensure approval request emails are handled as expected.

To set the email system properties

You must have configured your email service, so you can enter the address is Step 2 below.

1. Go to **Build > Develop > Custom Settings** and click **Manage** for **Approvals System Properties**.
2. Click **Edit** and ensure the **Email Approval Service Address** is the one that was automatically generated by the system when you configured the email service.
3. Select the checkbox for **Enable Email Approval Response** and click **Save**.
4. To enable approval email response ensure that you follow the steps at Enable Email Approval Response.

> ⓘ Queue email address cannot be used for notifications. If Queue has "Send Email to Members" box un-checked - email of current user will be used for notifications (not queue email address). This is due to Salesforce limitations which we cannot bypass.

When someone replies to an approval request via email, the address their email is sent to is the one set in the Email Approval Service Address field and selecting Enable Email Approval Response means they can enter *Approve*, *Approved*, *Yes*, *Reject*, *Rejected*, or *No* to decide on the approval without logging into Salesforce.

Bounce back emails

If you make a spelling mistake when replying to an approval request email the system can send you a 'bounce back' email to inform you an error has occurred. As long as the previous

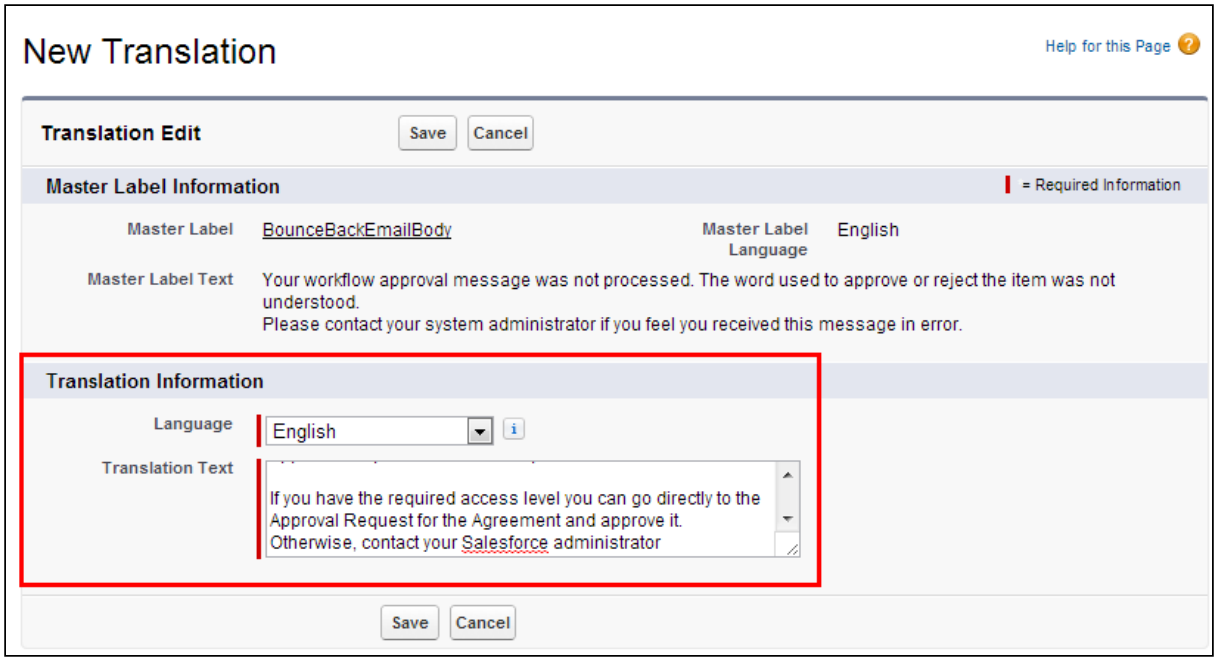
tasks have been completed for the email service and email system properties, this will occur automatically.

When you reply to an approval request email by typing in one of the supported words (*Approve*, *Approved*, *Yes*, *Reject*, *Rejected*, *No*), the approval request status will be updated accordingly in the system. If you make a typo when you do this, no update can be made. This feature enables you to be informed of the mistake and make the necessary correction to ensure the approval process progresses as intended.The email will be sent to the approver who made the mistake only.If you do not want to use the default text for the email, you can use Salesforce Translation Workbench functionality to customize the text. A custom label is used to provide the content for the body of the email and you can use translation text to override this.

To customize the bounce back email text

You must have enabled the Translation Workbench for your org.

1. Go to **Build > Create > Custom Labels** and select **BounceBackEmailBody**.
2. Click **New Local Translations / Overrides** to display the New Translation page.
3. Select the **Language** and enter the text you want included in the email and click **Save**.



The next time a bounce back email is sent, the body of the email will use the content in the **Translation Text** field for the corresponding language.

You can edit the text as required and also add different versions for multiple languages.

Setting up Email Templates and Alerts

Default email templates provide the ability to use Intelligent Approvals out of the box and begin testing approval process workflows quickly; however, if you want to have bespoke emails for your organization or you want to attach documents to approval request emails, you must use custom email templates. Email alerts are triggered by workflow rules and can make use of the same email templates.

Typically the easiest way to create your own custom template is to clone one of the email templates provided in the unmanaged Conga samples package and then edit it. You cannot clone the standard email templates included in the *Conga Approvals Management* package.

The *Conga Approvals Management* package provides email template examples for Agreements, Opportunities, and Term Exception Approvals. *Conga Quote/Proposal Approvals Management* provides examples for Quotes/Proposals and Quote Line Items.

After that has been done, you can then override the default email templates for all of your approval processes using Approvals Custom Config Custom Settings, APTS_ApprovalConfig Admin Settings, or on an ad-hoc basis for specific approval processes.

Note

The selection of the email template to use is determined as follows:

- Email templates selected in an **Approval Process** override templates in **Approvals Custom Config.**
- Templates in **Approvals Custom Config** override templates in **APTS_AppprovalConfig.**

If you do not configure email templates in any of the three fields, you cannot send approval emails.

To customize email templates

You must have installed the *Conga Approvals Management* and *Conga Quote/Proposal Approvals Management* packages.

Go to **Administration Setup > Communication Templates > Email Templates.**

1. From the **Folder** list, select **Conga Email Templates** and then select the email template you want to use as the basis for your customized template.

2. Click **Clone**, enter the required template details and click **Save**. **Template Unique Name** is the value that must be used when you override the default email templates.
3. Click **Edit Template** to display the Visualforce editor and make the required changes to the template. You should be familiar with Visualforce pages and apex code to complete this step.

> **Attention**
>
> When you edit the text and code in a template you should be sure that the *Approval Request ID* and *Approval User ID* fields are still included in the template. These are required for the approvals engine to route emails correctly and retain the ability to decide on an email by responding to it. When you customize an email template ensure that you retain the Context.recipientSO.Id email markup instead of using Context.requestSO.Apttus_Approval__Assigned_To_Id__c. Not retaining the original markup causes an email notification issue where some of the recipients do not receive an email notification.
> Ensure that you set the Target Object Id in the template. For more information, refer to TargetObjectId.

4. Click **Save** to return to the main template view page.
5. In the **Standard Attachments** section, select **Attach File** and select a file you want attached to each request.
6. Click **Send Test** and **Verify Merge Fields** to display the Preview Template window.
7. Select a user and an approval request for an object that matches the object the template will be used for, select any email address and click **OK**. The email can be used to verify that all the fields have the expected values and that any attachments are included.
8. To enable approval email response ensure that you follow the steps at Enable Email Approval Response.

Templates that are **Available For Use** can now be used for any subsequent approval requests.

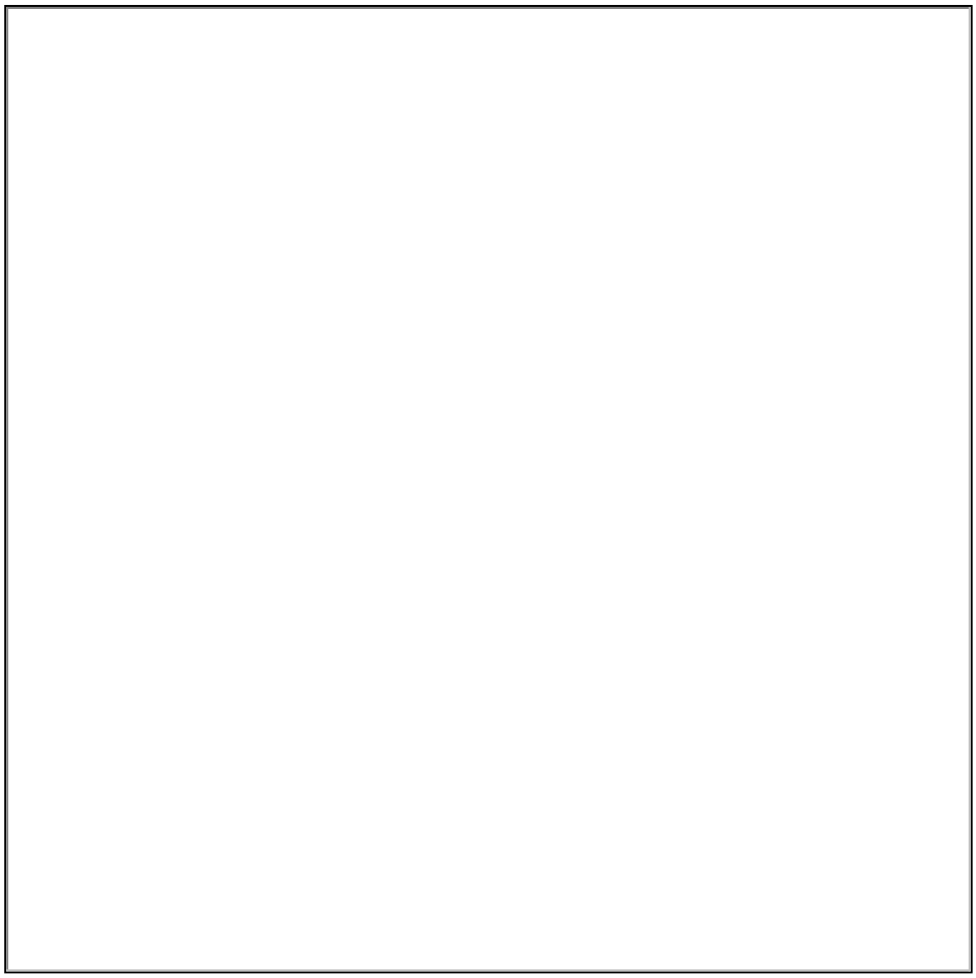
You can make the template a default template or to override the default templates on an ad-hoc basis, you can now use this template when you create an approval process.

To override default email templates with Custom settings

You must have customized templates you can reference.

1. Go to **Build > Develop > Custom Settings** and click **Approvals Custom Config.**

2. From the **Custom Setting Definition Detail** section, click **Manage** to display the current objects associated with Intelligent Approvals.
3. Click **Edit** beside the object that you want to edit email templates for.
4. Enter the **Name** of the customized email template in the appropriate field.



5. Click **Save** when complete.

The templates referenced in the template fields will be used as the global Intelligent Approval default email templates for that object. When the Process Definition Notification Templates value fields are left blank, while creating an approval process, these templates will be used.

To override default email templates with Admin settings

You must have customized templates you can reference.

1. Select the **Admin** tab - or - select ☐ and select **Admin**.

2. Click **Go** to display all of the admin settings.
3. If **APTS_ApprovalConfig** is listed, select it, then click **Edit** and proceed to Step 5.
4. If it is not listed click **New Admin**, for **Name** enter `APTS_ApprovalConfig` and for **Value** enter `XML`.
5. In the **Code** field enter the references to the custom email templates for each of the objects you are using with Intelligent Approvals.
   See, Approval Email Notification Templates, if you need help with formatting of the XML code.

The templates referenced in the Code field will be used as the global Intelligent Approval default email templates. When the Process Definition **Notification Templates** value fields are left blank, these templates will be used.

To edit email alerts

Go to **Build > Create > Workflow & Approvals > Email Alerts**

1. Click **Edit** beside **Approval Request Comments Entered**. Since it is a part of the managed package, you cannot change the description or unique name.
2. Click ☐ and select the required email template.
3. Select the **Recipient Type** and the related recipients and click **Save**.

Subsequent email alerts that are sent when approval request comments are entered, will use the new email template. Any other email alerts can be edited in the same manner.

Settings for Email Reply

Create email service: Go to **Setup > Develop > Email Services.**

Accept Email From should contain the domains which we plan to use: For Example: Accept Email From: apttus.com, yahoo.com, gmail.com

In the Email Addresses section, click **New Email Address** and **Accept Email** From: apttus.com, yahoo.com, gmail.com and click **Save**. New email address should be created in the format "approval_email_service@t8l0k46ikux0mfg3lk083tqdn2qetmjc7cosf2m5v92hp70s1.11-fk7eai.cs18.apex.sandbox.salesforce.com".

Custom Label setting:

Edit the **Custom Label ApttusEMailServiceAddress**. Ensure that the value of the custom label matches that of the Email Address specified in Email Services.

In the **Local Translations / Overrides** section, click **New Local Translations / Overrides**.

Copy the string (Email Address) from Email Service- Email Addresses section and paste it in the Translation Text box.

Click **Save**.

Approvals System Properties:

Email Approval Service Address should contain same string from Email Service- Email Addresses page.

Email Templates:

Should contain the reply to as below:

replyTo="{!$Label.Apttus_Approval__ApttusEMailServiceAddress}"

> Whenever a reassigned or a delegate approver approves or rejects a request, the next approver does not recieve an email Notification. If the previous approver has a Yahoo email address, the email is not routed to the next approver. Follow the DMARC guidelines and the steps listed at Google Support and Yahoo Support to ensure email notifications are generated and sent.

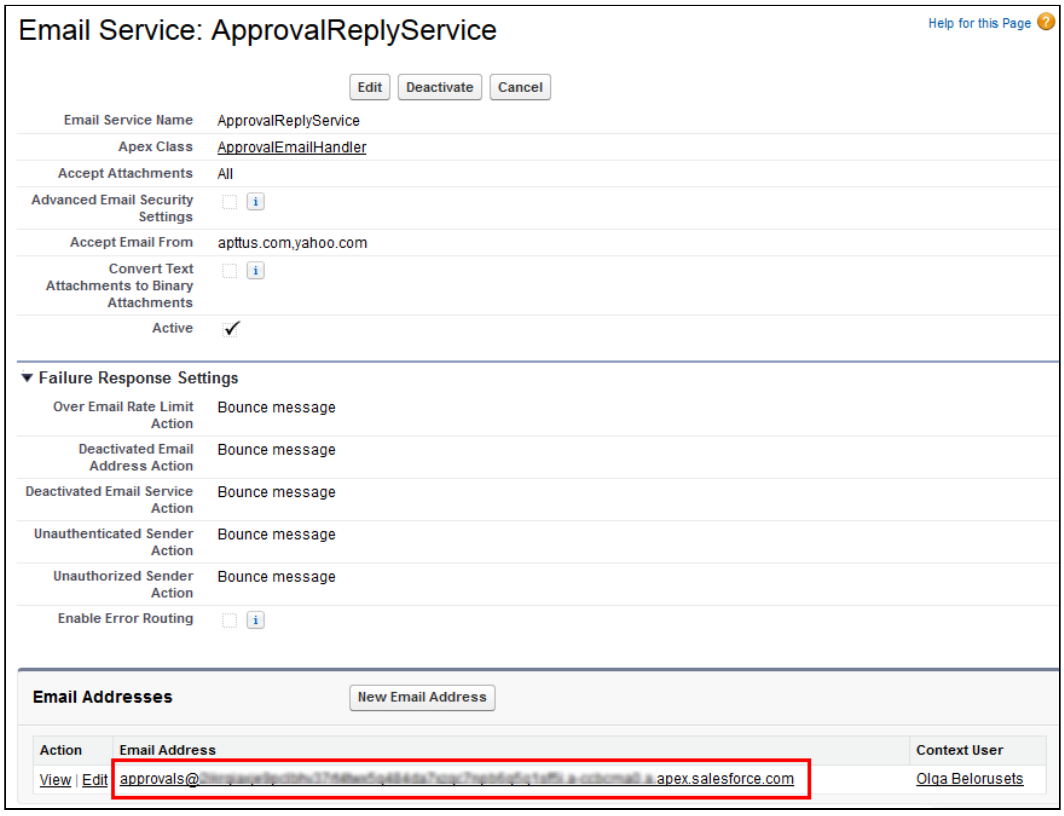
Configuring Custom Labels for Email Templates

There are five new email templates used by Consolidated Approvals and to ensure that emails are sent to the correct address, you must configure a custom label used by the emails. Each email template references the same label, so you only need to configure the one custom label as it affects each template.

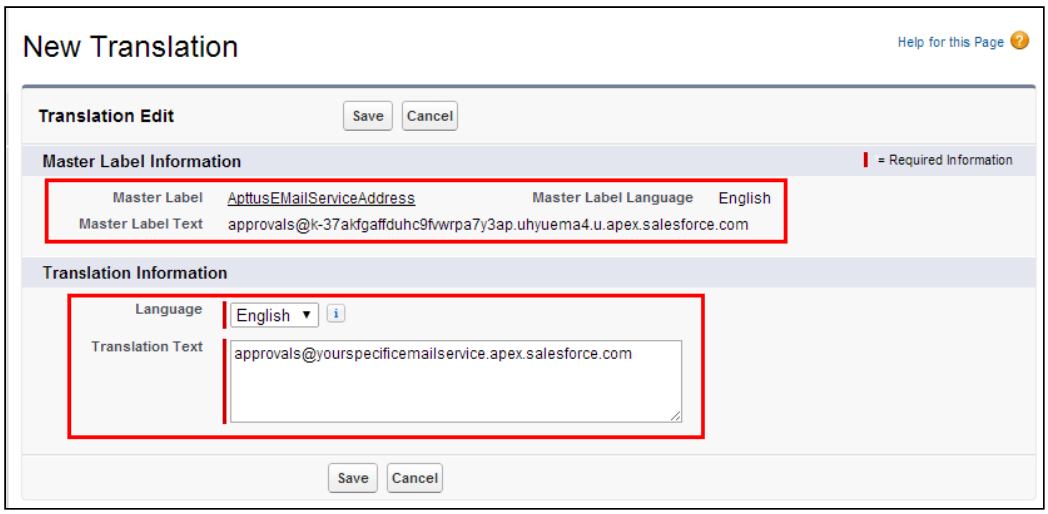
```
<messaging:emailTemplate subject="{!$Label.ApprovalRequestsFor} '{!relatedTo.Object_Name__c}'
                         recipientType="User"
                         relatedToType="Approval_Request__c"
                         replyTo="{!$Label.ApttusEMailServiceAddress}"
                         language="{!recipient.LanguageLocaleKey}">
```

You must have already configured your email service.

1. From **Setup**, go to **Develop > Email Services** and click the **Email Service Name** *ApprovalReplyService.*
2. Copy the system generated **Email Address**, which is required in the upcoming steps.



3. From **Setup**, go to **Create > Custom Labels** and select **ApttusEmailServiceAddress**.
4. Click **New Local Translations / Overrides** to display the Translation Edit page.
5. The **Master Label Information** section displays the text that is used by default, which points to a dummy email service that you cannot use with your org.



6. In the **Translation Information section**, select **English** from the **Language** list and in the **Translation Text** field, paste in the email address copied in Step 2.
7. Click **Save**.

The label is now updated with the email address for your org.

When consolidated approval emails are next sent, they will be routed to the correct users.

Setting the Config Custom Classes

1. Go to **Setup > Build > Develop > Custom Settings** and click **Manage** for **Config Custom Classes**.
2. Click **Edit** beside **Custom Classes**.
3. For **Cart Approval Callback Class**, ensure the value is`Apttus_CQApprov.CartApprovalCallback`. This is required to have the *Approval Required* icon and *Submit for Approval* button in the Shopping Cart.
4. For **Deal Optimizer Callback Class**, ensure the value is`Apttus_CPQOpti.DealMaximizerCallback`. This is required to have the *Approval Required* icon and *Submit for Approval* button available when using Deal Maximizer.

Configuring Status Picklist values

For a shopping cart/product configuration to go through the approval process correctly, you need to have approval values available from the *Status* field.

To add values to the Status picklist

1. From **Setup**, go to **Create > Objects** and click **Product Configuration**.
2. From **Custom Fields & Relationships**, click **Status**.
3. From **Picklist Values**, click **New** and add *Approval Required* and *Pending Approval*.
4. Click **Save**.

These values will now be available from the Status picklist for a Product Configuration object record, which also means the status can appear in the Status field at the top left of the Shopping Cart.

Adding Values to Context Object Picklists

1. Go to **Setup > Build > Create > Objects** and click **Approval Process**.
2. From **Custom Fields & Relationships,** click **Object Label**.
3. From **Picklist Values**, click **New**, add `Product Configuration` and click **Save**.
4. From **Custom Fields & Relationships,** click **Object Type**.

5. From **Picklist Values**, click **New**, add `Apttus_Config2_ProductConfiguration__c` and click **Save.**

Setting Approval Flag to True

If you define an approval process for a specific object and want the process to be initiated on click of a button, navigate to **Create > Objects > <<Object Name>>Quote > <<Field Name>>Configure Products(Field) > Edit** the formula field and add **useAdvancedApproval=true**. For example, IF (LEN(Apttus_QPConfig__PriceListId__c) > 0 , HYPERLINK("/apex/Apttus_QPConfig__ProposalConfiguration?id=" &Id&"&**useAdvancedApproval=true**&useDealOptimizer=true", IMAGE("/resource/ Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"), NULL).

Creating Search Filters Approvals

The two Filter Type options (Child Filter and Entry Criteria) serve two different purposes.

Filter Type	Purpose
Entry Criteria	This is required for approvals to be intiated from the Shopping Cart. To improve performance, the system will check if there are any Entry Criteria search filters that match the attributes in the Shopping Cart and set the approval status to Approval Required. You can have the entry criteria at the line item to set the approval status to Approval Required or Approval Triggered.
Child Filter	This is required to have a header level approval step entry criteria that makes use of the line item fields for the products in the Shopping Cart. After you create a child filter, you then need to create step entry criteria using child filters that mirrors the expression used in the child filter.

Configuring Search Filters

Search filters of type Entry Criteria are used to check whether the contents of a shopping cart require approval before it can be finalized.

If you want approvals to be triggered at the *Shopping Cart Header* level, then you need to configure search filters based on the *Product Configuration* context object. If you want approvals to be triggered at a specific Cart Line Item level, then you would configure

search filters for the Line Item context object. Additionally if you want consolidated approvals for the whole Cart/Product Configuration, then you would create both.

Configuring Search Filters - this adds Product Configuration to the Search Filter object picklist so you can create search filters that use these objects for their entry criteria

To create Search Filters (Approvals) - the Search Filters are used to determine if the products in the Shopping Cart require approval.

To configure Search Filters (Approvals)

1. Go to **Setup > Build > Create > Objects** and select **Search Filter (Approvals)**.
2. From the **Custom Fields & Relationships** section, select **Business Object**.
3. From **Picklist Values**, click **New** and enter the API names for the cart and the line items:`Apttus_Config2__ProductConfiguration__c`for Header level approvals and`Apttus_Config2__LineItem__c`for Line Item level approvals.
4. Click **Save**. The values are added to the **Picklist Values** list.
5. Click **Back to Search Filter (Approvals)** and from the **Custom Fields & Relationships** section, select **Filter Type**.
6. From **Picklist Values**, click **New** and enter`Child Filter`and`Entry Criteria`.
7. Click **Save**.

The values are added to the picklists, which are required to create search filters.

You can create *Search Filters (Approvals)* for controlling which carts are checked to see if they need approval.

The search filter field size accommodates 16000 characters.

To create Search Filters (Approvals)

1. Click ✚ and select **Search Filter (Approvals)**.
2. Click **New Search Filter (Approvals)** and select either:
   - **Product Configuration** - select this if you want the filter to be at the Product Configuration/Shopping Cart (Header) context object level.
   - **Line Item** - select this if you want the filter to be at the Line Item context object level, which are the individual line items within the cart.
3. Click **Next** to display the *Configure Filter* page.
4. From **Filter Type**, select one of the following:
   - **Child Filter** - This enables you to have step entry criteria within your approval process that references the child record fields of the main context object. Typically this enables you to use Line Item fields with your *Product*

Configuration approval process. The *Line Item Child Filters* you create here, enable you to have them available for approval step entry criteria.

- **Entry Criteria** - is used to indicate that the search filter will be used for the *Approval Required* check, which checks to see whether the quote or quote line item values require the shopping cart to undergo approval. You are effectively having an additional level of entry criteria for approvals here, before the system searches for matching approval processes. You must have at least one of these for approvals to be available from the shopping cart.

5. Enter a required **Filter Name** and **Sequence**. The sequence is used to determine the order in which the search filters will check if approvals are required.
6. Enter the filter criteria, which is used to determine if the quote requires approval.
7. Click **Save**.

The Search Filter is now active and if it is a Filter Type of *Entry Criteria* it will be used by the system to determine if a Shopping Cart requires approval.

You must create an approval process that can be used with the Shopping Cart.

Permission Sets for Accessing Approval Center

You must provide the following permission sets to your users so they can access the Approval Center:

- Conga Grid Users
- Conga Grid Visualforce pages

Prerequisites

Ensure that Conga Grid package is installed in your Salesforce Org.

To add users to Conga Grid Permission Set

1. Navigate to **Setup.**
2. From the setup menu on left, go to **Administrators > Users → Permission Sets.**
3. Select **Conga Grid Users** permission sets from the list.
4. Click **Manage Assignments > click Add Assignments.**
5. Click the checkbox to the left of the Users that need access and click **Assign.**
6. Click **Done.**

To add users to Conga Grid Visualforce Pages Permission Set

1. Navigate to **Setup.**
2. From the setup menu on left, go to **Administrators > Users → Permission Sets.**
3. Click **Conga Grid Visualforce Pages** from the list.
4. Click **Manage Assignments > click Add Assignments.**
5. Select the respective user(s) from the list and click
6. Click **Done.**

Access Control and Permissions

Approvals Administrators need to provide permissions to users to perform approval actions using Conga Approvals. This topic contains the following information:

Action Permissions

As an Approvals administrator, you can assign security permissions for users so that the application actions are accessible to only certain user profiles.

There are two categories of actions related to approvals:

- Submitter Actions
- Approver Actions

Submitter Actions

Submitter actions are performed by the user who is trying to submit an object for approval. These actions include:

- Preview Approvals
- Submit for Approvals
- Cancel / Recall Approvals

- View Approvals History / Comments

Following users can perform Submitter Actions:

- Internal user (user who has direct access to Salesforce Org)
- Community User (External users who are authorized to take approval actions)

Approver Actions

Approver actions are performed by the user who is part of the approval chain for a given object. These actions include:

- Approve
- Reject
- Reassign
- Add Ad hoc approver
- Take Ownership of Queue / Role requests
- Add comment
- Email Approval
- Email Rejection

Following users can perform approval actions:

- Internal user (user who has direct access to Salesforce Org)
- Community User (External users who are authorized to take approval actions)

Other Approval Users are:

- Backup User
- Delegate User
- System Admin
- Backup Admin

Users need the following permissions to perform any approval actions:

- Read/Write access to the context objects on which the approval is required (primary object and child objects). Read/Write access is needed as the system updates certain fields on these objects, such as Approval Status, Approval Preview Status, Status, when user performs various approvals actions using the user credentials.
- Read/Write access to Approval Request, Approval Request history, and Approval Process instance as the system creates and updates records in these objects when the user performs various approvals actions using the user credentials.
- Read access on other approval objects, such as Approval Process, Approval Rule, Approval Rule Entry, Approval Rule Assignee, Approval Rule Dimension, and Approval Matrix, to access the approval process and rule details.

Approvals Object	OWD (Org Wide Default) Settings	Sharing Rules	Approve/ Reject Actions	Reassign Approver	Add Ad-hoc Approver
Approval Request Object	Public Read Only	None	Only My steps. Not for anyone else	• Can reassign My Approval request • Cannot reassign others approval Request	Can not add an Ad hoc approver at any step
	Private	None	Only My steps. Not for anyone else	• Can reassign My Approval request • Cannot reassign others approval Request	Can not add an Ad hoc approver at any step.
	Public Read/ Write	None	Only My steps. Not for anyone else	• Can reassign Approval request at any step	Can add Ad hoc approver for any step

Approvals Object	OWD (Org Wide Default) Settings	Sharing Rules	Approve/ Reject Actions	Reassign Approver	Add Ad-hoc Approver
(Preferred Setting)	Public Read/ Write	Public Read Only	Only My steps. Not for anyone else	• Can reassign My Approval request • Cannot reassign others approval Request	Can add Ad hoc approver for any step

The following table describes the permissions required by different user profiles:

Approval Object	Description	Submitter	Approver	Backup Admin	Backup Approver	Delegate Approver	Approval Admin
Approval Request	Tracks the approvals needed from a given user on a given object	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write
Approval Request History	Stores history of the approval requests	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write

Approval Object	Description	Submitter	Approver	Backup Admin	Backup Approver	Delegate Approver	Approval Admin
Approval Process	Stores the definition of the approvals needed for a given object. This include • General Information • Step and step dependencies • Initial and final submission actions • Email notification templates	• Read	• Read	• Read	• Read	• Read	• Read • Write
Approval Process Instance	Represents an instance of a single end-to-end approval process that is generated during approval execution	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write
Approval Rule	Similar to approval process. Also represents a sub-process / child process linked to a given step in the primary approval process.	• Read	• Read	• Read	• Read	• Read	• Read • Write

Approval Object	Description	Submitter	Approver	Backup Admin	Backup Approver	Delegate Approver	Approval Admin
Approval Rule Assignee	Similar to approval process.	• Read	• Read	• Read	• Read	• Read	• Read • Write
Approval Rule Dimension	Represents an approval rule dimension.	• Read	• Read	• Read	• Read	• Read	• Read • Write
Approval Rule Entry	Represents a single entry in an approval rule.	• Read	• Read	• Read • Write	• Read	• Read	• Read • Write
Approval Matrix	Specifies user approval levels and authorized discount percentages.	• Read	• Read	• Read	• Read	• Read	• Read • Write
Context Object	Represents the approval context object.	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write	• Read • Write

Visibility of Approvals Data

Approval data, such as Approval Requests and Approval Request history, is available to any user who has Read access to the approval context objects as well as to the objects used for Approvals.

Access Control

As an Approvals administrator, you can control who can access approval actions. There are two categories of actions related to approvals:

- Submitter Actions
- Approver Actions

Approval Actions Control

The access to Approval action is currently controlled on the standard **Approvals** page, **My Approval** page, and from the **Approval Center.**

- Only authorized users can perform Approve, Reject, Reassign, Take Ownership of Queue /Role approval actions:
    - The assignee of the request
    - Delegate / Backup for the assignee
    - Backup Admin User
    - System Admin User
- Only users who receive assignment/reassignment notifications can perform email approval actions.
- Only the following authorized users can add Ad hoc approver:
    - The assignee of the request
    - Delegate / Backup for the assignee
    - Backup Admin User
    - System Admin User
    - In addition, the current assignee on any request on the My Approvals page can add Ad-hoc approver for any request that is Assigned or Not Submitted status

Submitter Actions Access Control

The access to Submitter actions such as Preview Approvals, Submit for Approval, Cancel/ Recall approval, is available to any user who has Read/Write access to the approval context objects as well as to the objects used for Approvals (Approval Request, Approval Process Instance, Approval Request History)

You can also implement additional controls at the project level by writing triggers on the respective actions as follows:

Actions	Trigger to control further access
Preview Approvals Needed	Write a trigger on Insert of Approval Request record
Submit for Approval	Write a trigger on the creation of Approval Process Instance
Cancel / Recall Approvals	Write a trigger on status update of Approval Process Instance

Recommended Permissions for Approval Administrators

Select one of the following recommended permission topics for Administrators:

- Application Permissions for Administrators
- Apex Class Permissions for Administrators
- Custom Setting Permissions for Administrators
- Field Permissions for Administrators
- Object Permissions for Administrators
- Page Access Permissions for Administrators
- Record Type Visibility Permissions for Administrators
- Tab Visibility Permissions for Administrators

Application Permissions for Administrators

The following table displays the permissions required by administrators for Approval Applications:

Application	Visibility
Conga Approvals	TRUE
Conga Approvals Setup	TRUE
Apttus__ApttusContractManagement	TRUE
Apttus__ApttusContractManagement_Lightning	TRUE

For information on setting application permissions, refer to this Salesforce article.

Apex Class Permissions for Administrators

The following table displays the permissions required by administrators for Apex Classes:

Apex Class	Enabled
AP	TRUE
AbstractAgreementTermExActionController	TRUE
AbstractApprovalEmailController	TRUE
AbstractApprovalEngineProcessTerm	TRUE
AbstractApprovalsTest	TRUE
AbstractQueryCallback	TRUE
AbstractSObjectApprovals2	TRUE
AdhocApprovalGroup	TRUE
AdhocApprovalProcess	TRUE
AdhocApprovalProcessFactory	TRUE
AdhocApprovalTestSupport	TRUE
AdhocApproverFactory	TRUE
AdhocProcessRuntimeControllerTest	TRUE
AgreementApprovalEmailController	TRUE
AgreementApprovalSummaryController	TRUE
AgreementLIApprovalEmailControllerTest	TRUE

Apex Class	Enabled
AgreementLItemApprovalSummaryController	TRUE
AgreementTEApprovalEmailControllerTest	TRUE
AgreementTermExAddTest	TRUE
AgreementTermExApprovalSummaryController	TRUE
AgreementTermExApprovalsController	TRUE
AgreementTermExCancelController	TRUE
AgreementTermExEditController	TRUE
AgreementTermExHistoryController	TRUE
AgreementTermExPreviewController	TRUE
AgreementTermExSubmitController	TRUE
AgreementTermExWrapper	TRUE
ApprovalActionTest2	TRUE
ApprovalChildProcess	TRUE
ApprovalConfigTest	TRUE
ApprovalContext	TRUE
ApprovalContextCancelTest	TRUE
ApprovalContextPreviewController	TRUE
ApprovalContextSubmitController	TRUE
ApprovalCustomConfig	TRUE
ApprovalData	TRUE

Apex Class	Enabled
ApprovalDataTransferAgent	TRUE
ApprovalEmailHandler	TRUE
ApprovalEngine2	TRUE
ApprovalEngineConsolidationSupport	TRUE
ApprovalEnginePrescanSupport	TRUE
ApprovalEngineProcessAgreementTermEx	TRUE
ApprovalEngineProcessContextQueueable	TRUE
ApprovalEngineProcessRequest	TRUE
ApprovalEngineStepSupport	TRUE
ApprovalEngineSupport	TRUE
ApprovalEngineTestATESupport	TRUE
ApprovalEngineTestAgmtSupport	TRUE
ApprovalEngineTestOpptySupport	TRUE
ApprovalHistoryControllerTest	TRUE
ApprovalMatrixAssigneeSupport	TRUE
ApprovalMatrixGenerateTest	TRUE
ApprovalMatrixResolver	TRUE
ApprovalMatrixViewController	TRUE
ApprovalPolicySupport	TRUE

Apex Class	Enabled
ApprovalProcessDefnController	TRUE
ApprovalProcessDefnListController	TRUE
ApprovalProcessFactory	TRUE
ApprovalProcessFinalActionController	TRUE
ApprovalProcessInitialActionController	TRUE
ApprovalProcessInstance	TRUE
ApprovalProcessQryHelper	TRUE
ApprovalProcessResolverTest	TRUE
ApprovalProcessStepControllerTest	TRUE
ApprovalProcessTest2	TRUE
ApprovalRequestAttachmentController	TRUE
ApprovalRequestEscalateController	TRUE
ApprovalRequestQryHelper	TRUE
ApprovalRequestReassignController2	TRUE
ApprovalRequestSupport	TRUE
ApprovalRequestsListTest	TRUE
ApprovalRequiredCheckTest	TRUE
ApprovalRuleAssignee	TRUE
ApprovalRuleController	TRUE
ApprovalRuleCriteriaEditController	TRUE

Apex Class	Enabled
ApprovalRuleDimension	TRUE
ApprovalRuleDimensionControllerTest	TRUE
ApprovalRuleDimensionSupport	TRUE
ApprovalRuleEntryController	TRUE
ApprovalRuleEntryFactory	TRUE
ApprovalRuleSupport	TRUE
ApprovalStatusUpdateBatchJob	TRUE
ApprovalSubprocess	TRUE
ApprovalSummaryLaunchController	TRUE
ApprovalSystemException	TRUE
ApprovalUserSupportTest	TRUE
ApprovalsController	TRUE
ApprovalsWebService	TRUE
ApprovalsWebServiceTest	TRUE
AssigneeCacheSupport	TRUE
AsyncActionSupport	TRUE
AsyncJobSupport	TRUE
AttachmentWrapper	TRUE
BackupAdminSupportTest	TRUE

Apex Class	Enabled
BackupApproverActionTest	TRUE
BackupApproverSupport	TRUE
BackupUserSearchController	TRUE
BatchPreviewApprovals	TRUE
BulkActionRequest	TRUE
ChildSOSupport	TRUE
ClauseApprovalsWebServiceTest	TRUE
ContextApprovalEmailControllerTest	TRUE
CriteriaCacheTest	TRUE
CurrencySupportTest	TRUE
CustomAssigneeInfoTest	TRUE
CustomClass	TRUE
CustomSOSupport	TRUE
DashboardSupportTest	TRUE
ESAPISupport	TRUE
EmailSupport	TRUE
EmailTemplateFactory	TRUE
EmailTemplateTest	TRUE
EscalationReminderEmailController	TRUE
EscalationSupport	TRUE

Apex Class	Enabled
ExprSupport	TRUE
FieldMetadata	TRUE
FileSupportTest	TRUE
FormulaFieldController	TRUE
FormulaFieldFactory	TRUE
IApprovalEngine	TRUE
InstanceUrlController	TRUE
ManagedObject	TRUE
ManagedObjectTest	TRUE
MyApprovalsController	TRUE
MyApprovalsEmailController	TRUE
MyApprovalsLaunchController	TRUE
NextLevelApproverController	TRUE
NullObject	TRUE
ObjectValue	TRUE
OneOrMoreItemsFailedException	TRUE
OpportunityApprovalSummaryTest	TRUE
OpportunityPartnersListTest	TRUE
OpportunityProductsListTest	TRUE

Apex Class	Enabled
OpptyApprovalEmailControllerTest	TRUE
PostInstallBatchApprovalRequests	TRUE
PostInstallBatchCopyProcessId	TRUE
PostInstallScript	TRUE
PreviewSubmitApprovalsController	TRUE
Property	TRUE
QuerySpecFactory	TRUE
ReassignUserSearchTest	TRUE
ReminderInstanceSupport	TRUE
ReminderJobTest	TRUE
ReminderTest	TRUE
RuntimeContext	TRUE
SOQLSupport	TRUE
SObjectApprovalContextParam	TRUE
SObjectApprovals2SubmitController	TRUE
SObjectApprovals2Test	TRUE
SObjectApprovalsController2	TRUE
SObjectApprovalsSupport	TRUE
SObjectConstants	TRUE
SObjectMetadata	TRUE

Apex Class	Enabled
SObjectSupport	TRUE
SearchFilter	TRUE
SearchFilterControllerTest	TRUE
SearchFilterDetailControllerTest	TRUE
SubmissionComments	TRUE
SystemUtil	TRUE
TaskConstants	TRUE
TaskSupport	TRUE
TermExApprovalCloneController	TRUE
TermExApprovalController	TRUE
TermExApprovalFactory	TRUE
TermExSupport	TRUE
TransferInflightRequests	TRUE
UIAbstractSearchController	TRUE
UIAssigneeController	TRUE
UIDataTableController	TRUE
UIDependsOnControllerTest	TRUE
UIDisplayFieldNamesControllerTest	TRUE
UIPageConfig	TRUE

Apex Class	Enabled
UIPageControllerBase	TRUE
UIPageSupport	TRUE
UIReapprovalFilterControllerTest	TRUE
UISearchEmailTemplateController	TRUE
UISearchFilterControllerTest	TRUE
UISearchRoleController	TRUE
UISearchUserController	TRUE

For information on setting apex class permissions, refer to this Salesforce article.

Custom Setting Permissions for Administrators

The following table displays the permissions required by administrators for Approval Custom Settings:

Custom Setting Name	Enabled
ApprovalsCustomClasses__c	TRUE
ApprovalsCustomConfig__c	TRUE
ApprovalsSystemProperties__c	TRUE

For information on setting custom setting permissions, refer to this Salesforce article.

Field Permissions for Administrators

The following table displays the permissions required by administrators for Approval Fields:

Field	Editable	Readable
AdhocApprovalGroup__c.DependsOn__c	TRUE	TRUE

Field	Editable	Readable
AdhocApprovalGroup__c.GroupName__c	TRUE	TRUE
AdhocApprovalGroup__c.GroupSequence__c	TRUE	TRUE
AdhocApprovalProcess__c.AttachmentIds__c	TRUE	TRUE
AdhocApprovalProcess__c.BusinessObjectId__c	TRUE	TRUE
AdhocApprovalProcess__c.BusinessObjectType__c	TRUE	TRUE
AdhocApprovalProcess__c.DisplayFields__c	TRUE	TRUE
AdhocApprovalProcess__c.DisplayHeaderFields__c	TRUE	TRUE
AdhocApprovalProcess__c.ProcessComments__c	TRUE	TRUE
AdhocApprover__c.ApproverSequence__c	TRUE	TRUE
AdhocApprover__c.AssigneeId__c	TRUE	TRUE
AdhocApprover__c.AssigneeType__c	TRUE	TRUE
AdhocApprover__c.AssigneeValue__c	TRUE	TRUE
AdhocApprover__c.AutoReapprovalEnabled__c	TRUE	TRUE
AdhocApprover__c.DependsOn__c	TRUE	TRUE
AdhocApprover__c.IsReviewer__c	TRUE	TRUE
AdhocApprover__c.SendEmail__c	TRUE	TRUE
ApprovalJob__c.ApexChildJobId__c	TRUE	TRUE
ApprovalJob__c.ApexParentJobId__c	TRUE	TRUE

Field	Editable	Readable
ApprovalJob__c.ApprovalRequestId__c	TRUE	TRUE
ApprovalJob__c.Status__c	TRUE	TRUE
ApprovalProcessInstance__c.AssignmentEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.AttachmentIds__c	TRUE	TRUE
ApprovalProcessInstance__c.BusinessObjectLink__c	FALSE	TRUE
ApprovalProcessInstance__c.BusinessObjectType__c	TRUE	TRUE
ApprovalProcessInstance__c.CancellationEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.ConsolidationVersionNumber__c	TRUE	TRUE
ApprovalProcessInstance__c.Data__c	TRUE	TRUE
ApprovalProcessInstance__c.EndTime__c	TRUE	TRUE
ApprovalProcessInstance__c.EscalationEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.InstanceNumber__c	FALSE	TRUE
ApprovalProcessInstance__c.NotifyOnlyEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.OriginalParentApprovalProcessId__c	TRUE	TRUE
ApprovalProcessInstance__c.PrevProcessInstanceId__c	TRUE	TRUE

Field	Editable	Readable
ApprovalProcessInstance__c.ReassignmentEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.ReminderEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.StartTime__c	TRUE	TRUE
ApprovalProcessInstance__c.Status__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeAutoEscalate__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeDescription__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeEscalateToId__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeEscalateToName__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeEscalateToType__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeExpectedDaysToComplete__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeExpectedHoursToComplete__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeId__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeLabel__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeType__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeValue__c	TRUE	TRUE

Field	Editable	Readable
ApprovalRuleAssignee__c.AutoComplete__c	TRUE	TRUE
ApprovalRuleAssignee__c.DependsOn__c	TRUE	TRUE
ApprovalRuleAssignee__c.NotifyOnly__c	TRUE	TRUE
ApprovalRuleAssignee__c.SendEmail__c	TRUE	TRUE
ApprovalRuleAssignee__c.SendReminders__c	TRUE	TRUE
ApprovalRuleAssignee__c.Sequence__c	TRUE	TRUE
ApprovalRuleAssignee__c.SkipUnresolvedAssignee__c	TRUE	TRUE
ApprovalRuleDimension__c.BusinessObject__c	TRUE	TRUE
ApprovalRuleDimension__c.Description__c	TRUE	TRUE
ApprovalRuleDimension__c.DimensionType__c	TRUE	TRUE
ApprovalRuleEntry__c.AutoReapprovalCriteria__c	TRUE	TRUE
ApprovalRuleEntry__c.AutoReapprove__c	TRUE	TRUE
ApprovalRuleEntry__c.ConditionFieldNames__c	TRUE	TRUE
ApprovalRuleEntry__c.Condition__c	TRUE	TRUE
ApprovalRuleEntry__c.CriteriaFieldNames__c	TRUE	TRUE
ApprovalRuleEntry__c.DependsOn__c	TRUE	TRUE
ApprovalRuleEntry__c.Description__c	TRUE	TRUE
ApprovalRuleEntry__c.Dimension1Value__c	TRUE	TRUE
ApprovalRuleEntry__c.Dimension2Value__c	TRUE	TRUE

Field	Editable	Readable
ApprovalRuleEntry__c.Dimension3Value__c	TRUE	TRUE
ApprovalRuleEntry__c.Dimension4Value__c	TRUE	TRUE
ApprovalRuleEntry__c.Dimension5Value__c	TRUE	TRUE
ApprovalRuleEntry__c.Dimension6Value__c	TRUE	TRUE
ApprovalRuleEntry__c.EntryLabel__c	TRUE	TRUE
ApprovalRuleEntry__c.IncludePrevious__c	TRUE	TRUE
ApprovalRuleEntry__c.SelectValueType__c	TRUE	TRUE
ApprovalRuleEntry__c.SelectValue__c	TRUE	TRUE
ApprovalRule__c.Active__c	TRUE	TRUE
ApprovalRule__c.ApprovalCount__c	TRUE	TRUE
ApprovalRule__c.ApprovalPercent__c	TRUE	TRUE
ApprovalRule__c.ApprovalPolicy__c	TRUE	TRUE
ApprovalRule__c.BusinessObject__c	TRUE	TRUE
ApprovalRule__c.ContinuePolicyApprovalOnAReject__c	TRUE	TRUE
ApprovalRule__c.CriteriaFieldNames__c	TRUE	TRUE
ApprovalRule__c.Criteria__c	TRUE	TRUE
ApprovalRule__c.Description__c	TRUE	TRUE
ApprovalRule__c.Dimension1Id__c	TRUE	TRUE
ApprovalRule__c.Dimension1ValueType__c	TRUE	TRUE

Field	Editable	Readable
ApprovalRule__c.Dimension2Id__c	TRUE	TRUE
ApprovalRule__c.Dimension2ValueType__c	TRUE	TRUE
ApprovalRule__c.Dimension3Id__c	TRUE	TRUE
ApprovalRule__c.Dimension3ValueType__c	TRUE	TRUE
ApprovalRule__c.Dimension4Id__c	TRUE	TRUE
ApprovalRule__c.Dimension4ValueType__c	TRUE	TRUE
ApprovalRule__c.Dimension5Id__c	TRUE	TRUE
ApprovalRule__c.Dimension5ValueType__c	TRUE	TRUE
ApprovalRule__c.Dimension6Id__c	TRUE	TRUE
ApprovalRule__c.Dimension6ValueType__c	TRUE	TRUE
ApprovalRule__c.EffectiveDate__c	TRUE	TRUE
ApprovalRule__c.ExpirationDate__c	TRUE	TRUE
ApprovalRule__c.MatchRule__c	TRUE	TRUE
ApprovalRule__c.RuleType__c	TRUE	TRUE
ApprovalRule__c.StopProcessingMoreRules__c	TRUE	TRUE
Approval_Matrix__c.Approval_Level__c	TRUE	TRUE
Approval_Matrix__c.Approver_1_Authorized_Discount__c	TRUE	TRUE
Approval_Matrix__c.Approver_1_Level__c	TRUE	TRUE
Approval_Matrix__c.Approver_1__c	TRUE	TRUE

Field	Editable	Readable
Approval_Matrix__c.Approver_2_Authorized_Discount__c	TRUE	TRUE
Approval_Matrix__c.Approver_2_Level__c	TRUE	TRUE
Approval_Matrix__c.Approver_2__c	TRUE	TRUE
Approval_Matrix__c.Approver_3_Authorized_Discount__c	TRUE	TRUE
Approval_Matrix__c.Approver_3_Level__c	TRUE	TRUE
Approval_Matrix__c.Approver_3__c	TRUE	TRUE
Approval_Matrix__c.Approver_4_Authorized_Discount__c	TRUE	TRUE
Approval_Matrix__c.Approver_4_Level__c	TRUE	TRUE
Approval_Matrix__c.Approver_4__c	TRUE	TRUE
Approval_Matrix__c.Approver_5_Authorized_Discount__c	TRUE	TRUE
Approval_Matrix__c.Approver_5_Level__c	TRUE	TRUE
Approval_Matrix__c.Approver_5__c	TRUE	TRUE
Approval_Matrix__c.Approver_6_Authorized_Discount__c	TRUE	TRUE
Approval_Matrix__c.Approver_6_Level__c	TRUE	TRUE
Approval_Matrix__c.Approver_6__c	TRUE	TRUE
Approval_Matrix__c.Description__c	TRUE	TRUE
Approval_Matrix__c.Error_Log__c	TRUE	TRUE

Field	Editable	Readable
Approval_Matrix__c.Is_Active__c	TRUE	TRUE
Approval_Matrix__c.Is_Valid__c	TRUE	TRUE
Approval_Matrix__c.Matrix_Owner__c	TRUE	TRUE
Approval_Matrix__c.Next_Level_Approver__c	TRUE	TRUE
Approval_Matrix__c.User__c	TRUE	TRUE
Approval_Process__c.Active__c	TRUE	TRUE
Approval_Process__c.AdhocAllowApprovedRequestsToBeDeleted__c	TRUE	TRUE
Approval_Process__c.AdhocAutoReapprovalStatusLabel__c	TRUE	TRUE
Approval_Process__c.AdhocStepAutoReapprove__c	TRUE	TRUE
Approval_Process__c.ApprovalCommentsMandatory__c	TRUE	TRUE
Approval_Process__c.ApprovalSummaryPage__c	TRUE	TRUE
Approval_Process__c.Assignment_Email_Template__c	TRUE	TRUE
Approval_Process__c.BackupAdminUser__c	TRUE	TRUE
Approval_Process__c.CancelPendingRequestsOnAReject__c	TRUE	TRUE
Approval_Process__c.Cancellation_Email_Template__c	TRUE	TRUE
Approval_Process__c.ChildObjectType__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.ConsolidateApprovals__c	TRUE	TRUE
Approval_Process__c.ConsolidateNotifications__c	TRUE	TRUE
Approval_Process__c.ContextType__c	TRUE	TRUE
Approval_Process__c.ContinuePendingApprovalsOnAReject__c	TRUE	TRUE
Approval_Process__c.CriteriaFieldNames__c	TRUE	TRUE
Approval_Process__c.CustomMyApprovalsPage__c	TRUE	TRUE
Approval_Process__c.CustomPreviewPage__c	TRUE	TRUE
Approval_Process__c.DependsOn__c	TRUE	TRUE
Approval_Process__c.Description__c	TRUE	TRUE
Approval_Process__c.EntryCriteriaFilterLogic__c	TRUE	TRUE
Approval_Process__c.Entry_Criteria_Active__c	TRUE	TRUE
Approval_Process__c.Entry_Criteria_BoolOper__c	TRUE	TRUE
Approval_Process__c.Entry_Criteria_Comparison_Type__c	TRUE	TRUE
Approval_Process__c.Entry_Criteria_Field_Value__c	TRUE	TRUE
Approval_Process__c.Entry_Criteria_Field__c	TRUE	TRUE
Approval_Process__c.Entry_Criteria__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.Escalation_Email_Template__c	TRUE	TRUE
Approval_Process__c.Final_Action_Field_Name_Source__c	TRUE	TRUE
Approval_Process__c.Final_Action_Field_Name__c	TRUE	TRUE
Approval_Process__c.Final_Action_Field_Update_Type__c	TRUE	TRUE
Approval_Process__c.Final_Action_Field_Value__c	TRUE	TRUE
Approval_Process__c.Final_Action_Type__c	TRUE	TRUE
Approval_Process__c.Final_Approval_Action__c	TRUE	TRUE
Approval_Process__c.Group_Action__c	TRUE	TRUE
Approval_Process__c.Initial_Submission_Action_Type__c	TRUE	TRUE
Approval_Process__c.Initial_Submission_Action__c	TRUE	TRUE
Approval_Process__c.Initial_Submission_Field_Name_Source__c	TRUE	TRUE
Approval_Process__c.Initial_Submission_Field_Name__c	TRUE	TRUE
Approval_Process__c.Initial_Submission_Field_Value__c	TRUE	TRUE
Approval_Process__c.IsAdhoc__c	TRUE	TRUE
Approval_Process__c.NotifyOnly_Email_Template__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.Object_Label__c	TRUE	TRUE
Approval_Process__c.Object_Name__c	TRUE	TRUE
Approval_Process__c.Object_Type__c	TRUE	TRUE
Approval_Process__c.OriginalInstanceProcessId__c	TRUE	TRUE
Approval_Process__c.ProcessAutoEscalate__c	TRUE	TRUE
Approval_Process__c.ProcessEscalateToId__c	TRUE	TRUE
Approval_Process__c.ProcessEscalateToName__c	TRUE	TRUE
Approval_Process__c.ProcessEscalateToType__c	TRUE	TRUE
Approval_Process__c.ProcessExpectedDaysToComplete__c	TRUE	TRUE
Approval_Process__c.ProcessExpectedHoursToComplete__c	TRUE	TRUE
Approval_Process__c.ProcessExpectedMinutesToComplete__c	TRUE	TRUE
Approval_Process__c.Process_Name__c	TRUE	TRUE
Approval_Process__c.Reassignment_Email_Template__c	TRUE	TRUE
Approval_Process__c.RejectionCommentsMandatory__c	TRUE	TRUE
Approval_Process__c.Reminder_Email_Template__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.Send_Email__c	TRUE	TRUE
Approval_Process__c.Sequence__c	TRUE	TRUE
Approval_Process__c.StepAutoEscalate__c	TRUE	TRUE
Approval_Process__c.StepAutoReapprovalCriteria__c	TRUE	TRUE
Approval_Process__c.StepAutoReapprove__c	TRUE	TRUE
Approval_Process__c.StepCarbonCopyAssigneeIds__c	TRUE	TRUE
Approval_Process__c.StepCarbonCopyUser__c	TRUE	TRUE
Approval_Process__c.StepDisplayFieldNameHeader__c	TRUE	TRUE
Approval_Process__c.StepDisplayFieldNames__c	TRUE	TRUE
Approval_Process__c.StepEscalateToId__c	TRUE	TRUE
Approval_Process__c.StepEscalateToName__c	TRUE	TRUE
Approval_Process__c.StepEscalateToType__c	TRUE	TRUE
Approval_Process__c.StepExpectedDaysToComplete__c	TRUE	TRUE
Approval_Process__c.StepExpectedHoursToComplete__c	TRUE	TRUE
Approval_Process__c.StepExpectedMinutesToComplete__c	TRUE	TRUE
Approval_Process__c.StepFilterConditionDescription__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.StepFilterLogic__c	TRUE	TRUE
Approval_Process__c.StepKeyFieldNames__c	TRUE	TRUE
Approval_Process__c.StepLabel__c	TRUE	TRUE
Approval_Process__c.StepRemindersActive__c	TRUE	TRUE
Approval_Process__c.StepSequence__c	TRUE	TRUE
Approval_Process__c.StepSkipUnresolvedAssignee__c	TRUE	TRUE
Approval_Process__c.StepSubmissionComment1Enabled__c	TRUE	TRUE
Approval_Process__c.StepSubmissionComment2Enabled__c	TRUE	TRUE
Approval_Process__c.StepSubmissionComment3Enabled__c	TRUE	TRUE
Approval_Process__c.StepSubmissionCommentsEnabled__c	TRUE	TRUE
Approval_Process__c.StepType__c	TRUE	TRUE
Approval_Process__c.Step_Active__c	TRUE	TRUE
Approval_Process__c.Step_Approver_Type__c	TRUE	TRUE
Approval_Process__c.Step_Assignee_Description__c	TRUE	TRUE
Approval_Process__c.Step_Assignee_Id__c	TRUE	TRUE
Approval_Process__c.Step_Assignee_Type__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.Step_Assignee__c	TRUE	TRUE
Approval_Process__c.Step_Auto_Complete__c	TRUE	TRUE
Approval_Process__c.Step_Filter_BoolOper__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Comments__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Comparison_Type__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Field_Number__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Field_Object_Source__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Field_Object__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Field_Value_Object_Source__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Field_Value_Object__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Field_Value__c	TRUE	TRUE
Approval_Process__c.Step_Filter_Field__c	TRUE	TRUE
Approval_Process__c.Step_Filter__c	TRUE	TRUE
Approval_Process__c.Step_Group_Action_Field_Name_Source__c	TRUE	TRUE
Approval_Process__c.Step_Group_Action_Field_Name__c	TRUE	TRUE
Approval_Process__c.Step_Group_Action_Field_Update_Type__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.Step_Group_Action_Field_Value__c	TRUE	TRUE
Approval_Process__c.Step_Group_Action_Type__c	TRUE	TRUE
Approval_Process__c.Step_Group_Action__c	TRUE	TRUE
Approval_Process__c.Step_Group_Active__c	TRUE	TRUE
Approval_Process__c.Step_Group_Name__c	TRUE	TRUE
Approval_Process__c.Step_Group_Rejection_Action__c	TRUE	TRUE
Approval_Process__c.Step_Group_Seq_Number__c	TRUE	TRUE
Approval_Process__c.Step_Group__c	TRUE	TRUE
Approval_Process__c.Step_Hierarchy_Driver__c	TRUE	TRUE
Approval_Process__c.Step_Name__c	TRUE	TRUE
Approval_Process__c.Step_Notify_Only__c	TRUE	TRUE
Approval_Process__c.Step_Seq_Number__c	TRUE	TRUE
Approval_Process__c.Step__c	TRUE	TRUE
Approval_Process__c.SubmissionComment1Label__c	TRUE	TRUE
Approval_Process__c.SubmissionComment2Label__c	TRUE	TRUE
Approval_Process__c.SubmissionComment3Label__c	TRUE	TRUE

Field	Editable	Readable
Approval_Process__c.SubmissionCommentsEnabledCount__c	TRUE	TRUE
Approval_Process__c.SubmissionCommentsEnabled__c	TRUE	TRUE
Approval_Process__c.SubmissionCommentsMandatory__c	TRUE	TRUE
Approval_Process__c.SubmissionCommentsPerStep__c	TRUE	TRUE
Approval_Process__c.SubmissionCommentsType__c	TRUE	TRUE
Approval_Request_History__c.Action_Approve_Id__c	TRUE	TRUE
Approval_Request_History__c.Action_Reassign_Id__c	TRUE	TRUE
Approval_Request_History__c.Action__c	FALSE	TRUE
Approval_Request_History__c.Active__c	TRUE	TRUE
Approval_Request_History__c.ActualApproverName__c	FALSE	TRUE
Approval_Request_History__c.Actual_Approver__c	TRUE	TRUE
Approval_Request_History__c.AgreementLineItemId__c	TRUE	TRUE
Approval_Request_History__c.ApprovalCount__c	TRUE	TRUE
Approval_Request_History__c.ApprovalFromEmail__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request_History__c.ApprovalPercent__c	TRUE	TRUE
Approval_Request_History__c.ApprovalPolicy__c	TRUE	TRUE
Approval_Request_History__c.Approval_Process__c	TRUE	TRUE
Approval_Request_History__c.Approval_Status__c	TRUE	TRUE
Approval_Request_History__c.Approver_Comments__c	TRUE	TRUE
Approval_Request_History__c.Assigned_To_Id__c	TRUE	TRUE
Approval_Request_History__c.Assigned_To_Link__c	FALSE	TRUE
Approval_Request_History__c.Assigned_To_Name__c	TRUE	TRUE
Approval_Request_History__c.Assigned_To_Type__c	TRUE	TRUE
Approval_Request_History__c.Assigned_To__c	TRUE	TRUE
Approval_Request_History__c.AutoEscalate__c	TRUE	TRUE
Approval_Request_History__c.AutoReapprove__c	TRUE	TRUE
Approval_Request_History__c.Auto_Complete__c	TRUE	TRUE
Approval_Request_History__c.Backup_From_User__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request_History__c.CanEscalate__c	TRUE	TRUE
Approval_Request_History__c.CarbonCopyAssigneeIds__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectId__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectLink__c	FALSE	TRUE
Approval_Request_History__c.ChildObjectName__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectRefId__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectType__c	TRUE	TRUE
Approval_Request_History__c.ContinuePolicyApprovalOnARReject__c	TRUE	TRUE
Approval_Request_History__c.CriteriaFieldNames__c	TRUE	TRUE
Approval_Request_History__c.DateApproved__c	TRUE	TRUE
Approval_Request_History__c.DateAssigned__c	TRUE	TRUE
Approval_Request_History__c.DateCancelled__c	TRUE	TRUE
Approval_Request_History__c.DateEscalated__c	TRUE	TRUE
Approval_Request_History__c.DateReassigned__c	TRUE	TRUE
Approval_Request_History__c.DateRejected__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request_History__c.Date__c	TRUE	TRUE
Approval_Request_History__c.DelegateApproverIds__c	TRUE	TRUE
Approval_Request_History__c.DelegateApproverNames__c	TRUE	TRUE
Approval_Request_History__c.DelegateApprover__c	TRUE	TRUE
Approval_Request_History__c.DependsOn__c	TRUE	TRUE
Approval_Request_History__c.EscalateToChain__c	TRUE	TRUE
Approval_Request_History__c.EscalateToId__c	TRUE	TRUE
Approval_Request_History__c.EscalateToName__c	TRUE	TRUE
Approval_Request_History__c.EscalateToType__c	TRUE	TRUE
Approval_Request_History__c.EscalatedToHighestLevel__c	TRUE	TRUE
Approval_Request_History__c.ExpectedCompletionDate__c	TRUE	TRUE
Approval_Request_History__c.ExpectedDaysToComplete__c	TRUE	TRUE
Approval_Request_History__c.ExpectedHoursToComplete__c	TRUE	TRUE
Approval_Request_History__c.ExpectedMinutesToComplete__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request_History__c.Group_Unique_Id__c	TRUE	TRUE
Approval_Request_History__c.Group__c	TRUE	TRUE
Approval_Request_History__c.HasAttachments__c	TRUE	TRUE
Approval_Request_History__c.HasDelegateApprover__c	TRUE	TRUE
Approval_Request_History__c.InEscalation__c	TRUE	TRUE
Approval_Request_History__c.Initial_Submitter__c	TRUE	TRUE
Approval_Request_History__c.Internal_Comments__c	TRUE	TRUE
Approval_Request_History__c.IsAdhocProcess__c	TRUE	TRUE
Approval_Request_History__c.IsAdhoc__c	TRUE	TRUE
Approval_Request_History__c.IsAutoReapprovalEnabled__c	TRUE	TRUE
Approval_Request_History__c.IsSubprocess__c	TRUE	TRUE
Approval_Request_History__c.Notify_Only__c	TRUE	TRUE
Approval_Request_History__c.ObjectRefId__c	TRUE	TRUE
Approval_Request_History__c.Object_Id_Link__c	FALSE	TRUE
Approval_Request_History__c.Object_Id__c	TRUE	TRUE
Approval_Request_History__c.Object_Name__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request_History__c.Object_Type__c	TRUE	TRUE
Approval_Request_History__c.ParentRequestId__c	TRUE	TRUE
Approval_Request_History__c.Parent_Agreement__c	TRUE	TRUE
Approval_Request_History__c.PrevAssignedToId__c	TRUE	TRUE
Approval_Request_History__c.PrevAssignedToName__c	TRUE	TRUE
Approval_Request_History__c.PrevAssignedToType__c	TRUE	TRUE
Approval_Request_History__c.ProcessInstanceId__c	TRUE	TRUE
Approval_Request_History__c.Rejection_Action__c	TRUE	TRUE
Approval_Request_History__c.Related_Agreement_Owner__c	TRUE	TRUE
Approval_Request_History__c.Related_Agreement_Requestor__c	TRUE	TRUE
Approval_Request_History__c.Related_Agreement_Term_Exception__c	TRUE	TRUE
Approval_Request_History__c.Related_Agreement__c	TRUE	TRUE
Approval_Request_History__c.Related_Opportunity_Owner__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request_History__c.Related_Opportunity__c	TRUE	TRUE
Approval_Request_History__c.RequestType__c	TRUE	TRUE
Approval_Request_History__c.Request_Comments__c	TRUE	TRUE
Approval_Request_History__c.SendReminders__c	TRUE	TRUE
Approval_Request_History__c.Send_Email__c	TRUE	TRUE
Approval_Request_History__c.SequenceString__c	FALSE	TRUE
Approval_Request_History__c.Sequence__c	TRUE	TRUE
Approval_Request_History__c.Status_Link__c	FALSE	TRUE
Approval_Request_History__c.StepLabel__c	TRUE	TRUE
Approval_Request_History__c.StepNameLink__c	FALSE	TRUE
Approval_Request_History__c.StepSequenceString__c	FALSE	TRUE
Approval_Request_History__c.StepSequence__c	TRUE	TRUE
Approval_Request_History__c.Step_Group_Seq_Number__c	TRUE	TRUE
Approval_Request_History__c.Step_Name__c	TRUE	TRUE
Approval_Request_History__c.Step__c	TRUE	TRUE
Approval_Request_History__c.SubmissionComment1__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request_History__c.SubmissionComment2__c	TRUE	TRUE
Approval_Request_History__c.SubmissionComment3__c	TRUE	TRUE
Approval_Request_History__c.SubprocessDependsOn__c	TRUE	TRUE
Approval_Request_History__c.SubprocessName__c	TRUE	TRUE
Approval_Request_History__c.SubprocessSequence__c	TRUE	TRUE
Approval_Request_History__c.SubstepDependsOn__c	TRUE	TRUE
Approval_Request_History__c.SubstepName__c	TRUE	TRUE
Approval_Request_History__c.SubstepSequence__c	TRUE	TRUE
Approval_Request_History__c.Workflow_Trigger_Added_Comments__c	TRUE	TRUE
Approval_Request__c.Action_Approve_Id__c	TRUE	TRUE
Approval_Request__c.Action_Prefix__c	FALSE	TRUE
Approval_Request__c.Action_Reassign_Id__c	TRUE	TRUE
Approval_Request__c.Action__c	FALSE	TRUE
Approval_Request__c.Active__c	TRUE	TRUE
Approval_Request__c.ActualApproverName__c	FALSE	TRUE

Field	Editable	Readable
Approval_Request__c.Actual_Approver__c	TRUE	TRUE
Approval_Request__c.AgreementLineItemId__c	TRUE	TRUE
Approval_Request__c.ApprovalCount__c	TRUE	TRUE
Approval_Request__c.ApprovalFromEmail__c	TRUE	TRUE
Approval_Request__c.ApprovalPercent__c	TRUE	TRUE
Approval_Request__c.ApprovalPolicy__c	TRUE	TRUE
Approval_Request__c.Approval_Process__c	TRUE	TRUE
Approval_Request__c.Approval_Status__c	TRUE	TRUE
Approval_Request__c.Approver_Comments__c	TRUE	TRUE
Approval_Request__c.Assigned_To_Id__c	TRUE	TRUE
Approval_Request__c.Assigned_To_Link__c	FALSE	TRUE
Approval_Request__c.Assigned_To_Name__c	TRUE	TRUE
Approval_Request__c.Assigned_To_Type__c	TRUE	TRUE
Approval_Request__c.Assigned_To__c	TRUE	TRUE
Approval_Request__c.AutoEscalate__c	TRUE	TRUE
Approval_Request__c.AutoReapprove__c	TRUE	TRUE
Approval_Request__c.Auto_Complete__c	TRUE	TRUE
Approval_Request__c.Backup_From_User__c	TRUE	TRUE
Approval_Request__c.CanEscalate__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request__c.CarbonCopyAssigneeIds__c	TRUE	TRUE
Approval_Request__c.ChildObjectId__c	TRUE	TRUE
Approval_Request__c.ChildObjectLink__c	FALSE	TRUE
Approval_Request__c.ChildObjectName__c	TRUE	TRUE
Approval_Request__c.ChildObjectRefId__c	TRUE	TRUE
Approval_Request__c.ChildObjectType__c	TRUE	TRUE
Approval_Request__c.ContinuePolicyApprovalOnAReject__c	TRUE	TRUE
Approval_Request__c.CriteriaFieldNames__c	TRUE	TRUE
Approval_Request__c.DateApproved__c	TRUE	TRUE
Approval_Request__c.DateAssigned__c	TRUE	TRUE
Approval_Request__c.DateCancelled__c	TRUE	TRUE
Approval_Request__c.DateEscalated__c	TRUE	TRUE
Approval_Request__c.DateReassigned__c	TRUE	TRUE
Approval_Request__c.DateRejected__c	TRUE	TRUE
Approval_Request__c.Date__c	TRUE	TRUE
Approval_Request__c.DelegateApproverIds__c	TRUE	TRUE
Approval_Request__c.DelegateApproverNames__c	TRUE	TRUE
Approval_Request__c.DelegateApprover__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request__c.DependsOn__c	TRUE	TRUE
Approval_Request__c.EscalateToChain__c	TRUE	TRUE
Approval_Request__c.EscalateToId__c	TRUE	TRUE
Approval_Request__c.EscalateToName__c	TRUE	TRUE
Approval_Request__c.EscalateToType__c	TRUE	TRUE
Approval_Request__c.EscalatedToHighestLevel__c	TRUE	TRUE
Approval_Request__c.ExpectedCompletionDate__c	TRUE	TRUE
Approval_Request__c.ExpectedDaysToComplete__c	TRUE	TRUE
Approval_Request__c.ExpectedHoursToComplete__c	TRUE	TRUE
Approval_Request__c.ExpectedMinutesToComplete__c	TRUE	TRUE
Approval_Request__c.Group_Unique_Id__c	TRUE	TRUE
Approval_Request__c.Group__c	TRUE	TRUE
Approval_Request__c.HasAttachments__c	TRUE	TRUE
Approval_Request__c.HasDelegateApprover__c	TRUE	TRUE
Approval_Request__c.InEscalation__c	TRUE	TRUE
Approval_Request__c.Initial_Submitter__c	TRUE	TRUE
Approval_Request__c.Internal_Comments__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request__c.IsAdhocProcess__c	TRUE	TRUE
Approval_Request__c.IsAdhoc__c	TRUE	TRUE
Approval_Request__c.IsAutoReapprovalEnabled__c	TRUE	TRUE
Approval_Request__c.IsSubprocess__c	TRUE	TRUE
Approval_Request__c.Notify_Only__c	TRUE	TRUE
Approval_Request__c.ObjectRefId__c	TRUE	TRUE
Approval_Request__c.Object_Id_Link__c	FALSE	TRUE
Approval_Request__c.Object_Id__c	TRUE	TRUE
Approval_Request__c.Object_Name__c	TRUE	TRUE
Approval_Request__c.Object_Type__c	TRUE	TRUE
Approval_Request__c.ParentRequestId__c	TRUE	TRUE
Approval_Request__c.Parent_Agreement__c	TRUE	TRUE
Approval_Request__c.PrevAssignedToId__c	TRUE	TRUE
Approval_Request__c.PrevAssignedToName__c	TRUE	TRUE
Approval_Request__c.PrevAssignedToType__c	TRUE	TRUE
Approval_Request__c.ProcessInstanceId__c	TRUE	TRUE
Approval_Request__c.Rejection_Action__c	TRUE	TRUE
Approval_Request__c.Related_Agreement_Owner__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request__c.Related_Agreement_Requestor__c	TRUE	TRUE
Approval_Request__c.Related_Agreement_Term_Exception__c	TRUE	TRUE
Approval_Request__c.Related_Agreement__c	TRUE	TRUE
Approval_Request__c.Related_Opportunity_Owner__c	TRUE	TRUE
Approval_Request__c.Related_Opportunity__c	TRUE	TRUE
Approval_Request__c.RequestType__c	TRUE	TRUE
Approval_Request__c.Request_Comments__c	TRUE	TRUE
Approval_Request__c.SendReminders__c	TRUE	TRUE
Approval_Request__c.Send_Email__c	TRUE	TRUE
Approval_Request__c.SequenceString__c	FALSE	TRUE
Approval_Request__c.Sequence__c	TRUE	TRUE
Approval_Request__c.Status_Link__c	FALSE	TRUE
Approval_Request__c.StepLabel__c	TRUE	TRUE
Approval_Request__c.StepNameLink__c	FALSE	TRUE
Approval_Request__c.StepSequenceString__c	FALSE	TRUE
Approval_Request__c.StepSequence__c	TRUE	TRUE
Approval_Request__c.Step_Group_Seq_Number__c	TRUE	TRUE
Approval_Request__c.Step_Name__c	TRUE	TRUE

Field	Editable	Readable
Approval_Request__c.Step__c	TRUE	TRUE
Approval_Request__c.SubmissionComment1__c	TRUE	TRUE
Approval_Request__c.SubmissionComment2__c	TRUE	TRUE
Approval_Request__c.SubmissionComment3__c	TRUE	TRUE
Approval_Request__c.SubprocessDependsOn__c	TRUE	TRUE
Approval_Request__c.SubprocessName__c	TRUE	TRUE
Approval_Request__c.SubprocessSequence__c	TRUE	TRUE
Approval_Request__c.SubstepDependsOn__c	TRUE	TRUE
Approval_Request__c.SubstepName__c	TRUE	TRUE
Approval_Request__c.SubstepSequence__c	TRUE	TRUE
Approval_Request__c.Workflow_Trigger_Added_Comments__c	TRUE	TRUE
Apttus__APTS_Admin__c.Apttus__Code__c	TRUE	TRUE
Apttus__APTS_Admin__c.Apttus__Value__c	TRUE	TRUE
Apttus__APTS_Agreement__c.Approval_Status__c	TRUE	TRUE
Apttus__APTS_Agreement__c.LineItem_Approval_Status__c	TRUE	TRUE
Apttus__APTS_Agreement__c.Term_Exception_Approval_Status__c	TRUE	TRUE
Apttus__Term_Exception__c.Apttus__Active__c	TRUE	TRUE

Field	Editable	Readable
Apttus__Term_Exception__c.Apttus__Agreement_Types__c	TRUE	TRUE
Apttus__Term_Exception__c.Apttus__Approval_Required__c	TRUE	TRUE
Apttus__Term_Exception__c.Apttus__Description__c	TRUE	TRUE
Apttus__Term_Exception__c.Apttus__Exception_Type__c	TRUE	TRUE
Apttus__Term_Exception__c.Apttus__ReferenceId__c	TRUE	TRUE
AsyncActionInfo__c.AsyncActionJobId__c	TRUE	TRUE
AsyncActionInfo__c.EndTime__c	TRUE	TRUE
AsyncActionInfo__c.ErrorMsgInfo__c	TRUE	TRUE
AsyncActionInfo__c.ErrorStackInfo__c	TRUE	TRUE
AsyncActionInfo__c.ObjectName__c	TRUE	TRUE
AsyncActionInfo__c.StartTime__c	TRUE	TRUE
AsyncActionInfo__c.Status__c	TRUE	TRUE
AsyncActionInfo__c.SubmitTime__c	TRUE	TRUE
AsyncActionInfo__c.Submitter__c	TRUE	TRUE
Backup_Approver__c.Backup_Action__c	FALSE	TRUE
Backup_Approver__c.Backup_User__c	TRUE	TRUE
Backup_Approver__c.Cancellation_Date__c	TRUE	TRUE

Field	Editable	Readable
Backup_Approver__c.Comment__c	TRUE	TRUE
Backup_Approver__c.Current_User__c	TRUE	TRUE
Backup_Approver__c.DelegateUserIds__c	TRUE	TRUE
Backup_Approver__c.Effective_Date__c	TRUE	TRUE
Backup_Approver__c.Expiration_Date__c	TRUE	TRUE
Backup_Approver__c.InEffect__c	TRUE	TRUE
Backup_Approver__c.IsActive__c	TRUE	TRUE
Backup_Approver__c.IsDelegate__c	TRUE	TRUE
Backup_Approver__c.SuppressDelegateNotification__c	TRUE	TRUE
Backup_Approver__c.Transfer_in_flight__c	TRUE	TRUE
FormulaField__c.BusinessObject__c	TRUE	TRUE
FormulaField__c.Description__c	TRUE	TRUE
FormulaField__c.Formula__c	TRUE	TRUE
FormulaField__c.Type__c	TRUE	TRUE
Global_Discount_Policy__c.Approval_Level__c	TRUE	TRUE
Global_Discount_Policy__c.Authorized_Discount_Percent__c	TRUE	TRUE
Global_Discount_Policy__c.Description__c	TRUE	TRUE
Opportunity.Approval_Status__c	TRUE	TRUE

Field	Editable	Readable
ReminderInstance__c.ApprovalRequestId__c	TRUE	TRUE
ReminderInstance__c.Completed__c	TRUE	TRUE
ReminderInstance__c.Label__c	TRUE	TRUE
ReminderInstance__c.ProcessStepId__c	TRUE	TRUE
ReminderInstance__c.ScheduledFireTime__c	TRUE	TRUE
ReminderInstance__c.Status__c	TRUE	TRUE
ReminderInstance__c.StepSequence__c	TRUE	TRUE
ReminderInstance__c.StepType__c	TRUE	TRUE
ReminderInstance__c.SubprocessRuleEntryAssigneeId__c	TRUE	TRUE
ReminderInstance__c.SubprocessRuleEntryAssigneeSeq__c	TRUE	TRUE
Reminder__c.DurationUnit__c	TRUE	TRUE
Reminder__c.DurationValue__c	TRUE	TRUE
Reminder__c.Label__c	TRUE	TRUE
Reminder__c.ProcesssStepId__c	TRUE	TRUE
Reminder__c.StepSequence__c	TRUE	TRUE
Reminder__c.StepType__c	TRUE	TRUE
Reminder__c.SubprocessRuleEntryAssigneeId__c	TRUE	TRUE
Reminder__c.SubprocessRuleEntryAssigneeSeq__c	TRUE	TRUE

Field	Editable	Readable
Reminder__c.SubprocessRuleEntryId__c	TRUE	TRUE
SearchFilter__c.BusinessObject__c	TRUE	TRUE
SearchFilter__c.CriteriaFieldNames__c	TRUE	TRUE
SearchFilter__c.Criteria__c	TRUE	TRUE
SearchFilter__c.Description__c	TRUE	TRUE
SearchFilter__c.Sequence__c	TRUE	TRUE
SearchFilter__c.UseType__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Active__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Description__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Field__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Join__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Number__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Object_Qualifier__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Object__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Operator__c	TRUE	TRUE

Field	Editable	Readable
Term_Exception_Approval_Condition__c.TermExceptionApprovalId__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Term_Exception_Approval__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Value__c	TRUE	TRUE
Term_Exception_Approval__c.Active__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_Description__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_Id__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_Type__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_User__c	TRUE	TRUE
Term_Exception_Approval__c.AutoEscalate__c	TRUE	TRUE
Term_Exception_Approval__c.DependsOn__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToChain__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToId__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToName__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToType__c	TRUE	TRUE
Term_Exception_Approval__c.ExpectedDaysToComplete__c	TRUE	TRUE
Term_Exception_Approval__c.ExpectedHoursToComplete__c	TRUE	TRUE

Field	Editable	Readable
Term_Exception_Approval__c.ExpectedMinutesToComplete__c	TRUE	TRUE
Term_Exception_Approval__c.FilterLogic__c	TRUE	TRUE
Term_Exception_Approval__c.Notify_Only__c	TRUE	TRUE
Term_Exception_Approval__c.Send_Email__c	TRUE	TRUE
Term_Exception_Approval__c.Sequence__c	TRUE	TRUE
Term_Exception_Approval__c.SkipUnresolvedAssignee__c	TRUE	TRUE

For information on field permissions, refer to this Salesforce article.

Object Permissions for Administrators

The following table displays the permissions required by administrators for Approval Objects:

Object	allowCreate	allowDelete	allowEdit	allowRead	modifyAllRecords	viewAllRecords
AdhocApprovalGroup__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
AdhocApprovalProcess__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
AdhocApprover__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ApprovalJob__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ApprovalProcessInstance__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

Object	allowCreate	allowDelete	allowEdit	allowRead	modifyAllRecords	viewAllRecords
ApprovalRuleAssignee__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ApprovalRuleDimension__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ApprovalRuleEntry__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ApprovalRule__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Approval_Matrix__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Approval_Process__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Approval_Request_History__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Approval_Request__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Apttus__APTS_Admin__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Apttus__APTS_Agreement__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Apttus__Term_Exception__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
AsyncActionInfo__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Backup_Approver__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
FormulaField__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

Object	allowCreate	allowDelete	allowEdit	allowRead	modifyAllRecords	viewAllRecords
Global_Discount_Policy__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Opportunity	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ReminderInstance__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Reminder__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
SearchFilter__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Term_Exception_Approval_Condition__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Term_Exception_Approval__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

For information on setting object permissions, refer to this Salesforce article.

Page Access Permissions for Administrators

The following table displays the permissions required by administrators for Approval Pages:

Page Access	Enabled
AddApproverSearchMyApprovals	TRUE
AdhocProcessRuntime	TRUE
AgreementApprovalSummary	TRUE
AgreementApprovalSummarySf1	TRUE
AgreementLineItemApprovalSummary	TRUE

Page Access	Enabled
AgreementTermExAdd	TRUE
AgreementTermExApprovalSummary	TRUE
AgreementTermExApprovals	TRUE
AgreementTermExCancel	TRUE
AgreementTermExEdit	TRUE
AgreementTermExHistory	TRUE
AgreementTermExPreview	TRUE
AgreementTermExSubmit	TRUE
ApprovalAction	TRUE
ApprovalActionSf1	TRUE
ApprovalContextCancel	TRUE
ApprovalContextPreview	TRUE
ApprovalContextSubmit	TRUE
ApprovalMatrixGenerate	TRUE
ApprovalMatrixView	TRUE
ApprovalProcessDefn	TRUE
ApprovalProcessDefnList	TRUE
ApprovalProcessFinalAction	TRUE
ApprovalProcessInitialAction	TRUE
ApprovalProcessSelectEmailTemplate	TRUE

Page Access	Enabled
ApprovalProcessStep	TRUE
ApprovalProcessStepView	TRUE
ApprovalRequestAttachment	TRUE
ApprovalRequestEscalate	TRUE
ApprovalRequestReassign2	TRUE
ApprovalRequestReassign2Sf1	TRUE
ApprovalRequestsList	TRUE
ApprovalRequestsListSf1	TRUE
ApprovalRule	TRUE
ApprovalRuleCriteriaEdit	TRUE
ApprovalRuleDimension	TRUE
ApprovalRuleEntry	TRUE
ApprovalSummaryLaunch	TRUE
ApprovalSummaryLaunchAddCommentSf1	TRUE
ApprovalSummaryLaunchApproveRejectSf1	TRUE
ApprovalSummaryLaunchReassignSf1	TRUE
ApprovalsJSLibInclude	TRUE
BackupApprover	TRUE
BackupApproverAction	TRUE

Page Access	Enabled
BackupUserSearch	TRUE
DelegateApprovers	TRUE
FormulaField	TRUE
MatrixUserSearch	TRUE
MyApprovals	TRUE
MyApprovalsLaunch	TRUE
MyApprovalsSf1	TRUE
NextLevelApprover	TRUE
OpportunityApprovalSummary	TRUE
OpportunityPartnersList	TRUE
OpportunityProductsList	TRUE
PreviewSubmitApprovals	TRUE
QueueSearch	TRUE
ReassignUserSearch	TRUE
ReassignUserSearchMyApprovals	TRUE
SObjectApprovals2Submit	TRUE
SObjectApprovals2SubmitSf1	TRUE
SearchFilter	TRUE
SearchFilterDetail	TRUE
TermExApproval	TRUE

Page Access	Enabled
TermExApprovalClone	TRUE

For information on setting page access permissions, refer to this Salesforce article.

Record Type Visibility Permissions for Administrators

The following table displays the permissions required by administrators for Approval Record Types:

Record Type Visibility	Enabled
Approval_Process__c.ApprovalProcess	TRUE
Approval_Process__c.EntryCriteria	TRUE
Approval_Process__c.FinalAction	TRUE
Approval_Process__c.InitialSubmissionAction	TRUE
Approval_Process__c.Instance	TRUE
Approval_Process__c.Step	TRUE
Approval_Process__c.StepFilterCriteria	TRUE
Approval_Process__c.StepGroup	TRUE
Approval_Process__c.StepGroupAction	TRUE

For information on setting record type visibilities, refer to this Salesforce article.

Tab Visibility Permissions for Administrators

The following table displays the permissions required by administrators for Tab Settings:

Tab Setting	Visibility
ApprovalRuleDimension__c	Visible

Tab Setting	Visibility
ApprovalRule__c	Visible
Approval_Matrix__c	Visible
Approval_Process__c	Visible
Approval_Request__c	Visible
Apttus__APTS_Admin__c	Visible
Apttus__APTS_Agreement__c	Visible
Apttus__Term_Exception__c	Visible
Backup_Approver__c	Visible
FormulaField__c	Visible
Global_Discount_Policy__c	Visible
SearchFilter__c	Visible
Term_Exception_Approval__c	Visible

For information on tab settings, refer to this Salesforce article.

Recommended Permissions for Approval End Users

Select one of the following recommended permission topics for End Users:

- Application Permissions for Users
- Apex Class Permissions for Users
- Custom Setting Permissions for Users
- Field Permissions for Users
- Object Permissions for Users
- Page Access Permissions for Users
- Record Type Visibility Permissions for Users
- Tab Visibility Permissions for Users

Application Permissions for Users

The following table displays the permissions required by users for Approval Applications:

Application	Visibility
Conga Approvals	TRUE
Conga Approvals Setup	TRUE
Apttus__ApttusContractManagement	TRUE
Apttus__ApttusContractManagement_Lightning	TRUE

For information on setting application permissions, refer to this Salesforce article

Apex Class Permissions for Users

The following table displays the permissions required by administrators for Users:

Apex Class	Enabled
AP	TRUE
AbstractAdhocApprovalEngine	TRUE
AbstractAgreementTermExActionController	TRUE
AbstractApprovalContextActionController	TRUE
AbstractApprovalEmailController	TRUE
AbstractApprovalEngine	TRUE
AbstractApprovalEngineProcessTerm	TRUE
AbstractApprovalEngineTest	TRUE
AbstractApprovalsTest	TRUE

Apex Class	Enabled
AbstractEngine	TRUE
AbstractQueryCallback	TRUE
AbstractSObjectApprovals	TRUE
AbstractSObjectApprovals2	TRUE
AdhocApprovalEngine	TRUE
AdhocApprovalGroup	TRUE
AdhocApprovalGroupFactory	TRUE
AdhocApprovalProcess	TRUE
AdhocApprovalProcessDTO	TRUE
AdhocApprovalProcessFactory	TRUE
AdhocApprovalTest	TRUE
AdhocApprovalTestSupport	TRUE
AdhocApprover	TRUE
AdhocApproverFactory	TRUE
AdhocProcessRuntimeController	TRUE
AdhocProcessRuntimeControllerTest	TRUE
AdhocProcessRuntimeSupport	TRUE
AgreementApprovalEmailController	TRUE
AgreementApprovalEmailControllerTest	TRUE
AgreementApprovalSummaryController	TRUE

Apex Class	Enabled
AgreementApprovalSummaryControllerTest	TRUE
AgreementLIApprovalEmailControllerTest	TRUE
AgreementLIApprovalSummaryControllerTest	TRUE
AgreementLItemApprovalSummaryController	TRUE
AgreementLineItemApprovalEmailController	TRUE
AgreementTEApprovalEmailControllerTest	TRUE
AgreementTermExAddController	TRUE
AgreementTermExAddTest	TRUE
AgreementTermExApprovalEmailController	TRUE
AgreementTermExApprovalSummaryController	TRUE
AgreementTermExApprovalSummaryTest	TRUE
AgreementTermExApprovalsController	TRUE
AgreementTermExApprovalsTest	TRUE
AgreementTermExCancelController	TRUE
AgreementTermExCancelTest	TRUE
AgreementTermExEditController	TRUE
AgreementTermExEditTest	TRUE
AgreementTermExHistoryController	TRUE
AgreementTermExHistoryTest	TRUE

Apex Class	Enabled
AgreementTermExPreviewController	TRUE
AgreementTermExPreviewTest	TRUE
AgreementTermExSubmitController	TRUE
AgreementTermExSubmitTest	TRUE
AgreementTermExWrapper	TRUE
ApprovalActionController2	TRUE
ApprovalActionTest2	TRUE
ApprovalAssigneeResolver	TRUE
ApprovalChildProcess	TRUE
ApprovalConfig	TRUE
ApprovalConfigTest	TRUE
ApprovalConstants	TRUE
ApprovalContext	TRUE
ApprovalContextCancelController	TRUE
ApprovalContextCancelTest	TRUE
ApprovalContextFactory	TRUE
ApprovalContextPreviewController	TRUE
ApprovalContextPreviewTest	TRUE
ApprovalContextSubmitController	TRUE
ApprovalContextSubmitTest	TRUE

Apex Class	Enabled
ApprovalCustomConfig	TRUE
ApprovalCustomConfigTest	TRUE
ApprovalData	TRUE
ApprovalDataTest	TRUE
ApprovalDataTransferAgent	TRUE
ApprovalDeleteSupport	TRUE
ApprovalEmailHandler	TRUE
ApprovalEmailHandlerTest	TRUE
ApprovalEngine2	TRUE
ApprovalEngineBatchActionHandler	TRUE
ApprovalEngineConsolidationSupport	TRUE
ApprovalEngineFactory	TRUE
ApprovalEnginePrescanSupport	TRUE
ApprovalEnginePreviewHelper	TRUE
ApprovalEngineProcessAgreementTermEx	TRUE
ApprovalEngineProcessContextObject	TRUE
ApprovalEngineProcessContextQueueable	TRUE
ApprovalEngineProcessDealTerm	TRUE
ApprovalEngineProcessRequest	TRUE

Apex Class	Enabled
ApprovalEngineProcessRequestQueueable	TRUE
ApprovalEngineStepSupport	TRUE
ApprovalEngineSubmitHelper	TRUE
ApprovalEngineSupport	TRUE
ApprovalEngineTestATE	TRUE
ApprovalEngineTestATESupport	TRUE
ApprovalEngineTestAgmt	TRUE
ApprovalEngineTestAgmtSupport	TRUE
ApprovalEngineTestOppty	TRUE
ApprovalEngineTestOpptySupport	TRUE
ApprovalHistoryController	TRUE
ApprovalHistoryControllerTest	TRUE
ApprovalHistorySupport	TRUE
ApprovalMatrixAssigneeSupport	TRUE
ApprovalMatrixGenerateController	TRUE
ApprovalMatrixGenerateTest	TRUE
ApprovalMatrixGenerator	TRUE
ApprovalMatrixResolver	TRUE
ApprovalMatrixSupport	TRUE
ApprovalMatrixViewController	TRUE

Apex Class	Enabled
ApprovalMatrixViewTest	TRUE
ApprovalPolicySupport	TRUE
ApprovalProcess2	TRUE
ApprovalProcessDefnController	TRUE
ApprovalProcessDefnControllerTest	TRUE
ApprovalProcessDefnListController	TRUE
ApprovalProcessDefnListTest	TRUE
ApprovalProcessFactory	TRUE
ApprovalProcessFactoryTest	TRUE
ApprovalProcessFinalActionController	TRUE
ApprovalProcessFinalActionTest	TRUE
ApprovalProcessInitialActionController	TRUE
ApprovalProcessInitialActionTest	TRUE
ApprovalProcessInstance	TRUE
ApprovalProcessInstanceFactory	TRUE
ApprovalProcessQryHelper	TRUE
ApprovalProcessResolver	TRUE
ApprovalProcessResolverTest	TRUE
ApprovalProcessStepController	TRUE

Apex Class	Enabled
ApprovalProcessStepControllerTest	TRUE
ApprovalProcessSupport	TRUE
ApprovalProcessTest2	TRUE
ApprovalQryHelper	TRUE
ApprovalRequestAttachmentController	TRUE
ApprovalRequestAttachmentTest	TRUE
ApprovalRequestEscalateController	TRUE
ApprovalRequestEscalateTest	TRUE
ApprovalRequestQryHelper	TRUE
ApprovalRequestQryHelperTest	TRUE
ApprovalRequestReassignController2	TRUE
ApprovalRequestReassignTest	TRUE
ApprovalRequestSupport	TRUE
ApprovalRequestsListController	TRUE
ApprovalRequestsListTest	TRUE
ApprovalRequiredCheck	TRUE
ApprovalRequiredCheckTest	TRUE
ApprovalRule2	TRUE
ApprovalRuleAssignee	TRUE
ApprovalRuleAssigneeFactory	TRUE

Apex Class	Enabled
ApprovalRuleController	TRUE
ApprovalRuleControllerTest	TRUE
ApprovalRuleCriteriaEditController	TRUE
ApprovalRuleCriteriaEditControllerTest	TRUE
ApprovalRuleDimension	TRUE
ApprovalRuleDimensionController	TRUE
ApprovalRuleDimensionControllerTest	TRUE
ApprovalRuleDimensionFactory	TRUE
ApprovalRuleDimensionSupport	TRUE
ApprovalRuleEntry	TRUE
ApprovalRuleEntryController	TRUE
ApprovalRuleEntryControllerTest	TRUE
ApprovalRuleEntryFactory	TRUE
ApprovalRuleFactory	TRUE
ApprovalRuleSupport	TRUE
ApprovalRuleTest	TRUE
ApprovalStatusUpdateBatchJob	TRUE
ApprovalStatusUpdateBatchJobTest	TRUE
ApprovalSubprocess	TRUE

Apex Class	Enabled
ApprovalSubstep	TRUE
ApprovalSummaryLaunchController	TRUE
ApprovalSummaryLaunchTest	TRUE
ApprovalSystemException	TRUE
ApprovalUserSupport	TRUE
ApprovalUserSupportTest	TRUE
ApprovalUtil2	TRUE
ApprovalsController	TRUE
ApprovalsControllerTest	TRUE
ApprovalsWebService	TRUE
ApprovalsWebServiceSupport	TRUE
ApprovalsWebServiceTest	TRUE
Assignee	TRUE
AssigneeCacheSupport	TRUE
AsyncActionInfoEmailController	TRUE
AsyncActionSupport	TRUE
AsyncActionSupportTest	TRUE
AsyncJobSupport	TRUE
AttachmentSupport	TRUE
AttachmentWrapper	TRUE

Apex Class	Enabled
BackupAdminSupport	TRUE
BackupAdminSupportTest	TRUE
BackupApproverActionController	TRUE
BackupApproverActionTest	TRUE
BackupApproverController	TRUE
BackupApproverSupport	TRUE
BackupApproverTest	TRUE
BackupUserSearchController	TRUE
BackupUserSearchTest	TRUE
BatchPreviewApprovals	TRUE
BatchSubmitForApprovals	TRUE
BulkActionRequest	TRUE
BulkCtxObjectParam	TRUE
ChildSOSupport	TRUE
ClauseApprovalsWebService	TRUE
ClauseApprovalsWebServiceTest	TRUE
ContextApprovalEmailController	TRUE
ContextApprovalEmailControllerTest	TRUE
CriteriaCache	TRUE

Apex Class	Enabled
CriteriaCacheTest	TRUE
CurrencySupport	TRUE
CurrencySupportTest	TRUE
CustomAssigneeInfo	TRUE
CustomAssigneeInfoTest	TRUE
CustomAssigneeSupport	TRUE
CustomClass	TRUE
CustomClassLoader	TRUE
CustomSOSupport	TRUE
DashboardSupport	TRUE
DashboardSupportTest	TRUE
DefaultQueryCallback	TRUE
ESAPISupport	TRUE
EmailRecipient	TRUE
EmailSupport	TRUE
EmailTemplateConstants	TRUE
EmailTemplateFactory	TRUE
EmailTemplateSupport	TRUE
EmailTemplateTest	TRUE
EscalationAssignee	TRUE

Apex Class	Enabled
EscalationReminderEmailController	TRUE
EscalationReminderEmailControllerTest	TRUE
EscalationSupport	TRUE
EscalationWorkflowTest	TRUE
ExprSupport	TRUE
ExprTest	TRUE
FieldMetadata	TRUE
FileSupport	TRUE
FileSupportTest	TRUE
FormulaField	TRUE
FormulaFieldController	TRUE
FormulaFieldControllerTest	TRUE
FormulaFieldFactory	TRUE
FormulaFieldSupport	TRUE
IApprovalEngine	TRUE
InboundAttachmentHandlerTest	TRUE
InstanceUrlController	TRUE
LookupUserSupport	TRUE
ManagedObject	TRUE

Apex Class	Enabled
ManagedObjectConstants	TRUE
ManagedObjectTest	TRUE
MiscClassTest	TRUE
MyApprovalsController	TRUE
MyApprovalsControllerTest	TRUE
MyApprovalsEmailController	TRUE
MyApprovalsEmailControllerTest	TRUE
MyApprovalsLaunchController	TRUE
MyApprovalsLaunchControllerTest	TRUE
NextLevelApproverController	TRUE
NextLevelApproverTest	TRUE
NullObject	TRUE
ObjectCache	TRUE
ObjectValue	TRUE
ObjectValueTest	TRUE
OneOrMoreItemsFailedException	TRUE
OpportunityApprovalSummaryController	TRUE
OpportunityApprovalSummaryTest	TRUE
OpportunityPartnersListController	TRUE
OpportunityPartnersListTest	TRUE

Apex Class	Enabled
OpportunityProductsListController	TRUE
OpportunityProductsListTest	TRUE
OpptyApprovalEmailController	TRUE
OpptyApprovalEmailControllerTest	TRUE
PostInstallBatchApprovalProcesses	TRUE
PostInstallBatchApprovalRequests	TRUE
PostInstallBatchBackupApprovers	TRUE
PostInstallBatchCopyProcessId	TRUE
PostInstallBatchUpdDelegateApproverNames	TRUE
PostInstallScript	TRUE
PostInstallScriptTest	TRUE
PreviewSubmitApprovalsController	TRUE
PreviewSubmitApprovalsTest	TRUE
Property	TRUE
QuerySpec	TRUE
QuerySpecFactory	TRUE
ReassignUserSearchController	TRUE
ReassignUserSearchTest	TRUE
RelatedUserSupport	TRUE

Apex Class	Enabled
ReminderInstanceSupport	TRUE
ReminderJob	TRUE
ReminderJobTest	TRUE
ReminderSupport	TRUE
ReminderTest	TRUE
ReminderTestSupport	TRUE
RuntimeContext	TRUE
SOQLConstants	TRUE
SOQLSupport	TRUE
SOQLTest	TRUE
SObjectApprovalContextParam	TRUE
SObjectApprovalContextParam2	TRUE
SObjectApprovals2SubmitController	TRUE
SObjectApprovals2SubmitTest	TRUE
SObjectApprovals2Test	TRUE
SObjectApprovalsController	TRUE
SObjectApprovalsController2	TRUE
SObjectApprovalsSample	TRUE
SObjectApprovalsSupport	TRUE
SObjectApprovalsTest	TRUE

Apex Class	Enabled
SObjectConstants	TRUE
SObjectDescribeInfo	TRUE
SObjectMetadata	TRUE
SObjectMetadataTest	TRUE
SObjectSupport	TRUE
SObjectTest	TRUE
SearchFilter	TRUE
SearchFilterController	TRUE
SearchFilterControllerTest	TRUE
SearchFilterDetailController	TRUE
SearchFilterDetailControllerTest	TRUE
StepRule	TRUE
SubmissionComments	TRUE
SubmissionCommentsSupport	TRUE
SystemUtil	TRUE
SystemUtilTest	TRUE
TaskConstants	TRUE
TaskFactory	TRUE
TaskSupport	TRUE

Apex Class	Enabled
TermExApproval	TRUE
TermExApprovalCloneController	TRUE
TermExApprovalCloneTest	TRUE
TermExApprovalController	TRUE
TermExApprovalControllerTest	TRUE
TermExApprovalFactory	TRUE
TermExApprovalFactoryTest	TRUE
TermExSupport	TRUE
TestSupport	TRUE
TransferInflightRequests	TRUE
TransferInflightRequestsQueueable	TRUE
UIAbstractSearchController	TRUE
UIAbstractSelectItemsController	TRUE
UIAssigneeController	TRUE
UIAssigneeControllerTest	TRUE
UIDataTableController	TRUE
UIDependsOnController	TRUE
UIDependsOnControllerTest	TRUE
UIDisplayFieldNamesController	TRUE
UIDisplayFieldNamesControllerTest	TRUE

Apex Class	Enabled
UIEscalationAssigneeController	TRUE
UIPageConfig	TRUE
UIPageConfigTest	TRUE
UIPageControllerBase	TRUE
UIPageControllerBaseTest	TRUE
UIPageSupport	TRUE
UIReapprovalFilterController	TRUE
UIReapprovalFilterControllerTest	TRUE
UISearchControllerTest	TRUE
UISearchEmailTemplateController	TRUE
UISearchFilterController	TRUE
UISearchFilterControllerTest	TRUE
UISearchQueueController	TRUE
UISearchRoleController	TRUE
UISearchRuleController	TRUE
UISearchUserController	TRUE
UserContext	TRUE

For information on setting apex class permissions, refer to this Salesforce article.

Custom Setting Permissions for Users

The following table displays the permissions required by users for Approval Custom Settings:

Custom Setting Name	Enabled
ApprovalsCustomClasses__c	TRUE
ApprovalsCustomConfig__c	TRUE
ApprovalsSystemProperties__c	TRUE

For information on setting custom setting permissions, refer to this Salesforce article.

Field Permissions for Users

The following table displays the permissions required by users for Approval Fields:

Field	Editable	Readble
AdhocApprovalGroup__c.DependsOn__c	TRUE	TRUE
AdhocApprovalGroup__c.GroupName__c	TRUE	TRUE
AdhocApprovalGroup__c.GroupSequence__c	TRUE	TRUE
AdhocApprovalProcess__c.AttachmentIds__c	TRUE	TRUE
AdhocApprovalProcess__c.BusinessObjectId__c	TRUE	TRUE
AdhocApprovalProcess__c.BusinessObjectType__c	TRUE	TRUE
AdhocApprovalProcess__c.DisplayFields__c	TRUE	TRUE
AdhocApprovalProcess__c.DisplayHeaderFields__c	TRUE	TRUE
AdhocApprovalProcess__c.ProcessComments__c	TRUE	TRUE

Field	Editable	Readble
AdhocApprover__c.ApproverSequence__c	TRUE	TRUE
AdhocApprover__c.AssigneeId__c	TRUE	TRUE
AdhocApprover__c.AssigneeType__c	TRUE	TRUE
AdhocApprover__c.AssigneeValue__c	TRUE	TRUE
AdhocApprover__c.AutoReapprovalEnabled__c	TRUE	TRUE
AdhocApprover__c.DependsOn__c	TRUE	TRUE
AdhocApprover__c.IsReviewer__c	TRUE	TRUE
AdhocApprover__c.SendEmail__c	TRUE	TRUE
ApprovalJob__c.ApexChildJobId__c	TRUE	TRUE
ApprovalJob__c.ApexParentJobId__c	TRUE	TRUE
ApprovalJob__c.ApprovalRequestId__c	TRUE	TRUE
ApprovalJob__c.Status__c	TRUE	TRUE
ApprovalProcessInstance__c.AssignmentEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.AttachmentIds__c	TRUE	TRUE
ApprovalProcessInstance__c.BusinessObjectLink__c	FALSE	TRUE
ApprovalProcessInstance__c.BusinessObjectType__c	TRUE	TRUE
ApprovalProcessInstance__c.CancellationEmailTemplate__c	TRUE	TRUE

Field	Editable	Readble
ApprovalProcessInstance__c.ConsolidationVersionNumber__c	TRUE	TRUE
ApprovalProcessInstance__c.Data__c	TRUE	TRUE
ApprovalProcessInstance__c.EndTime__c	TRUE	TRUE
ApprovalProcessInstance__c.EscalationEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.InstanceNumber__c	FALSE	TRUE
ApprovalProcessInstance__c.NotifyOnlyEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.OriginalParentApprovalProcessId__c	TRUE	TRUE
ApprovalProcessInstance__c.PrevProcessInstanceId__c	TRUE	TRUE
ApprovalProcessInstance__c.ReassignmentEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.ReminderEmailTemplate__c	TRUE	TRUE
ApprovalProcessInstance__c.StartTime__c	TRUE	TRUE
ApprovalProcessInstance__c.Status__c	TRUE	TRUE
ApprovalRuleAssignee__c.AssigneeAutoEscalate__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeDescription__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeEscalateToId__c	FALSE	TRUE

Field	Editable	Readble
ApprovalRuleAssignee__c.AssigneeEscalateToName__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeEscalateToType__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeExpectedDaysToComplete__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeExpectedHoursToComplete__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeId__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeLabel__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeType__c	FALSE	TRUE
ApprovalRuleAssignee__c.AssigneeValue__c	FALSE	TRUE
ApprovalRuleAssignee__c.AutoComplete__c	FALSE	TRUE
ApprovalRuleAssignee__c.DependsOn__c	FALSE	TRUE
ApprovalRuleAssignee__c.NotifyOnly__c	FALSE	TRUE
ApprovalRuleAssignee__c.SendEmail__c	FALSE	TRUE
ApprovalRuleAssignee__c.SendReminders__c	FALSE	TRUE
ApprovalRuleAssignee__c.Sequence__c	FALSE	TRUE
ApprovalRuleAssignee__c.SkipUnresolvedAssignee__c	FALSE	TRUE
ApprovalRuleDimension__c.BusinessObject__c	FALSE	TRUE
ApprovalRuleDimension__c.Description__c	FALSE	TRUE

Field	Editable	Readble
ApprovalRuleDimension__c.DimensionType__c	FALSE	TRUE
ApprovalRuleEntry__c.AutoReapprovalCriteria__c	FALSE	TRUE
ApprovalRuleEntry__c.AutoReapprove__c	FALSE	TRUE
ApprovalRuleEntry__c.ConditionFieldNames__c	FALSE	TRUE
ApprovalRuleEntry__c.Condition__c	FALSE	TRUE
ApprovalRuleEntry__c.CriteriaFieldNames__c	FALSE	TRUE
ApprovalRuleEntry__c.DependsOn__c	FALSE	TRUE
ApprovalRuleEntry__c.Description__c	FALSE	TRUE
ApprovalRuleEntry__c.Dimension1Value__c	FALSE	TRUE
ApprovalRuleEntry__c.Dimension2Value__c	FALSE	TRUE
ApprovalRuleEntry__c.Dimension3Value__c	FALSE	TRUE
ApprovalRuleEntry__c.Dimension4Value__c	FALSE	TRUE
ApprovalRuleEntry__c.Dimension5Value__c	FALSE	TRUE
ApprovalRuleEntry__c.Dimension6Value__c	FALSE	TRUE
ApprovalRuleEntry__c.EntryLabel__c	FALSE	TRUE
ApprovalRuleEntry__c.IncludePrevious__c	FALSE	TRUE
ApprovalRuleEntry__c.SelectValueType__c	FALSE	TRUE
ApprovalRuleEntry__c.SelectValue__c	FALSE	TRUE
ApprovalRule__c.Active__c	FALSE	TRUE

Field	Editable	Readble
ApprovalRule__c.ApprovalCount__c	FALSE	TRUE
ApprovalRule__c.ApprovalPercent__c	FALSE	TRUE
ApprovalRule__c.ApprovalPolicy__c	FALSE	TRUE
ApprovalRule__c.BusinessObject__c	FALSE	TRUE
ApprovalRule__c.ContinuePolicyApprovalOnARej ect__c	FALSE	TRUE
ApprovalRule__c.CriteriaFieldNames__c	FALSE	TRUE
ApprovalRule__c.Criteria__c	FALSE	TRUE
ApprovalRule__c.Description__c	FALSE	TRUE
ApprovalRule__c.Dimension1Id__c	FALSE	TRUE
ApprovalRule__c.Dimension1ValueType__c	FALSE	TRUE
ApprovalRule__c.Dimension2Id__c	FALSE	TRUE
ApprovalRule__c.Dimension2ValueType__c	FALSE	TRUE
ApprovalRule__c.Dimension3Id__c	FALSE	TRUE
ApprovalRule__c.Dimension3ValueType__c	FALSE	TRUE
ApprovalRule__c.Dimension4Id__c	FALSE	TRUE
ApprovalRule__c.Dimension4ValueType__c	FALSE	TRUE
ApprovalRule__c.Dimension5Id__c	FALSE	TRUE
ApprovalRule__c.Dimension5ValueType__c	FALSE	TRUE
ApprovalRule__c.Dimension6Id__c	FALSE	TRUE

Field	Editable	Readble
ApprovalRule__c.Dimension6ValueType__c	FALSE	TRUE
ApprovalRule__c.EffectiveDate__c	FALSE	TRUE
ApprovalRule__c.ExpirationDate__c	FALSE	TRUE
ApprovalRule__c.MatchRule__c	FALSE	TRUE
ApprovalRule__c.RuleType__c	FALSE	TRUE
ApprovalRule__c.StopProcessingMoreRules__c	FALSE	TRUE
Approval_Matrix__c.Approval_Level__c	FALSE	TRUE
Approval_Matrix__c.Approver_1_Authorized_Discount__c	FALSE	TRUE
Approval_Matrix__c.Approver_1_Level__c	FALSE	TRUE
Approval_Matrix__c.Approver_1__c	FALSE	TRUE
Approval_Matrix__c.Approver_2_Authorized_Discount__c	FALSE	TRUE
Approval_Matrix__c.Approver_2_Level__c	FALSE	TRUE
Approval_Matrix__c.Approver_2__c	FALSE	TRUE
Approval_Matrix__c.Approver_3_Authorized_Discount__c	FALSE	TRUE
Approval_Matrix__c.Approver_3_Level__c	FALSE	TRUE
Approval_Matrix__c.Approver_3__c	FALSE	TRUE
Approval_Matrix__c.Approver_4_Authorized_Discount__c	FALSE	TRUE
Approval_Matrix__c.Approver_4_Level__c	FALSE	TRUE

Field	Editable	Readble
Approval_Matrix__c.Approver_4__c	FALSE	TRUE
Approval_Matrix__c.Approver_5_Authorized_Discount__c	FALSE	TRUE
Approval_Matrix__c.Approver_5_Level__c	FALSE	TRUE
Approval_Matrix__c.Approver_5__c	FALSE	TRUE
Approval_Matrix__c.Approver_6_Authorized_Discount__c	FALSE	TRUE
Approval_Matrix__c.Approver_6_Level__c	FALSE	TRUE
Approval_Matrix__c.Approver_6__c	FALSE	TRUE
Approval_Matrix__c.Description__c	FALSE	TRUE
Approval_Matrix__c.Error_Log__c	FALSE	TRUE
Approval_Matrix__c.Is_Active__c	FALSE	TRUE
Approval_Matrix__c.Is_Valid__c	FALSE	TRUE
Approval_Matrix__c.Matrix_Owner__c	FALSE	TRUE
Approval_Matrix__c.Next_Level_Approver__c	FALSE	TRUE
Approval_Matrix__c.User__c	FALSE	TRUE
Approval_Process__c.Active__c	FALSE	TRUE
Approval_Process__c.AdhocAllowApprovedRequestsToBeDeleted__c	FALSE	TRUE
Approval_Process__c.AdhocAutoReapprovalStatusLabel__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.AdhocStepAutoReapprove__c	FALSE	TRUE
Approval_Process__c.ApprovalCommentsMandatory__c	FALSE	TRUE
Approval_Process__c.ApprovalSummaryPage__c	FALSE	TRUE
Approval_Process__c.Assignment_Email_Template__c	FALSE	TRUE
Approval_Process__c.BackupAdminUser__c	FALSE	TRUE
Approval_Process__c.CancelPendingRequestsOnAReject__c	FALSE	TRUE
Approval_Process__c.Cancellation_Email_Template__c	FALSE	TRUE
Approval_Process__c.ChildObjectType__c	FALSE	TRUE
Approval_Process__c.ConsolidateApprovals__c	FALSE	TRUE
Approval_Process__c.ConsolidateNotifications__c	FALSE	TRUE
Approval_Process__c.ContextType__c	FALSE	TRUE
Approval_Process__c.ContinuePendingApprovalsOnAReject__c	FALSE	TRUE
Approval_Process__c.CriteriaFieldNames__c	FALSE	TRUE
Approval_Process__c.CustomMyApprovalsPage__c	FALSE	TRUE
Approval_Process__c.CustomPreviewPage__c	FALSE	TRUE
Approval_Process__c.DependsOn__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.Description__c	FALSE	TRUE
Approval_Process__c.EntryCriteriaFilterLogic__c	FALSE	TRUE
Approval_Process__c.Entry_Criteria_Active__c	FALSE	TRUE
Approval_Process__c.Entry_Criteria_BoolOper__c	FALSE	TRUE
Approval_Process__c.Entry_Criteria_Comparison_Type__c	FALSE	TRUE
Approval_Process__c.Entry_Criteria_Field_Value__c	FALSE	TRUE
Approval_Process__c.Entry_Criteria_Field__c	FALSE	TRUE
Approval_Process__c.Entry_Criteria__c	FALSE	TRUE
Approval_Process__c.Escalation_Email_Template__c	FALSE	TRUE
Approval_Process__c.Final_Action_Field_Name_Source__c	FALSE	TRUE
Approval_Process__c.Final_Action_Field_Name__c	FALSE	TRUE
Approval_Process__c.Final_Action_Field_Update_Type__c	FALSE	TRUE
Approval_Process__c.Final_Action_Field_Value__c	FALSE	TRUE
Approval_Process__c.Final_Action_Type__c	FALSE	TRUE
Approval_Process__c.Final_Approval_Action__c	FALSE	TRUE
Approval_Process__c.Group_Action__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.Initial_Submission_Action_Type__c	FALSE	TRUE
Approval_Process__c.Initial_Submission_Action__c	FALSE	TRUE
Approval_Process__c.Initial_Submission_Field_Name_Source__c	FALSE	TRUE
Approval_Process__c.Initial_Submission_Field_Name__c	FALSE	TRUE
Approval_Process__c.Initial_Submission_Field_Value__c	FALSE	TRUE
Approval_Process__c.IsAdhoc__c	FALSE	TRUE
Approval_Process__c.NotifyOnly_Email_Template__c	FALSE	TRUE
Approval_Process__c.Object_Label__c	FALSE	TRUE
Approval_Process__c.Object_Name__c	FALSE	TRUE
Approval_Process__c.Object_Type__c	FALSE	TRUE
Approval_Process__c.OriginalInstanceProcessId__c	FALSE	TRUE
Approval_Process__c.ProcessAutoEscalate__c	FALSE	TRUE
Approval_Process__c.ProcessEscalateToId__c	FALSE	TRUE
Approval_Process__c.ProcessEscalateToName__c	FALSE	TRUE
Approval_Process__c.ProcessEscalateToType__c	FALSE	TRUE
Approval_Process__c.ProcessExpectedDaysToComplete__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.ProcessExpectedHoursToComplete__c	FALSE	TRUE
Approval_Process__c.ProcessExpectedMinutesToComplete__c	FALSE	TRUE
Approval_Process__c.Process_Name__c	FALSE	TRUE
Approval_Process__c.Reassignment_Email_Template__c	FALSE	TRUE
Approval_Process__c.RejectionCommentsMandatory__c	FALSE	TRUE
Approval_Process__c.Reminder_Email_Template__c	FALSE	TRUE
Approval_Process__c.Send_Email__c	FALSE	TRUE
Approval_Process__c.Sequence__c	FALSE	TRUE
Approval_Process__c.StepAutoEscalate__c	FALSE	TRUE
Approval_Process__c.StepAutoReapprovalCriteria__c	FALSE	TRUE
Approval_Process__c.StepAutoReapprove__c	FALSE	TRUE
Approval_Process__c.StepCarbonCopyAssigneeIds__c	FALSE	TRUE
Approval_Process__c.StepCarbonCopyUser__c	FALSE	TRUE
Approval_Process__c.StepDisplayFieldNameHeader__c	FALSE	TRUE
Approval_Process__c.StepDisplayFieldNames__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.StepEscalateToId__c	FALSE	TRUE
Approval_Process__c.StepEscalateToName__c	FALSE	TRUE
Approval_Process__c.StepEscalateToType__c	FALSE	TRUE
Approval_Process__c.StepExpectedDaysToComplete__c	FALSE	TRUE
Approval_Process__c.StepExpectedHoursToComplete__c	FALSE	TRUE
Approval_Process__c.StepExpectedMinutesToComplete__c	FALSE	TRUE
Approval_Process__c.StepFilterConditionDescription__c	FALSE	TRUE
Approval_Process__c.StepFilterLogic__c	FALSE	TRUE
Approval_Process__c.StepKeyFieldNames__c	FALSE	TRUE
Approval_Process__c.StepLabel__c	FALSE	TRUE
Approval_Process__c.StepRemindersActive__c	FALSE	TRUE
Approval_Process__c.StepSequence__c	FALSE	TRUE
Approval_Process__c.StepSkipUnresolvedAssignee__c	FALSE	TRUE
Approval_Process__c.StepSubmissionComment1Enabled__c	FALSE	TRUE
Approval_Process__c.StepSubmissionComment2Enabled__c	FALSE	TRUE
Approval_Process__c.StepSubmissionComment3Enabled__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.StepSubmissionCommentsEnabled__c	FALSE	TRUE
Approval_Process__c.StepType__c	FALSE	TRUE
Approval_Process__c.Step_Active__c	FALSE	TRUE
Approval_Process__c.Step_Approver_Type__c	FALSE	TRUE
Approval_Process__c.Step_Assignee_Description__c	FALSE	TRUE
Approval_Process__c.Step_Assignee_Id__c	FALSE	TRUE
Approval_Process__c.Step_Assignee_Type__c	FALSE	TRUE
Approval_Process__c.Step_Assignee__c	FALSE	TRUE
Approval_Process__c.Step_Auto_Complete__c	FALSE	TRUE
Approval_Process__c.Step_Filter_BoolOper__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Comments__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Comparison_Type__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Field_Number__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Field_Object_Source__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Field_Object__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Field_Value_Object_Source__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.Step_Filter_Field_Value_Object__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Field_Value__c	FALSE	TRUE
Approval_Process__c.Step_Filter_Field__c	FALSE	TRUE
Approval_Process__c.Step_Filter__c	FALSE	TRUE
Approval_Process__c.Step_Group_Action_Field_Name_Source__c	FALSE	TRUE
Approval_Process__c.Step_Group_Action_Field_Name__c	FALSE	TRUE
Approval_Process__c.Step_Group_Action_Field_Update_Type__c	FALSE	TRUE
Approval_Process__c.Step_Group_Action_Field_Value__c	FALSE	TRUE
Approval_Process__c.Step_Group_Action_Type__c	FALSE	TRUE
Approval_Process__c.Step_Group_Action__c	FALSE	TRUE
Approval_Process__c.Step_Group_Active__c	FALSE	TRUE
Approval_Process__c.Step_Group_Name__c	FALSE	TRUE
Approval_Process__c.Step_Group_Rejection_Action__c	FALSE	TRUE
Approval_Process__c.Step_Group_Seq_Number__c	FALSE	TRUE
Approval_Process__c.Step_Group__c	FALSE	TRUE
Approval_Process__c.Step_Hierarchy_Driver__c	FALSE	TRUE

Field	Editable	Readble
Approval_Process__c.Step_Name__c	FALSE	TRUE
Approval_Process__c.Step_Notify_Only__c	FALSE	TRUE
Approval_Process__c.Step_Seq_Number__c	FALSE	TRUE
Approval_Process__c.Step__c	FALSE	TRUE
Approval_Process__c.SubmissionComment1Label__c	FALSE	TRUE
Approval_Process__c.SubmissionComment2Label__c	FALSE	TRUE
Approval_Process__c.SubmissionComment3Label__c	FALSE	TRUE
Approval_Process__c.SubmissionCommentsEnabledCount__c	FALSE	TRUE
Approval_Process__c.SubmissionCommentsEnabled__c	FALSE	TRUE
Approval_Process__c.SubmissionCommentsMandatory__c	FALSE	TRUE
Approval_Process__c.SubmissionCommentsPerStep__c	FALSE	TRUE
Approval_Process__c.SubmissionCommentsType__c	FALSE	TRUE
Approval_Request_History__c.Action_Approve_Id__c	TRUE	TRUE
Approval_Request_History__c.Action_Reassign_Id__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request_History__c.Action__c	FALSE	TRUE
Approval_Request_History__c.Active__c	TRUE	TRUE
Approval_Request_History__c.ActualApproverName__c	FALSE	TRUE
Approval_Request_History__c.Actual_Approver__c	TRUE	TRUE
Approval_Request_History__c.AgreementLineItemId__c	TRUE	TRUE
Approval_Request_History__c.ApprovalCount__c	TRUE	TRUE
Approval_Request_History__c.ApprovalFromEmail__c	TRUE	TRUE
Approval_Request_History__c.ApprovalPercent__c	TRUE	TRUE
Approval_Request_History__c.ApprovalPolicy__c	TRUE	TRUE
Approval_Request_History__c.Approval_Process__c	TRUE	TRUE
Approval_Request_History__c.Approval_Status__c	TRUE	TRUE
Approval_Request_History__c.Approver_Comments__c	TRUE	TRUE
Approval_Request_History__c.Assigned_To_Id__c	TRUE	TRUE
Approval_Request_History__c.Assigned_To_Link__c	FALSE	TRUE
Approval_Request_History__c.Assigned_To_Name__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request_History__c.Assigned_To_Type__c	TRUE	TRUE
Approval_Request_History__c.Assigned_To__c	TRUE	TRUE
Approval_Request_History__c.AutoEscalate__c	TRUE	TRUE
Approval_Request_History__c.AutoReapprove__c	TRUE	TRUE
Approval_Request_History__c.Auto_Complete__c	TRUE	TRUE
Approval_Request_History__c.Backup_From_User__c	TRUE	TRUE
Approval_Request_History__c.CanEscalate__c	TRUE	TRUE
Approval_Request_History__c.CarbonCopyAssigneelds__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectId__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectLink__c	FALSE	TRUE
Approval_Request_History__c.ChildObjectName__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectRefId__c	TRUE	TRUE
Approval_Request_History__c.ChildObjectType__c	TRUE	TRUE
Approval_Request_History__c.ContinuePolicyApprovalOnAReject__c	TRUE	TRUE
Approval_Request_History__c.CriteriaFieldNames__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request_History__c.DateApproved__c	TRUE	TRUE
Approval_Request_History__c.DateAssigned__c	TRUE	TRUE
Approval_Request_History__c.DateCancelled__c	TRUE	TRUE
Approval_Request_History__c.DateEscalated__c	TRUE	TRUE
Approval_Request_History__c.DateReassigned__c	TRUE	TRUE
Approval_Request_History__c.DateRejected__c	TRUE	TRUE
Approval_Request_History__c.Date__c	TRUE	TRUE
Approval_Request_History__c.DelegateApproverIds__c	TRUE	TRUE
Approval_Request_History__c.DelegateApproverNames__c	TRUE	TRUE
Approval_Request_History__c.DelegateApprover__c	TRUE	TRUE
Approval_Request_History__c.DependsOn__c	TRUE	TRUE
Approval_Request_History__c.EscalateToChain__c	TRUE	TRUE
Approval_Request_History__c.EscalateToId__c	TRUE	TRUE
Approval_Request_History__c.EscalateToName__c	TRUE	TRUE
Approval_Request_History__c.EscalateToType__c	TRUE	TRUE
Approval_Request_History__c.EscalatedToHighestLevel__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request_History__c.ExpectedCompletionDate__c	TRUE	TRUE
Approval_Request_History__c.ExpectedDaysToComplete__c	TRUE	TRUE
Approval_Request_History__c.ExpectedHoursToComplete__c	TRUE	TRUE
Approval_Request_History__c.ExpectedMinutesToComplete__c	TRUE	TRUE
Approval_Request_History__c.Group_Unique_Id__c	TRUE	TRUE
Approval_Request_History__c.Group__c	TRUE	TRUE
Approval_Request_History__c.HasAttachments__c	TRUE	TRUE
Approval_Request_History__c.HasDelegateApprover__c	TRUE	TRUE
Approval_Request_History__c.InEscalation__c	TRUE	TRUE
Approval_Request_History__c.Initial_Submitter__c	TRUE	TRUE
Approval_Request_History__c.Internal_Comments__c	TRUE	TRUE
Approval_Request_History__c.IsAdhocProcess__c	TRUE	TRUE
Approval_Request_History__c.IsAdhoc__c	TRUE	TRUE
Approval_Request_History__c.IsAutoReapprovalEnabled__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request_History__c.IsSubprocess__c	TRUE	TRUE
Approval_Request_History__c.Notify_Only__c	TRUE	TRUE
Approval_Request_History__c.ObjectRefId__c	TRUE	TRUE
Approval_Request_History__c.Object_Id_Link__c	FALSE	TRUE
Approval_Request_History__c.Object_Id__c	TRUE	TRUE
Approval_Request_History__c.Object_Name__c	TRUE	TRUE
Approval_Request_History__c.Object_Type__c	TRUE	TRUE
Approval_Request_History__c.ParentRequestId__c	TRUE	TRUE
Approval_Request_History__c.Parent_Agreement__c	TRUE	TRUE
Approval_Request_History__c.PrevAssignedToId__c	TRUE	TRUE
Approval_Request_History__c.PrevAssignedToName__c	TRUE	TRUE
Approval_Request_History__c.PrevAssignedToType__c	TRUE	TRUE
Approval_Request_History__c.ProcessInstanceId__c	TRUE	TRUE
Approval_Request_History__c.Rejection_Action__c	TRUE	TRUE
Approval_Request_History__c.Related_Agreement_Owner__c	TRUE	TRUE
Approval_Request_History__c.Related_Agreement_Requestor__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request_History__c.Related_Agreement_Term_Exception__c	TRUE	TRUE
Approval_Request_History__c.Related_Agreement__c	TRUE	TRUE
Approval_Request_History__c.Related_Opportunity_Owner__c	TRUE	TRUE
Approval_Request_History__c.Related_Opportunity__c	TRUE	TRUE
Approval_Request_History__c.RequestType__c	TRUE	TRUE
Approval_Request_History__c.Request_Comments__c	TRUE	TRUE
Approval_Request_History__c.SendReminders__c	TRUE	TRUE
Approval_Request_History__c.Send_Email__c	TRUE	TRUE
Approval_Request_History__c.SequenceString__c	FALSE	TRUE
Approval_Request_History__c.Sequence__c	TRUE	TRUE
Approval_Request_History__c.Status_Link__c	FALSE	TRUE
Approval_Request_History__c.StepLabel__c	TRUE	TRUE
Approval_Request_History__c.StepNameLink__c	FALSE	TRUE
Approval_Request_History__c.StepSequenceString__c	FALSE	TRUE
Approval_Request_History__c.StepSequence__c	TRUE	TRUE
Approval_Request_History__c.Step_Group_Seq_Number__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request_History__c.Step_Name__c	TRUE	TRUE
Approval_Request_History__c.Step__c	TRUE	TRUE
Approval_Request_History__c.SubmissionComment1__c	TRUE	TRUE
Approval_Request_History__c.SubmissionComment2__c	TRUE	TRUE
Approval_Request_History__c.SubmissionComment3__c	TRUE	TRUE
Approval_Request_History__c.SubprocessDependsOn__c	TRUE	TRUE
Approval_Request_History__c.SubprocessName__c	TRUE	TRUE
Approval_Request_History__c.SubprocessSequence__c	TRUE	TRUE
Approval_Request_History__c.SubstepDependsOn__c	TRUE	TRUE
Approval_Request_History__c.SubstepName__c	TRUE	TRUE
Approval_Request_History__c.SubstepSequence__c	TRUE	TRUE
Approval_Request_History__c.Workflow_Trigger_Added_Comments__c	TRUE	TRUE
Approval_Request__c.Action_Approve_Id__c	TRUE	TRUE
Approval_Request__c.Action_Prefix__c	FALSE	TRUE
Approval_Request__c.Action_Reassign_Id__c	TRUE	TRUE
Approval_Request__c.Action__c	FALSE	TRUE

Field	Editable	Readble
Approval_Request__c.Active__c	TRUE	TRUE
Approval_Request__c.ActualApproverName__c	FALSE	TRUE
Approval_Request__c.Actual_Approver__c	TRUE	TRUE
Approval_Request__c.AgreementLineItemId__c	TRUE	TRUE
Approval_Request__c.ApprovalCount__c	TRUE	TRUE
Approval_Request__c.ApprovalFromEmail__c	TRUE	TRUE
Approval_Request__c.ApprovalPercent__c	TRUE	TRUE
Approval_Request__c.ApprovalPolicy__c	TRUE	TRUE
Approval_Request__c.Approval_Process__c	TRUE	TRUE
Approval_Request__c.Approval_Status__c	TRUE	TRUE
Approval_Request__c.Approver_Comments__c	TRUE	TRUE
Approval_Request__c.Assigned_To_Id__c	TRUE	TRUE
Approval_Request__c.Assigned_To_Link__c	FALSE	TRUE
Approval_Request__c.Assigned_To_Name__c	TRUE	TRUE
Approval_Request__c.Assigned_To_Type__c	TRUE	TRUE
Approval_Request__c.Assigned_To__c	TRUE	TRUE
Approval_Request__c.AutoEscalate__c	TRUE	TRUE
Approval_Request__c.AutoReapprove__c	TRUE	TRUE
Approval_Request__c.Auto_Complete__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request__c.Backup_From_User__c	TRUE	TRUE
Approval_Request__c.CanEscalate__c	TRUE	TRUE
Approval_Request__c.CarbonCopyAssigneeIds__c	TRUE	TRUE
Approval_Request__c.ChildObjectId__c	TRUE	TRUE
Approval_Request__c.ChildObjectLink__c	FALSE	TRUE
Approval_Request__c.ChildObjectName__c	TRUE	TRUE
Approval_Request__c.ChildObjectRefId__c	TRUE	TRUE
Approval_Request__c.ChildObjectType__c	TRUE	TRUE
Approval_Request__c.ContinuePolicyApprovalOnAReject__c	TRUE	TRUE
Approval_Request__c.CriteriaFieldNames__c	TRUE	TRUE
Approval_Request__c.DateApproved__c	TRUE	TRUE
Approval_Request__c.DateAssigned__c	TRUE	TRUE
Approval_Request__c.DateCancelled__c	TRUE	TRUE
Approval_Request__c.DateEscalated__c	TRUE	TRUE
Approval_Request__c.DateReassigned__c	TRUE	TRUE
Approval_Request__c.DateRejected__c	TRUE	TRUE
Approval_Request__c.Date__c	TRUE	TRUE
Approval_Request__c.DelegateApproverIds__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request__c.DelegateApproverNames__c	TRUE	TRUE
Approval_Request__c.DelegateApprover__c	TRUE	TRUE
Approval_Request__c.DependsOn__c	TRUE	TRUE
Approval_Request__c.EscalateToChain__c	TRUE	TRUE
Approval_Request__c.EscalateToId__c	TRUE	TRUE
Approval_Request__c.EscalateToName__c	TRUE	TRUE
Approval_Request__c.EscalateToType__c	TRUE	TRUE
Approval_Request__c.EscalatedToHighestLevel__c	TRUE	TRUE
Approval_Request__c.ExpectedCompletionDate__c	TRUE	TRUE
Approval_Request__c.ExpectedDaysToComplete__c	TRUE	TRUE
Approval_Request__c.ExpectedHoursToComplete__c	TRUE	TRUE
Approval_Request__c.ExpectedMinutesToComplete__c	TRUE	TRUE
Approval_Request__c.Group_Unique_Id__c	TRUE	TRUE
Approval_Request__c.Group__c	TRUE	TRUE
Approval_Request__c.HasAttachments__c	TRUE	TRUE
Approval_Request__c.HasDelegateApprover__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request__c.InEscalation__c	TRUE	TRUE
Approval_Request__c.Initial_Submitter__c	TRUE	TRUE
Approval_Request__c.Internal_Comments__c	TRUE	TRUE
Approval_Request__c.IsAdhocProcess__c	TRUE	TRUE
Approval_Request__c.IsAdhoc__c	TRUE	TRUE
Approval_Request__c.IsAutoReapprovalEnabled__c	TRUE	TRUE
Approval_Request__c.IsSubprocess__c	TRUE	TRUE
Approval_Request__c.Notify_Only__c	TRUE	TRUE
Approval_Request__c.ObjectRefId__c	TRUE	TRUE
Approval_Request__c.Object_Id_Link__c	FALSE	TRUE
Approval_Request__c.Object_Id__c	TRUE	TRUE
Approval_Request__c.Object_Name__c	TRUE	TRUE
Approval_Request__c.Object_Type__c	TRUE	TRUE
Approval_Request__c.ParentRequestId__c	TRUE	TRUE
Approval_Request__c.Parent_Agreement__c	TRUE	TRUE
Approval_Request__c.PrevAssignedToId__c	TRUE	TRUE
Approval_Request__c.PrevAssignedToName__c	TRUE	TRUE
Approval_Request__c.PrevAssignedToType__c	TRUE	TRUE
Approval_Request__c.ProcessInstanceId__c	TRUE	TRUE

Field	Editable	Readble
Approval_Request__c.Rejection_Action__c	TRUE	TRUE
Approval_Request__c.Related_Agreement_Owner__c	TRUE	TRUE
Approval_Request__c.Related_Agreement_Requestor__c	TRUE	TRUE
Approval_Request__c.Related_Agreement_Term_Exception__c	TRUE	TRUE
Approval_Request__c.Related_Agreement__c	TRUE	TRUE
Approval_Request__c.Related_Opportunity_Owner__c	TRUE	TRUE
Approval_Request__c.Related_Opportunity__c	TRUE	TRUE
Approval_Request__c.RequestType__c	TRUE	TRUE
Approval_Request__c.Request_Comments__c	TRUE	TRUE
Approval_Request__c.SendReminders__c	TRUE	TRUE
Approval_Request__c.Send_Email__c	TRUE	TRUE
Approval_Request__c.SequenceString__c	FALSE	TRUE
Approval_Request__c.Sequence__c	TRUE	TRUE
Approval_Request__c.Status_Link__c	FALSE	TRUE
Approval_Request__c.StepLabel__c	TRUE	TRUE
Approval_Request__c.StepNameLink__c	FALSE	TRUE
Approval_Request__c.StepSequenceString__c	FALSE	TRUE

Field	Editable	Readble
Approval_Request__c.StepSequence__c	TRUE	TRUE
Approval_Request__c.Step_Group_Seq_Number__c	TRUE	TRUE
Approval_Request__c.Step_Name__c	TRUE	TRUE
Approval_Request__c.Step__c	TRUE	TRUE
Approval_Request__c.SubmissionComment1__c	TRUE	TRUE
Approval_Request__c.SubmissionComment2__c	TRUE	TRUE
Approval_Request__c.SubmissionComment3__c	TRUE	TRUE
Approval_Request__c.SubprocessDependsOn__c	TRUE	TRUE
Approval_Request__c.SubprocessName__c	TRUE	TRUE
Approval_Request__c.SubprocessSequence__c	TRUE	TRUE
Approval_Request__c.SubstepDependsOn__c	TRUE	TRUE
Approval_Request__c.SubstepName__c	TRUE	TRUE
Approval_Request__c.SubstepSequence__c	TRUE	TRUE
Approval_Request__c.Workflow_Trigger_Added_Comments__c	TRUE	TRUE
Apttus__APTS_Admin__c.Apttus__Code__c	FALSE	TRUE
Apttus__APTS_Admin__c.Apttus__Value__c	FALSE	TRUE
Apttus__APTS_Agreement__c.Approval_Status__c	TRUE	TRUE
Apttus__APTS_Agreement__c.LineItem_Approval_Status__c	TRUE	TRUE

Field	Editable	Readble
Apttus__APTS_Agreement__c.Term_Exception_Approval_Status__c	TRUE	TRUE
Apttus__Term_Exception__c.Apttus__Active__c	FALSE	TRUE
Apttus__Term_Exception__c.Apttus__Agreement_Types__c	FALSE	TRUE
Apttus__Term_Exception__c.Apttus__Approval_Required__c	FALSE	TRUE
Apttus__Term_Exception__c.Apttus__Description__c	FALSE	TRUE
Apttus__Term_Exception__c.Apttus__Exception_Type__c	FALSE	TRUE
Apttus__Term_Exception__c.Apttus__ReferenceId__c	FALSE	TRUE
AsyncActionInfo__c.AsyncActionJobId__c	TRUE	TRUE
AsyncActionInfo__c.EndTime__c	TRUE	TRUE
AsyncActionInfo__c.ErrorMsgInfo__c	TRUE	TRUE
AsyncActionInfo__c.ErrorStackInfo__c	TRUE	TRUE
AsyncActionInfo__c.ObjectName__c	TRUE	TRUE
AsyncActionInfo__c.StartTime__c	TRUE	TRUE
AsyncActionInfo__c.Status__c	TRUE	TRUE
AsyncActionInfo__c.SubmitTime__c	TRUE	TRUE
AsyncActionInfo__c.Submitter__c	TRUE	TRUE

Field	Editable	Readble
Backup_Approver__c.Backup_Action__c	FALSE	TRUE
Backup_Approver__c.Backup_User__c	TRUE	TRUE
Backup_Approver__c.Cancellation_Date__c	TRUE	TRUE
Backup_Approver__c.Comment__c	TRUE	TRUE
Backup_Approver__c.Current_User__c	TRUE	TRUE
Backup_Approver__c.DelegateUserIds__c	TRUE	TRUE
Backup_Approver__c.Effective_Date__c	TRUE	TRUE
Backup_Approver__c.Expiration_Date__c	TRUE	TRUE
Backup_Approver__c.InEffect__c	TRUE	TRUE
Backup_Approver__c.IsActive__c	TRUE	TRUE
Backup_Approver__c.IsDelegate__c	TRUE	TRUE
Backup_Approver__c.SuppressDelegateNotification__c	TRUE	TRUE
Backup_Approver__c.Transfer_in_flight__c	TRUE	TRUE
FormulaField__c.BusinessObject__c	FALSE	TRUE
FormulaField__c.Description__c	FALSE	TRUE
FormulaField__c.Formula__c	FALSE	TRUE
FormulaField__c.Type__c	FALSE	TRUE
Global_Discount_Policy__c.Approval_Level__c	FALSE	TRUE
Global_Discount_Policy__c.Authorized_Discount_Percent__c	FALSE	TRUE

Field	Editable	Readble
Global_Discount_Policy__c.Description__c	FALSE	TRUE
Opportunity.Approval_Status__c	TRUE	TRUE
ReminderInstance__c.ApprovalRequestId__c	TRUE	TRUE
ReminderInstance__c.Completed__c	TRUE	TRUE
ReminderInstance__c.Label__c	TRUE	TRUE
ReminderInstance__c.ProcessStepId__c	TRUE	TRUE
ReminderInstance__c.ScheduledFireTime__c	TRUE	TRUE
ReminderInstance__c.Status__c	TRUE	TRUE
ReminderInstance__c.StepSequence__c	TRUE	TRUE
ReminderInstance__c.StepType__c	TRUE	TRUE
ReminderInstance__c.SubprocessRuleEntryAssigneeId__c	TRUE	TRUE
ReminderInstance__c.SubprocessRuleEntryAssigneeSeq__c	TRUE	TRUE
Reminder__c.DurationUnit__c	FALSE	TRUE
Reminder__c.DurationValue__c	FALSE	TRUE
Reminder__c.Label__c	FALSE	TRUE
Reminder__c.ProcesssStepId__c	FALSE	TRUE
Reminder__c.StepSequence__c	FALSE	TRUE
Reminder__c.StepType__c	FALSE	TRUE

Field	Editable	Readble
Reminder__c.SubprocessRuleEntryAssigneeId__c	FALSE	TRUE
Reminder__c.SubprocessRuleEntryAssigneeSeq__c	FALSE	TRUE
Reminder__c.SubprocessRuleEntryId__c	FALSE	TRUE
SearchFilter__c.BusinessObject__c	FALSE	TRUE
SearchFilter__c.CriteriaFieldNames__c	FALSE	TRUE
SearchFilter__c.Criteria__c	FALSE	TRUE
SearchFilter__c.Description__c	FALSE	TRUE
SearchFilter__c.Sequence__c	FALSE	TRUE
SearchFilter__c.UseType__c	FALSE	TRUE
Term_Exception_Approval_Condition__c.Active__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Description__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Field__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Join__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Number__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Object_Qualifier__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Object__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Operator__c	TRUE	TRUE

Field	Editable	Readble
Term_Exception_Approval_Condition__c.TermExceptionApprovalId__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Term_Exception_Approval__c	TRUE	TRUE
Term_Exception_Approval_Condition__c.Value__c	TRUE	TRUE
Term_Exception_Approval__c.Active__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_Description__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_Id__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_Type__c	TRUE	TRUE
Term_Exception_Approval__c.Assignee_User__c	TRUE	TRUE
Term_Exception_Approval__c.AutoEscalate__c	TRUE	TRUE
Term_Exception_Approval__c.DependsOn__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToChain__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToId__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToName__c	TRUE	TRUE
Term_Exception_Approval__c.EscalateToType__c	TRUE	TRUE
Term_Exception_Approval__c.ExpectedDaysToComplete__c	TRUE	TRUE

Field	Editable	Readble
Term_Exception_Approval__c.ExpectedHoursToComplete__c	TRUE	TRUE
Term_Exception_Approval__c.ExpectedMinutesToComplete__c	TRUE	TRUE
Term_Exception_Approval__c.FilterLogic__c	TRUE	TRUE
Term_Exception_Approval__c.Notify_Only__c	TRUE	TRUE
Term_Exception_Approval__c.Send_Email__c	TRUE	TRUE
Term_Exception_Approval__c.Sequence__c	TRUE	TRUE
Term_Exception_Approval__c.SkipUnresolvedAssignee__c	TRUE	TRUE

For information on field permissions, refer to this Salesforce article.

Object Permissions for Users

The following table displays the permissions required by users for Approval Objects:

Object	allowCreate	allowDelete	allowEdit	allowRead	modifyAllRecords	viewAllRecords
AdhocApprovalGroup__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
AdhocApprovalProcess__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
AdhocApprover__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ApprovalJob__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ApprovalProcessInstance__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

Object	allowCreate	allowDelete	allowEdit	allowRead	modifyAllRecords	viewAllRecords
ApprovalRuleAssignee__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
ApprovalRuleDimension__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
ApprovalRuleEntry__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
ApprovalRule__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Approval_Matrix__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Approval_Process__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Approval_Request_History__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Approval_Request__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Apttus__APTS_Admin__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Apttus__APTS_Agreement__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Apttus__Term_Exception__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
AsyncActionInfo__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Backup_Approver__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

Object	allowCreate	allowDelete	allowEdit	allowRead	modifyAllRecords	viewAllRecords
FormulaField__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Global_Discount_Policy__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Opportunity	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
ReminderInstance__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Reminder__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
SearchFilter__c	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Term_Exception_Approval_Condition__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Term_Exception_Approval__c	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

For information on setting object permissions, refer to this Salesforce article.

Page Access Permissions for Users

The following table displays the permissions required by users for Approval Pages:

Apex Page	Enabled
AdhocProcessRuntime	TRUE
AgreementApprovalSummary	TRUE
AgreementApprovalSummarySf1	TRUE
AgreementLineItemApprovalSummary	TRUE
AgreementTermExAdd	TRUE

Apex Page	Enabled
AgreementTermExApprovalSummary	TRUE
AgreementTermExApprovals	TRUE
AgreementTermExCancel	TRUE
AgreementTermExEdit	TRUE
AgreementTermExHistory	TRUE
AgreementTermExPreview	TRUE
AgreementTermExSubmit	TRUE
ApprovalAction	TRUE
ApprovalActionSf1	TRUE
ApprovalContextCancel	TRUE
ApprovalContextPreview	TRUE
ApprovalContextSubmit	TRUE
ApprovalMatrixGenerate	TRUE
ApprovalMatrixView	TRUE
ApprovalProcessDefn	TRUE
ApprovalProcessDefnList	TRUE
ApprovalProcessFinalAction	TRUE
ApprovalProcessInitialAction	TRUE
ApprovalProcessSelectEmailTemplate	TRUE

Apex Page	Enabled
ApprovalProcessStep	TRUE
ApprovalProcessStepView	TRUE
ApprovalRequestAttachment	TRUE
ApprovalRequestEscalate	TRUE
ApprovalRequestReassign2	TRUE
ApprovalRequestReassign2Sf1	TRUE
ApprovalRequestsList	TRUE
ApprovalRequestsListSf1	TRUE
ApprovalRule	TRUE
ApprovalRuleCriteriaEdit	TRUE
ApprovalRuleDimension	TRUE
ApprovalRuleEntry	TRUE
ApprovalSummaryLaunch	TRUE
ApprovalSummaryLaunchAddCommentSf1	TRUE
ApprovalSummaryLaunchApproveRejectSf1	TRUE
ApprovalSummaryLaunchReassignSf1	TRUE
ApprovalsJSLibInclude	TRUE
BackupApprover	TRUE
BackupApproverAction	TRUE
BackupUserSearch	TRUE

Apex Page	Enabled
DelegateApprovers	TRUE
FormulaField	TRUE
MatrixUserSearch	TRUE
MyApprovals	TRUE
MyApprovalsLaunch	TRUE
MyApprovalsSf1	TRUE
NextLevelApprover	TRUE
OpportunityApprovalSummary	TRUE
OpportunityPartnersList	TRUE
OpportunityProductsList	TRUE
PreviewSubmitApprovals	TRUE
QueueSearch	TRUE
ReassignUserSearch	TRUE
ReassignUserSearchMyApprovals	TRUE
SObjectApprovals2Submit	TRUE
SObjectApprovals2SubmitSf1	TRUE
SearchFilter	TRUE
SearchFilterDetail	TRUE
TermExApproval	TRUE

Apex Page	Enabled
TermExApprovalClone	TRUE

For information on setting page access permissions, refer to this Salesforce article.

Record Type Visibility Permissions for Users

The following table displays the permissions required by users for Approval Record Types:

Record Type Visibility	Enabled
Approval_Process__c.ApprovalProcess	TRUE
Approval_Process__c.EntryCriteria	TRUE
Approval_Process__c.FinalAction	TRUE
Approval_Process__c.InitialSubmissionAction	TRUE
Approval_Process__c.Instance	TRUE
Approval_Process__c.Step	TRUE
Approval_Process__c.StepFilterCriteria	TRUE
Approval_Process__c.StepGroup	TRUE
Approval_Process__c.StepGroupAction	TRUE

For information on setting record type visibilities, refer to this Salesforce article.

Tab Visibility Permissions for Users

The following table displays the permissions required by users for Tab Settings:

Tab Setting	Visibility
Approval_Request__c	Visible
Apttus__APTS_Agreement__c	Visible

Tab Setting	Visibility
Apttus__Term_Exception__c	Visible
Backup_Approver__c	Visible
Term_Exception_Approval__c	Visible

For information on tab settings, refer to this Salesforce article.

Asynchronous Processing of Approvals Records

You can also submit and process approval records asynchronously. For a large number of approval records, you can submit a batch job to process them in the background instead of waiting for the job to finish.

Asynchronous processing is available for:

- Submitting Approval Records
- Approving or Rejecting Approval Records

You can define a threshold to process the number of records in a synchronous mode. If the number of records to process exceeds the threshold value, the processing is done in an asynchronous manner. You can define the threshold in **Sync Approve Request Threshold** field in Approval System Properties.

Synchronous vs Asynchronous Processing of Approval Records

When you submit the approvals in a synchronous mode, you need to wait till all the records are submitted. The screen is not available for use while the processing happens and the screen keeps on auto-refreshing to reflect the status of the number of records processed.

In an asynchronous mode, a batch job is submitted in the background to process a large number of records. The screen is available for you to work on other records. You will receive an email once all the records are submitted or processed.

Configuring CPQ Approval Workflow

This section provides the workflow and the sequence of actions you have to define to enable CPQ Approvals.

To do...	How to define...
Install required packages	Install the latest approvals and dependent packages. Install the Quote or Proposal approvals package to sync the Approval Status and Approval Stage on the Quote header. However, if you are using CPQ approvals, you should not have any approval process defined on Quote/ Proposal object.
Approval Status Fields	Define *Approval Required* and *Pending Approval* pick list values.
Custom Settings	Create a record in Approval Custom Config for the primary object **ProductConfiguration__c** . Configure Cart Approval Callback class - **Apttus_CQApprov.CartApprovalCallback** in Config Custom Class for CPQ Approvals. If you want to use Header and line item approvals, set the approval context type to Multiple. Specify the label name for Line Item in the Child Object Context Type 2 field. For header only approvals, set the context type to Single.
Approvals System Properties	Create a record in Approval System Properties and enter the requisite details
Config System Properties	Navigate to Config System Properties, and select the **Skip Review** check box.

To do...	How to define...
Add Intelligent Approval Field for a Quote	Navigate to **Create > Objects > Quote > Configure Products(Field) >** copy the formula field and create a new formula field. Add the **useAdvancedApproval=true** parameter to the copied formula . For example, IF (LEN(Apttus_QPConfig__PriceListId__c) > 0 , HYPERLINK("/apex/Apttus_QPConfig__ProposalConfiguration?id=" &Id&"&**useAdvancedApproval=true** &useDealOptimizer=true", IMAGE("/resource/Apttus_QPConfig__Button_Configure", "Configure Products"),"_self"), NULL)
Approval Required Check	Create a search filter criteria for Approval Required Check for the header and line item objects (Apttus_Proposal__Proposal__c, Apttus_Config2__LineItem__c, Apttus_Config2__ProductConfiguration__c). These search filters are used for setting the approval status to Approval Required on the cart and cart line item object respectively. Approval Required Check trigger is default for CPQ Approvals.
Formula Fields (Approvals)	To use reference parameters from related objects, create Formula Fields (Approvals).
Approvals and My Approval Pages	Drag and drop the Approvals and My Approvals button to the page layout to execute approval actions from the Quote header page rather than from the shopping cart.
Email Notification Templates	Define Email Notification templates using sample templates from the Conga *Approvals Management* package.
Setting up Approval Rules	Create Approval Rules for each Sub Process and Child Process if you have multiple approvers for a process. Configure Approval Rule criteria. Select **Approval Rule Type** as *Condition* to use Auto Re-approval conditions. For each step in sub-process or child process, create approval rule entry, step label, Approval Condition, Auto Re-approval Condition, and Assignee, and step dependency.

To do...	How to define...
Approval Process Configuration	Create Approval Process. Configure process properties. Configure Initial and Final Submission Actions. To use Auto Re-approvals, set the cart status to **Ready for Finalization** in the final submission actions rather than Finalized to ensure that a new cart is not created every time.
Approval Process Step Configuration	Configure approval process step, step dependencies, display fields, comments labels. For each Standard Step also define Approval Condition, Auto Re-approval Condition, and Assignee. For each Sub Process or Child Process Steps and map them to corresponding approval rules. For Standard Steps that have approval conditions based on child objects (e.g. line items) define Search Filter (Approvals) of type Child Filter and use them in the step definition.

Configuring a Custom Approval Action for Non-Salesforce or Conga profiles	If you create a custom object to replace the default Approval Action, you must be sure to code the object correctly to handle`$User.id`
Setting up the Backup Administrator	The backup admin user is responsible for handling routing issues that may occur during the approval process, while admin profiles enable you to associate users – via their roles – with admin level access to Approvals.
Setting up Backup or Delegate Approvers	Backup approvers are required to 'catch' approval requests whose assignees cannot be identified and ensure approval processes progress as expected.
Setting up Email Templates and Alerts	While Conga provides default email templates for approval requests, if you want the emails to have attachments or have your own specific content, you must customize the templates.
Email Approval System Properties Settings	To enable approval requests to be decided via email, an email service and Approval System Properties settings for emails must be configured.

Setting up Custom Expressions	To help in creating advanced expressions for routing your approvals, you need to set up custom expressions.

Mandatory Configuration

This section lists the mandatory settings to setup Approvals for your objects.

- Approval System Properties
- Configuring Global Settings
- Setting up the Admin Profiles
- Configuring Email Templates
- Configuring Status Picklists
- Intelligent Approval Flag

Approval Process for CPQ

Conga Approvals offers the ability to setup approval processes that are currently not supported by Salesforce. It allows the customer to configure their various approval processes with greater flexibility and visibility, helping in overall compliance.

Approval steps assign approval requests to various users and define the chain of approval for a particular approval process. Each approval step specifies the attributes a record must have to advance to that approval step, the user who can approve requests for those records, and whether to allow backup assignees of the approver to approve the requests. Regardless of their complexity, when anyone in the process rejects an item, the entire process is considered rejected.

- Creating the Process Definition
- Checking the Approval Status of the Document
- Creating Initial Submission Actions
- Creating Final Actions
- Creating Approval Steps
- Setting up Auto-escalation
- Creating Expressions
- Optional Configuration

Performance and Scalability

Area	Detail
Multiple Approval Processes	**When to use multiple approval process?** When you have Region Based Approval variations, Business Unit Based Approvals variations, Channel Based Approvals variations. **How do I configure multiple approval processes?** Define mutually exclusive process entry criteria so that only one process is selected for a given primary object instance. **Why do I configure multiple approval processes?** Having multiple approval processes reduces number of approval steps / criteria system needs to execute
Approval Rule Entry Criteria	Specify minimum approval conditions in Approval Rule entry criteria for Sub Process or Child Process steps to avoid evaluation of all approval rule entries every time.
Approval Required Check	Provides users visibility to whether approval is needed for header or line items without executing the entire approval process. Provides a mechanism to avoid approval process evaluation for a large number of scenarios.

Types of Approval Steps and when do I use them?

- **Standard** - This is the existing method for creating header level approval steps.
- **Sub Process** - This is used when you want to select approval rules that are associated with the header level context object of the approval process. The assignee types are derived from the Approval Rule. Create a Sub-Process when you have multiple approvers for a set of criteria instead of defining multiple Standard steps for each criteria.
- **Child Process** - This is used when you want to select approval rules that are associated to child objects of the approval process header level context object such as you want to define an approval process for Proposal and Proposal Line Items. The assignee types are derived from the Approval Rule. Use a child process instead of a Subprocess and Standard step type when you want to populate the Approval request with the child object line items to be approved, such as proposal line items.

It is recommended that you use conditional criteria instead of Approval Dimensions and Approval Matrix while creating your approval rules and processes.

Configuring Approvals for Agreement and Related Objects

Configuration Overview

Approvals Management supports the Approvals and My Approvals pages out of the box for any object. With minimal configuration, you can set up both of these pages for any object and its child objects on which approvals is configured. This new capability overcomes the need for create custom Visualforce pages and Controllers for enabling approvals on any object, thus significantly simplifying the approval setup on any object and its related child objects.

Although these pages are now available out of the box, you have to configure Approval Required Check**.** You can get the Approval Required Check with the main Approvals Management package.

To create an approval process for Agreement header fields, Agreement Line Items, Agreement Term Exceptions, and Agreement Clauses, consider the following use cases for each of the Agreement objects.

For each of the following scenarios, you can view the Approval Requests, Approval Status and the changes in the header field or agreement clause for each approver on the My Approvals and Approvals home page.

1. Agreement header fields: You can set up an approval process and subsequent steps on the Agreement object to initiate an approval request for various scenarios, such as the total agreement amount > 10000, or Agreement type is MSA, an approval request is sent to the CFO of your company. Create an approval process for the Agreement header fields.

You can configure a child process for the following objects.

- **Agreement Clauses**: You can also initiate an approval request when any change is made to the clause of an agreement document. This is required when you want to ensure that all the changes which your customer makes to the existing clauses are approved before finalizing an agreement. It might be possible that your customer changes the Agreement Start Date and your CFO wants to approve the change before finalizing and signing the agreement with your customer.

For example, the agreement document contains a Warranty clause for 6 months and your customer wants to extend the warranty period to 1 year. Once your customer changes the period in the Warranty clause, you can set Approval process for the change in the Agreement Clause record.

An Approval with Agreement Clause is useful when you want to track addition of a new clause, modification of an existing clause or deletion of an existing clause from your agreement document. This can be done by tracking the Number of Clauses field of the Template object.

- **Agreement Line Items**: With the help of Approvals and Contract Management, you can also create approval processes to track any changes made to the Agreement Line Items and accordingly initiate an approval request.

  For example, when your customer changes the price of a product listed in your agreement document, you might require your manager to approve the change before the agreement is finalized.

  Also, if there are addition or deletion of Line Items in your agreement document, and you want the higher authority to approve or reject the changes, Approvals can be used.

- **Agreement Term Exceptions**: It is also possible to initiate an approval request when a clause, associate with a term exception is checked in from X-Author Contracts.

  An Approval with Term Exception is useful when your customer is requesting a change in an existing clause which requires approval from the CFO of your company, and this clause is specific to a single customer.
  Considering the above example, if you renew an agreement for a specific customer and your customer wants to enter a new clause. In this scenario, you want your Legal team to approve the new clause before adding it to the renewed agreement. So, you add the new clause, associate it with a term exception and define an approval step of type child process.

After the Approvals package has been installed, there are *Setup* configuration options that can be changed. By default, the package installation sets up the majority of the application to be used as is; however, some organizational specific options can only be configured after the installation. Some post-package installation configuration must be done to best maximize the features available in Approvals. There are also default settings that you may want to change, to better suit your business.

This section provides the workflow and the sequence of actions you have to define to enable Custom Object Approvals for Agreement and its child objects.

To do...	How to define...
Install required packages	Install the latest approvals and dependent packages.
Adding Values to Approval picklists	Add object names to processes, rules, and dimensions.
Approvals System Properties	Create a record in **Approval System Properties** and enter the requisite details
To configure custom object settings	Create a record in **Approval Custom Config** for the primary custom object. Set up the **Context Type** as follows: • *Multiple*: If you are defining an approval process for the header and child object. • *Single*: If you are using approvals only on one object. In our example, enter *Multiple* because we want to setup the approval process on both, Agreement header fields and Agreement Line Items.
Add pick list values for approval fields	• Create Approval Status and Approval Preview Status fields on the header object. • Create Approval Status and Approval Preview Status fields on the child object.

To do...	How to define...

Configure Approval Required Check Settings	• Setup Search Filter (Entry Criteria) for header (e.g. Agreement).
	• Setup Search Filter (Entry Criteria) for line items (e.g. Agreement Line Items).
	• Create the Approval Required Check Trigger for the header and line item object.
	• Invoke the Approval Required Check API within the trigger for the header and line item object and click **Save**.
	• Set Header and line item approval status based on the Approval Required Check API.
	• Pass the line items that need to be evaluated for approvals when the line items and header information is created or updated.
	The 2 APIs you have to invoke are from the ApprovalRequiredCheck global class:
	  • doCheck (Id objId) – returns true if successful, otherwise false.
	  This API is Invoked as follows:
	  ApprovalRequiredCheck checker = new ApprovalRequiredCheck()
	  Checker.doCheck(objId)
	  • doCheck(SObject object) – return a map of approval-status-values keyed by subject IDs. The first entry in the map corresponds to the header context object.
	  This API is Invoked in a trigger before update as follows:
	  ApprovalRequiredCheck checker = new ApprovalRequiredCheck()
	  Checker.doCheck(object)
	Approval status at the line item as well as header should be set to *Approval Required* for **Preview** or **Submit** to work. Use the above API to create a trigger to invoke Approval Required Check.

To do...	How to define...
Configuring Approvals and My Approvals Page	You can now enable Approvals for your context object by configuring the following 2 buttons and placing them on your object record's page layout. • **Preview & Submit Approvals** - For actions such as **Preview**, **Submit**, **Recall for Submitter**. • **My Approvals** - For actions such as **Approve**, **Reject**, **Add Ad hoc Approver**, **Reassign**.
Setting up Email Templates and Alerts	While provides default email templates for approval requests, if you want the emails to have attachments or have your own specific content, you must customize the templates.
Set up Approval Process	Configure Approval Process • Use the Approval Required status as process entry criteria. • Create additional pick list values for Approval Status etc. • Setup approval rules on the child context object to be used for the step of type Child Process. • Setup approval rules on the primary context object to be used for the step of type Subprocess. • Approval rules of type Condition can also be defined if the approval rule entry criteria is complex expression. • Setup the initial and final submission actions.
Setting up Approval Rules	Create Approval Rules for each Sub Process and Child Process if you have multiple approvers for a process. Configure Approval Rule criteria. Select Approval Rule Type as Condition to use Auto Re-approval conditions. For each step in sub-process or child process, create approval rule entry, step label, Approval Condition, Auto Re-approval Condition, and Assignee, and step dependency.
Setting up Backup or Delegate Approvers	Backup approvers are required to 'catch' approval requests whose assignees cannot be identified and ensure approval processes progress as expected.

To do...	How to define...
Setting up the Backup Administrator	The backup admin user is responsible for handling routing issues that may occur during the approval process, while admin profiles enable you to associate users – via their roles – with admin level access to Approvals.
Set up Page Layout for Primary Object	Add Approvals and My Approvals buttons to primary object layout.

Optional Configuration

Auto-escalating Approval Requests	When an approval request is not approved, rejected, or reassigned within the allotted time, it can automatically be reassigned to a new approver so the approval process can continue.
Changing Approval Rule Step Behavior	Typically steps that cannot identify an assignee are sent to the backup administrator; however, you can change that behavior and have the those steps skipped instead.
Changing the Assignee Search page results list	This controls the number of assignees that can be displayed on a single search page. Typically the default value of twenty is suitable for your organization, but you can increase or decrease the value
Creating Approval Matrices	Matrices are used with *Global Discount Policies* to route approvals to users based on system wide default discount percentages.
Approval Request History	When an approval process is initiated – by submitting or resubmitting an approval – for an agreement, term exception, or opportunity, an approval history record is created. The Approval request history Object comprises information for old approval requests such as, the approval status, the date of the request, who it was assigned to, and who the actual approver was.

Submitting a Request with Attachments	Enables you to submit your approval request with an attachment. To submit a request with attachment, ensure that you have created custom email notification templates and specified the name of each of the templates in the Approval Process of the object.
Approval Submission Comments	Approval submission comments enable you to add personalized comments when you submit an approval request. You can have a single comment at the process level or up to three comments at the step level.
Child Filter Objects	From 7.0 onwards you can use a child process to configure approvals on child objects. However if you wish to create approval steps based on child object fields, you can use custom child object fields. Create an child filter expressions to filter the records based on child object fields.
Reporting on Approval Request Channels	As approval requests can be completed within Salesforce or outside of the system via email, this feature provides a way to identify which of the two is used for each approval request.
Opportunity records and ACM	When Advanced currency management (ACM) is enabled in your org, you cannot have any Opportunity record fields of data type currency in your email templates or approval summary page.

Best Practices

This section comprises the best practices to be considered while defining your approval processes.

Consolidate Notification?	Consolidated Approvals?	Behavior
Yes	No	Requests go on-hold if there are additional requests pending for approval. Once the last request becomes available for approval, all the requests for the assignee get assigned. A user gets only one email notification for approval.
Yes	Yes	Requests go On-Hold if there are more requests pending for approval. Once the last request becomes available for approval, only one of the requests for the assignee is assigned. All other requests remain On-Hold. What ever action user takes on an assigned request is propagated to all other requests. A user gets only one email notification for approval.
No	No	Requests never go On-Hold. Requests are assigned to the assignee as and when requests becomes available. A user gets a separate assignment notification for each request.
No	Yes	Not Supported.

Performance and Scalability

Area	Detail
Multiple Approval Processes	When to use multiple approval process? When you have Region Based Approval variations, Business Unit Based Approvals variations, Channel Based Approvals variations. How do I configure multiple approval processes? Define mutually exclusive process entry criteria so that only one process is selected for a given primary object instance. Why do I configure multiple approval processes? Having multiple approval processes reduces number of approval steps/criteria system needs to execute
Approval Rule Entry Criteria	Specify minimum approval conditions in Approval Rule entry criteria for Sub Process or Child Process steps to avoid the evaluation of all approval rule entries every time.
Approval Required Check	Provides users visibility to whether approval is needed for header or line items without executing the entire approval process. Provides a mechanism to avoid approval process evaluation for a large number of scenarios.

Types of Approval Steps and when do I use them?

- **Standard** - This is the existing method for creating header level approval steps.
- **Sub Process** - This is used when you want to select approval rules that are associated with the header level context object of the approval process. The assignee types are derived from the Approval Rule. Create a Sub-Process when you have multiple approvers for a set of criteria instead of defining multiple Standard steps for each criteria.
- **Child Process** - This is used when you want to select approval rules that are associated to child objects of the approval process header level context object such as you want to define an approval process for Proposal and Proposal Line Items. The assignee types are derived from the Approval Rule. Use a child process instead of a Subprocess and Standard step type when you want to populate the Approval request with the child object line items to be approved, such as Proposal Line Items.

  Note

> If you select *Rule* as an **Assignee Type**, the approval request is sent only to the first user that satisfies the filter criteria in the Approval Rule. If you have multiple approvers in a sequence, only the first approver is selected and assigned.

It is recommended that you use conditional criteria instead of Approval Dimensions and Approval Matrix while creating your approval rules and processes.

Mandatory Setting for Agreement Objects

This section outlines the mandatory settings to setup Approvals for your objects.

- Approval System Properties
- Global Settings for Custom Objects and its Related Objects
- Setting up the Admin Profiles
- Configuring Email Templates
- Setting up Intelligent Approval Flag

Configuring Email Templates for Agreement and its related Objects

Typically the easiest way to create your own custom template is to clone one of the email templates provided in the unmanaged Conga samples package and then edit it. After that has been done you can then override the default email templates for all of your approval processes using APTS_ApprovalConfig or on an ad-hoc basis for specific approval requests.

The Conga Approvals (Customization Samples) package provides email template examples for currently supported objects, as well as for Salesforce Cases, which can subsequently be used as the basis for creating email templates for custom objects you've configured yourself. Also note that Opportunity Product is also used to provide templates for further customization.

You have to specify the email templates in the Approval Process and the Approvals Custom Config settings.

As well as providing samples for templates and the summary page, the pages that are used to provide approval functionality on the custom object record page are also included.

Note

As this package is unmanaged, it cannot be upgraded. If you want to move to the newest version, you must uninstall the existing package and then install the new package. It is recommended you uninstall the samples package after you have created your custom templates and pages.

The Conga Approvals package contains email templates, components, Visual force pages and Apex controllers which can help you build you templates and approval summary pages for custom context objects.

Note

The samples contained in the package should not be used 'as is' and are only to be used in the development of your own bespoke templates and pages, and should not be installed in production.

Configuring Email Templates

For a process with header and child process steps, use the Conga Approval Email Templates. Notification templates are used to create the notifications sent to approvers to alert them to pending approvals and to allow them to approve or decline a request without logging in to Salesforce. The following approval email notification templates are available as a part of the standard Approvals Management package. These are generic email notification templates that can be used for approvals on any standard or custom objects without any change. However, these notification templates can be customized as needed. Clone these templates if you want to customize your own template.

Sample	Component Type	Description
Conga My Approvals Notification (Assignment)	Visualforce Email Template	This template is used to notify users they must decide on an approval request.
Conga My Approvals Notification (Reassignment)	Visualforce Email Template	This template is used to notify users that an approval request task initially assigned to them has been reassigned to someone else.

Sample	Component Type	Description
Conga My Approvals Notification (Escalation)	Visualforce Email Template	This template is used for auto-escalation. Configuration is required to activate this feature.
Conga My Approvals Notification (Cancellation)	Visualforce Email Template	This template is used to notify users that an approval request has been cancelled.
Conga My Approvals Notification (Notify Only)	Visualforce Email Template	This template is used to notify users only.

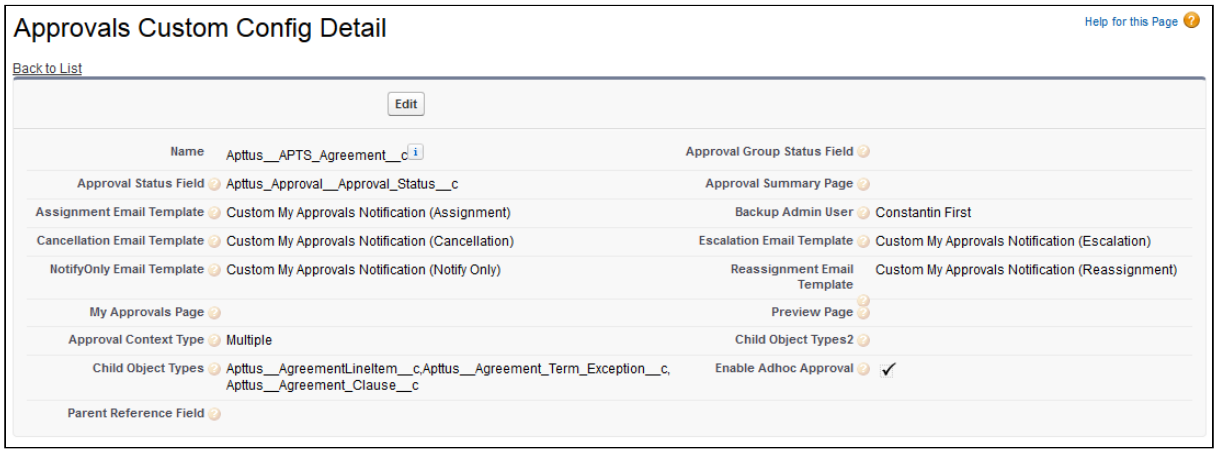
Configuring Global Settings for Agreement and its related Objects

Approvals custom configuration settings provide the global default values for using Approvals with different objects. The backup admin user, summary page, and email templates are designated here so that they do not need to be selected each time you create an approval process. You can choose to override them for specific processes, but this ensures approval requests will work correctly regardless.

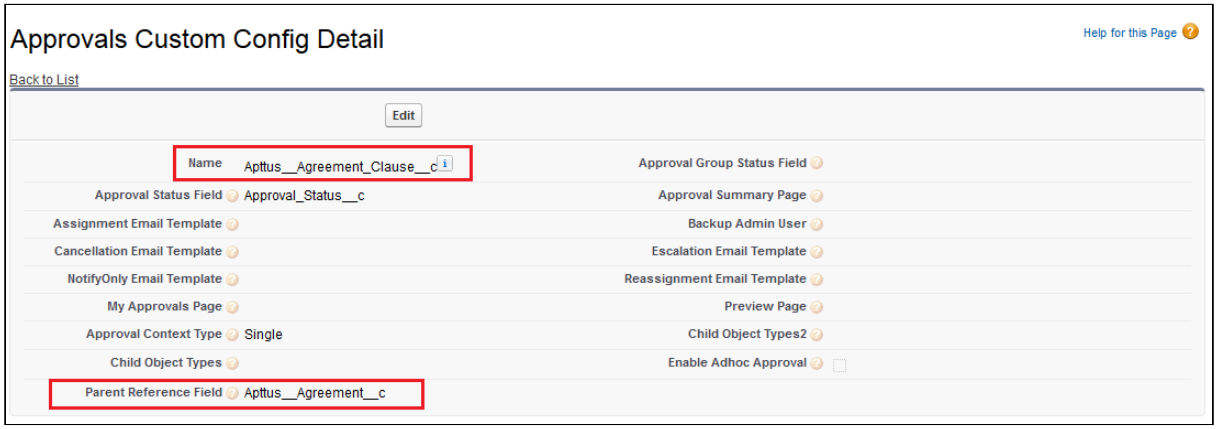
To configure global settings

You must have created the required custom pages and templates.

1. Go to **Build > Develop > Custom Settings** and click **Approvals Custom Config**.
2. From the **Custom Setting Definition Detail** section, click **Manage** to display the current objects associated with Approvals.
3. If it does not already exist, click **New** and enter Apttus__APTS_Agreement__c in the **Name** field, which is the API name for the agreement object to be used with approvals.
4. In the **Approval Status Field**, enter Apttus_Approval__Approval_Status__c or whatever the API name is for the picklist created for the custom object.
5. The custom summary page and templates you created should be entered in the **Approval Summary Page** and various **Email Template** fields. If you do not enter values, the default system values will be used.



6. Specify the **Approval Context Type** as *Single* or *Multiple.* If you want to use Header and line item approvals, set the **Approval Context Type** to *Multiple*. For header only approvals, set the **Approval Context Type** to *Single.* In our example, enter *Multiple* because we want to setup approval process on agreement header and line item level.
7. Specify the Child Object Type. The Child object type is associated with the approval process for a parent object. In our example, enter the API name of the child objects on which you want to setup approval process, such as Agreement Clause and Agreement Line Item.
8. You can enable ad hoc approvals for a custom object by selecting **Enable Ad Hoc Approval**.
9. In the **Parent Reference** field, enter the API name of the object with which you have either a master-detail or a look-up relationship. For example, if for the Agreement Term Exception record of **Approvals Custom Config**, if you want to create approval on Agreement object and it has a master detail relationship with Agreement Term Exception, so enter *Apttus__Agreement__c* in the **Parent Reference Field**.



10. Click **Save**.

The page and templates will be used by default for the next approval process created for the custom context object. When creating an approval process, you can override these templates if necessary.

Adding Values to the Agreement Picklist

When creating an approval process, to be able to associate it and its various components with a custom object, you must have the custom object available from approval picklists.

Any part of the approval process creation that includes linking approvals to a context business object, needs to have the custom object added to it. This includes the approval process, approval rule dimension, approval rule, and search filter.

To add a new Object to Approval Process picklist

1. Go to **Build > Create > Objects > Approval Process** and click **Object Label**.
2. In the **Picklist Values** section, click **New**.
3. Enter `Object Name` in the text field and then select **Approval Process**.
4. Click **Save**.

Object is added to the Picklist Values list for the Approval Process.

The custom object API name must be added to the pick list.

To add API name to Approval Process picklist

1. Go to **Build > Create > Objects > Approval Process** and click **Object Type**.
2. In the **Picklist Values** section, click **New**.
3. Enter `Object API name` in the text field and then select **Approval Process**.
4. Click **Save**.

The Object is added to the Picklist values and can now be associated with an approval process during the creation process.

To add API name to Approval Rule Dimension picklist

1. Go to **Build > Create > Objects > Approval Rule Dimension**.
2. In the **Custom Fields & Relationships** section, click **Business Object**.
3. In the **Picklist Values** section, click **New**.
4. Enter `Object API Name` in the text field.

5. Click **Save**.

The Object can now be associated with dimensions during the creation process

To add API name to Approval Rule picklist

1. Go to **Build > Create > Objects > Approval Rule**.
2. In the **Custom Fields & Relationships** section, click **Business Object**.
3. In the **Picklist Values** section, click **New**.
4. Enter `Object API name` in the text field.
5. Click **Save**.

The Object can now be associated with approval rules.

To add API name to Search Filter picklist

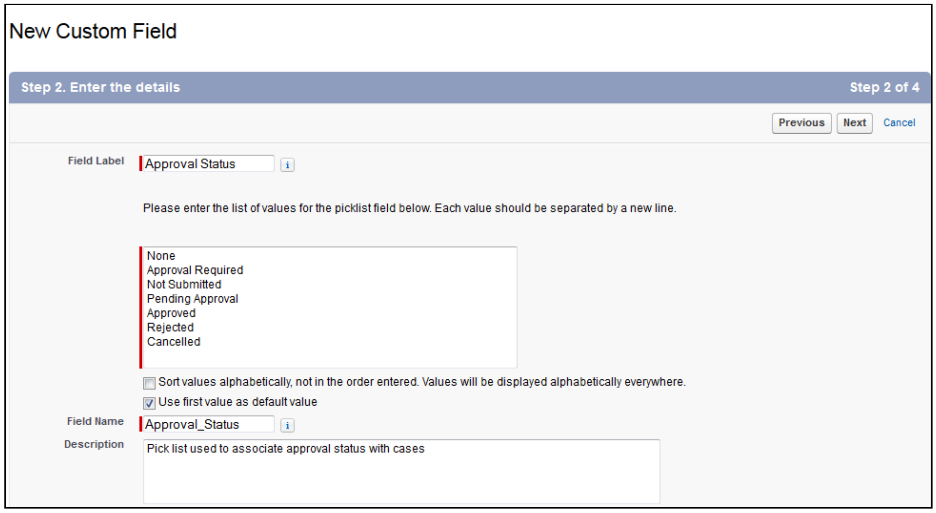
1. Go to **Build > Create > Objects > Search Filter**. In instances when there are multiple options, be sure to select the *Search Filter* associated with Conga *Approvals Management*.
2. In the **Custom Fields & Relationships** section, click **Business Object**.
3. In the **Picklist Values** section, click **New**.
4. Enter `Object API Name` in the text field.
5. Click **Save**.

Adding Approval Status to Agreement and its related Objects

After the picklists have been set for objects, it's time to set the object picklist field so that Approval Status is available on the object record.

To add a picklist field for approvals to objects

1. Go to **Build > Customize > Agreement** or its related objects, such as **Agreement Clause** and **Agreement Line Item**.
2. In the **Custom Fields & Relationships** section, click **New**.
3. Select **Picklist** data type and click **Next.**
4. On the Details page, enter the following to ensure approval statuses will be available for the custom object and click **Next**.



5. Select which users will be able to see the picklist and click **Next**.
6. Select which page layouts will include the picklist and click **Save**.

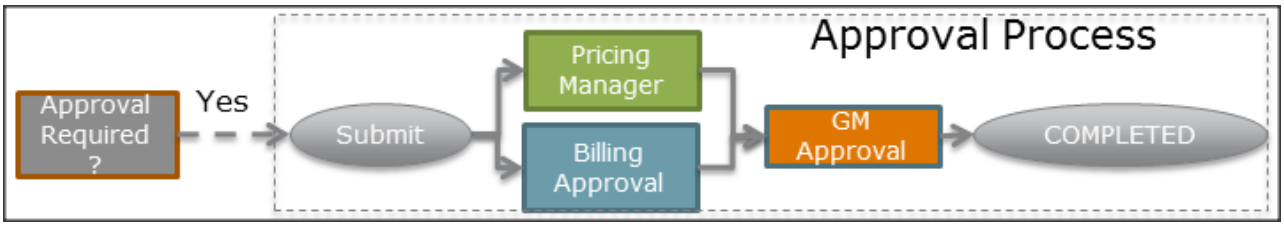
The **Approval Status** picklist is added to the **Custom Fields & Relationships** related list and the API Name *Approval_Status__c* is automatically added to the picklist.

Approval Required Check for Agreement

Approval Required check is performed to determine whether a custom object, Conga object, standard Salesforce object or its line item requires an approval and if an approval process needs to be executed. Approval Required check sets the approval status of the header or child object to *Approval Required*. Approval Required check does not evaluate the approval process, but rather performs a simple check outside of the approval process definition. Approval Required Check sets the approval status of the header or child objects items to *Approval Required* to make it easy to check whether approval is needed. The purpose of the Approval Required check is to reduce the overhead of performing detailed process evaluation to determine if approval is needed. While defining the Entry criteria of the Approval Process in conjunction with Approval Required check, you should use the *Approval Required* approval status as one of the criteria.

For example, Approval Required check is invoked prior to the execution or evaluation of the approval process. Approval criteria or conditions are defined using **Search Filters(Approvals)** of type *Entry Criteria*. Approval Required Check is primarily used with

CPQ approvals, but can also be used with approval on other objects, such as Agreement and its related objects.



Steps to Set up Approval Required check

- Adding Approval Status fields to an object
- Adding the custom object to the search filter
- Setting up the Approval Required check trigger
- Setting up an entry criteria for the Approval Required check

Workflow Rules: An alternative to Approval Required Check

If you do not wish to configure and use Approval Required Check, you can use the Salesforce functionality of Workflow Rules to set the Approval Status of your header or child object records to *Approval Required.* While defining Workflow rules, you must ensure that every time an update is made to the existing header or child object records, all the eligible entries are set to *Approval Required.*

For details on how to create and configure Workflow Rules to match your business requirements, see Workflow Rules.

To create an approval required check trigger

Approval Status picklist enables triggers in the Approval Required check process. Ensure that you have added Approval Status picklist for your context object by navigating to **Build > Custom object (Object Name) > Fields > New** and adding the values: *Not Submitted (default), Approval Required, Pending Approval, Approved, Rejected, Cancelled, Closed.*

1. Use the following two global methods of the ApprovalRequiredCheck class to create your Approval Trigger or Workflow Rule.
    a. doCheck(Id objId) – This API takes up the IDs of all the records of your context object and checks if these records satisfy the Entry Criteria defined at the Search Filter (Approvals). If the Entry Criteria is satisfied, this API returns true, otherwise false. For example, if you have approval processes defined for your

Agreement header fields and Agreement Line Items, this API checks if your Agreement record and Agreement Line Items satisfy the Entry Criteria (RecordType == NDA) returns True or False, and sets the Approval status of filtered records to Approval Required. Also, if there is a change in the value of your Agreement header or Agreement Line Item, and the entire approval request is re-submitted, this API considers all record IDs for Approval Required Check and re-evaluates them.
If a child object record, such as an Agreement Line Item is already Approved, this API does not consider these Line Items. For rest of the Approval Statuses, such as Cancelled, Rejected, Not Submitted, this API is useful.
This API is invoked as follows:
Apttus_Approval.ApprovalRequiredCheck checker = new Apttus_Approval.ApprovalRequiredCheck()
Boolean result =checker.doCheck(objId)

b. doCheck(SObject object) – This API returns a map of approval status values keyed by object ids for your context object. The first entry in the map corresponds to the header context object. The Key-Value pair indicates what Approval Status should be set for a specific record of your context object. For example, if you make any change in one of the field of Agreement header, such as Agreement End Date and the Approval Status for your Agreement record is *Cancelled.*
This API is invoked in a trigger before update as follows:
Apttus_Approval.ApprovalRequiredCheck checker = new Apttus_Approval.ApprovalRequiredCheck()
Boolean result = checker.doCheck(object)

2. Add the custom object to a search filter criteria by clicking **Create Objects > Search Filter (Approvals) > Business Object**. Add the API name of your custom object to the pick list.
3. Create a trigger that sets the Approval Required check for your custom object.
4. Navigate to **Conga Approvals Management > Search Filters (Approvals)** tab and set up your entry criteria in this section.

After you setup the Search Filter Criteria for your object, define the Approval Process for it. If the approval criteria is fulfilled, the approval status of the object changes based on the trigger you configure.

Creating Search Filters (Approvals)

The two Filter Type options (Child Filter and Entry Criteria) serve two different purposes.

Entry Criteria	This is required for approvals to be initiated for a custom object. To improve performance, the system will check if there are any Entry Criteria search filters that match the attributes of the custom object and set the approval status to Approval Required. You can have the entry criteria at the line item to set the approval status to Approval Required or Approval Triggered.
Child Filter	This is required to have a header-level approval-step entry criteria that makes use of the line item fields for the header custom object. After you create a child filter, you then need to create step-entry criteria using child filters that mirrors the expression used in the child filter.

Configuring and Creating Search Filters

Search filters of type *Entry-Criteria* are used to check whether the object requires approval before it can be finalized. If you want approvals to be triggered at the *header* level, then you need to configure search filters based on the context object. If you want approvals to be triggered at a Line Item level, then you would configure search filters for the Line Item context object. Additionally, if you want consolidated approvals for the whole object configuration, then you would create both. You must follow the procedure listed in the following section to configure and create search filters.

Configuring Search Filters - this adds custom objects to the Search Filter object picklist so you can create search filters that use these objects for their entry criteria.

Creating Search Filters - the Search Filters are used to determine if the products in the object require approval.

To configure Search Filters for Approvals

1. Go to **Setup > Build > Create > Objects** and select **Search Filter (Approvals)**.
2. From the **Custom Fields & Relationships** section, select **Business Object**.
3. From **Picklist Values**, click **New** and enter the API names for the Header object and the line items: `Header Object API name` for Header level approvals and `Child Object API name`for Line Item level approvals.
4. Click **Save**. The values are added to the **Picklist Values list**.
5. Click **Back to Search Filter (Approvals)** and from the **Custom Fields & Relationships** section, select **Filter Type**.

6. From **Picklist Values**, click **New** and enter `Child Filter` and `Entry Criteria`.
7. Click **Save**.

The values are added to the picklists, which are required to create search filters.

You can create *Search Filters (Approvals)* for controlling which objects are checked to see if they need approval.

To create Search Filters for Approvals

1. Click **All tabs** and select **Search Filter (Approvals).**
2. Click **New Search Filter (Approvals)** and select one of the following options.
   - **Header Object Name** - select this if you want the filter to be at the context object level.
   - **Child Object Name** - select this if you want the filter to be at the Line Item context object level, which are the individual line items within the Header object.
3. Click **Next**. The Configure Filter page appears.
4. From **Filter Type**, select one of the following options.
   - **Child Filter** - This enables you to have step entry criteria within your approval process that reference the child record fields of the main context object. This enables you to use line item fields with your Header Object Name approval process. The line item child Filters you create here enable you to have them available for approval step entry criteria.
   - **Entry Criteria** - Indicates that the search filter will be used for the Approval Required check, which checks whether the header or header line item values of an object required to undergo an approval. Essentially, you have and an additional level of entry criteria for approvals here, before the system searches for matching approval processes.

   > **Note**
   >
   > You must have set at least either the Entry Criteria or the Child Filter for approvals to be available from the parent object.

5. Enter a required **Filter Name** and **Sequence**. The sequence is used to determine the order in which the search filters will check if approvals are required.
6. Enter the **Filter criteria**, which is used to determine if the object requires approval.
7. Click **Save**.

The Search Filter is now active. If it is an Entry Criteria filter, the system will determine if a header object requires approval.

You must create an Approval Process that can be used with the header object.

Standard Configuration for Preview & Submit Approvals and My Approvals button for setting Approvals on any Object

Prerequisite:

Before you start configuring the new buttons on your context object, ensure that you have created a record with the API name of the context object in the **Approvals Custom Config** available at **Setup > Develop > Custom Settings**.

For example, if you want to configure Approvals on Agreement object and its Child objects, create a record for each of the objects in the **Approvals Custom Config** and perform the following steps. So, to configure Approvals for Agreement and Agreement Clause objects, create 2 records with the API names, Apttus__APTS_Agreement__c and Apttus__Agreement_Clause__c respectively. For more details, refer to Configuring Global Settings for Agreement and its related Objects.

To enable Approvals for your context object, you must configure the following 2 buttons and place them on your object record's page layout.

- **Preview & Submit Approval** - For actions such as **Preview**, **Submit**, **Recall for Submitter**.
- **My Approvals** - For actions such as **Approve**, **Reject**, **Add Ad hoc Approver**, **Reassign**.

You can follow any one of the following procedures to configure **Preview & Submit Approvals** button for your context object:

Creating a custom button for your context object:

1. Go to **Setup > Create > Objects** and search for your object.
2. In the **Buttons, Links, and Actions** section, click **New Button or Link.**
3. In the Label field, enter **Preview & Submit Approvals**.
4. In the **Display Type**, select *Detail Page Button*.
5. In the **Behavior** drop-down list, select *Display in existing window without sidebar or header*.
6. In the **Content Source** drop-down list, select *URL*.

7. Enter the following formula in the Advanced Formula section, **/apex/Apttus_Approval__PreviewSubmitApprovals?id={!context-object-api-name.Id}**
   Ensure that in the **context-object-api-name,** you enter the API name of your context object.
   In our example for Agreement object, enter *Apttus__APTS_Agreement__c* as **context-object-api-name**.
8. Click **Save**.
9. Go to a record of your object, click the **Edit Layout** link.
10. From the Buttons list, drag and drop the newly created button on your page layout and click **Save.**

Creating a custom formula field for your context object:

1. Go to **Setup > Create > Objects** and search for your object. In our example, select Agreement object of Contract Management package.
2. In the Custom Fields & Relationships section, click **New.**
3. Select **Formula** as the field type and click **Next**.
4. In the Field Label, enter **Preview & Submit Approvals**, select *Text* as **Formula Return Type** and click **Next**.
5. In the Advanced Formula text box, enter the following formula:
   IF(AND(NOT(ISPICKVAL(Approval_Status__c,"Pending Approval")), NOT(ISPICKVAL(Approval_Status__c,"Approved"))), HYPERLINK("/apex/Apttus_Approval__PreviewSubmitApprovals?Id="&Id, IMAGE("/resource/[button-image]", "Preview & Submit Approvals"),"_self"),NULL)

   Ensure that you enter the API name of the field you use for Approval Status of your context object, which you defined before configuring this button.
   Along with **ID** as a mandatory parameter, you can also append the following optional parameters to capture more details by a button click. The following parameters are applicable only for CPQ approvals

Parameter name	Datatype	Use of the parameter
ctxChildSObjectId	ID of the child context object	To view subset of approvals required for the specified child context object.

Parameter name	Datatype	Use of the parameter
returnId	ID of the object	To return to detail view of the object referred to by this identifier.
returnPage	string	To return to the specified page.
returnButtonLabel	string	To use the specified label for the return button. This is useful when this page is invoked from multiple locations or pages.
autoPreviewIndicator	boolean	To force the system to perform Preview before displaying approval requests.
hideSubmitWithAttachments	boolean	Indicator to instruct the system to hide **Submit with Attachments** button.
headerSObjectId	Id of the standard object	The ID to be rendered as a link in the header of the page, if different from the context object. This is used to display the Quote link instead of Cart or Product Configuration link.
headerSObjectTitle	string	The title or label to display as a link in the header next to the ID, if one wants to display a value other than the **Name** field value of the context object.

6. After checking the syntax of your formula field, click **Next**.
7. Select visibility for your required profiles and click **Next**.
8. Select appropriate page layouts on which you want to display the new field and click **Save**.

Similarly, for configuring **My Approvals** button for your context object, follow any one of the following procedures:

Creating a custom button for your context object

1. Go to **Setup > Create > Objects** and search for your object.
2. In the **Buttons, Links, and Actions** section, click **New Button or Link.**
3. In the Label field, enter **My Approvals**.
4. In the **Display Type**, select *Detail Page Button*.
5. In the **Behavior** drop-down list, select *Display in existing window without sidebar or header*.
6. In the **Content Source** drop-down list, select *URL*.
7. Enter the following formula in the Advanced Formula section,
   /apex/Apttus_Approval__MyApprovals?id={!context-object-api-name.Id}

   Ensure that in the **context-object-api-name**, you enter the API name of your context object.
   In our example for Agreement object, enter Apttus__APTS_Agreement__c as context-object-api-name.
8. Click **Save**.
9. Go to a record of your object, click the **Edit Layout** link.
10. From the Buttons list, drag and drop the newly created button on your page layout and click **Save.**

Creating a custom formula field for your context object

1. Go to **Setup > Create > Objects** and search for your object. In our example, select Agreement object of Contract Management package.
2. In the Custom Fields & Relationships section, click **New.**
3. Select **Formula** as the field type and click **Next**.
4. In the Field Label, enter **My Approvals**, select *Text* as **Formula Return Type** and click **Next**.
5. In the Advanced Formula text box, enter the following formula:
   IF(OR((ISPICKVAL(Approval_Status__c,"Pending Approval")), (ISPICKVAL(Approval_Status__c,"Approved"))), HYPERLINK("/apex/Apttus_Approval__MyApprovals?Id="&Id, IMAGE("/resource/[button-image]", "My Approvals"),"_self"),NULL)

   Ensure that you enter the API name of the field you use for Approval Status of your context object, which you defined before configuring this button.
6. After checking the syntax of your formula field, click **Next**.
7. Select visibility for your required profiles and click **Next**.

8. Select appropriate page layouts on which you want to display the new field and click **Save**.

Licensing Approvals for Agreement and its related Objects

For each user that needs to preview or submit approvals for a custom context object, they must have a license. This is handled through the *Conga Approvals Custom Management* package.

To license users

You must have installed the license package.

1. Go to **Build > Installed Packages** and click **Manage Licenses for Conga Custom Approvals Management**.
2. Click **Add Users** and search for who you want to license to use approvals with custom objects.

All users in the Licensed Users list can use Approvals with Custom Objects.

Creating Term Exception Approval Processes

Term Exception Approvals are unique to Term Exceptions and cannot be used with other objects, unlike a standard Approval Process that can be used with objects like Agreements, Opportunities, or Quote/Proposals.

An Approval with Term Exception is useful when your customer is requesting a change in an existing clause which requires approval from the CFO of your company, and this clause is specific to a single customer.
Considering the above example, if you renew an agreement for a specific customer and your customer wants to enter a new clause. In this scenario, you want your Legal team to approve the new clause before adding it to the renewed agreement. So, you add the new clause, associate it with a term exception and define a term exception approval for it.

A Term Exception can have multiple clauses and correspondingly multiple approval requests associated to it. These approval requests are configured using the Term Exception Approval records and are triggered when the user clicks Submit or Preview on the Agreement Term Exception record under an agreement.

The normal flow to trigger an approval request for an agreement term exception is:

1. Create a Term Exception Approval record and specify Term Exception Approval Conditions (optional).
2. For your agreement record, create Agreement Term Exception records for the Term Exception used in Step 1.
3. Submit or Preview the Agreement Term Exception record for approval.

Term exception approvals are made up of a main approval object, which can contain multiple approval conditions. The table below provides the agreement and the corresponding term status for term exceptions approval requests.

Agreement Status	**Term Status**
Approved	All terms are approved.
Rejected	One term is rejected. Note: If a user clicks Submit for Approval after a term is rejected, the Rejected term and new additions to terms are routed for an approval and status changes to Pending Approval.
Not Submitted	All terms are Not Submitted. Note: If a Valid To Date changes, or a new Term is added to a request in progress, the Agreement Status becomes Not Submitted. Only new terms should be routed for approval. If a user wants to re-submit all terms, they should Cancel and then click Submit for Approval again.
Cancelled	All terms are cancelled. Note: If a user clicks Cancel, status of all terms changes to Cancelled. If the user then clicks Submit for Approval, all terms should move to Pending Approval.
Pending Approval	Status of one or more term is Pending Approval (none are Rejected).
Closed	Order Operations will move the Agreement Status to Closed once orders are being booked. It should then no longer be possible for anyone to add or change terms.

Agreement Status	Term Status
Expired	An automated Agent will change the Agreement Status to Expired when a Valid To Date is passed. If there are changes to the Valid To Date, the Agreement Status should change to Not Submitted. If the user then clicks Submit for Approval, the Agreement Status and status of all terms is Pending Approval.

To create a Term Exception Approval

There must be an existing Term Exception to associate the approval with. For details on how to create Term Exception records, refer to the *Contract Management Administrator Guide*.

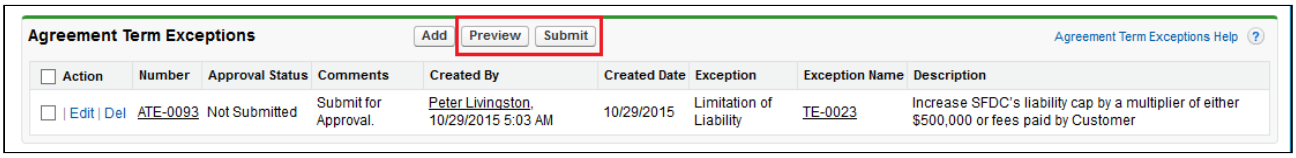
1. Select the **Term Exception Approvals** tab or click **All tabs** and select **Term Exception Approvals**.
2. Click **New** to display the Edit Term Exception Approval page.
3. Use the lookup to select the Term Exception and any dependencies.
4. Select the process approvers, by selecting from **Assigned Approver > Assignee Type**.

   Note

   If you select *Rule* as an **Assignee Type**, the approval request is sent only to the first user that satisfies the filter criteria in the Approval Rule. If you have multiple approvers in a sequence, only the first approver is selected and assigned.
5. Select **Send Email** to notify the assignee by email that they must approve or reject the object record and select **Notify Only** to only inform the assignee by email that the approval process has reached this step, without any actions to take.
6. Select **Active** to enable the approval to be used with the Term Exception
7. Click **Save**.

An approval process is now in place for the Term Exception. If needed, you can create Term Exception Approval Conditions; however, they are not a required part of a Term Exception Approval.

Now, you must create an Agreement Term Exception record under your Agreement, choose the specific Term Exception for which you configured Term Exception Approval and click **Submit.** The approval process will begin.



To create a Term Exception Approval Condition

1. Select a **Term Exception Approval** and click **New Term Exception Approval Condition**.
   By default, the condition is active and already has the Term Exception Approval field populated.
2. Enter values for the following:

Option	Description
Number	This is the sequence number. Conditions are evaluated in the order specified by this field. When a condition evaluates to *false*, remaining conditions are ignored.
Object Qualifier	This is required to qualify the object and associate it with the condition.
Object	This is the object the condition applies to. The API name of the field should be used.
Field	This is the field associated with the object, that is used to evaluate whether the condition is true or false. The API name of the field should be used.
Operator	This is used to compare *Field* and *Value*.
Value	These values are compared with *Field*. Multiple values can be included by separating them with a comma, such as`value1, value2, value3`. This creates an OR relationship between the values.

3. Set the Join field to AND or OR to establish its relationship with subsequent conditions.
4. Enter a **Description** and click **Save**.

Configuring Approvals for Custom Objects

After the Approvals package has been installed, there are Setup configuration options that can be changed.

By default, the package installation sets up the majority of the application to be used as is; however, some organizational specific options can only be configured after the installation. Some post-package installation configuration must be done to best maximize the features available in Approvals. There are also default settings that you may want to change, to better suit your business.

This section provides the workflow and the sequence of actions you have to define to enable Custom Object Approvals.

Workflow

To do...	How to define...
Install required packages	Install the latest approvals and dependent packages.
Adding Values to Approval picklists	Add object names to processes, rules, and dimensions.
Approvals System Properties	Create a record in Approval System Properties and enter the requisite details. For details, see Mandatory Setting for Agreement Objects.
To configure custom object settings	Create a record in Approval Custom Config for the primary custom object- Setup the context type as Multiple if you are defining an approval process for the header and child object and set it as Single if you are using approvals only on one object.
Add pick list values for approval fields	• Create Approval Status and Approval Preview Status fields on the header object. • Create Approval Status and Approval Preview Status fields on the child object.

To do...	How to define...

Configure Approval Required Check Settings	• Setup Search Filter (Entry Criteria) for header (e.g. Opportunity).
	• Setup Search Filter (Entry Criteria) for line items (e.g. Opportunity Products).
	• Create the Approval Required Check Trigger for the header and line item object.
	• Invoke the Approval Required Check API within the trigger for the header and line item object and click **Save**.
	• Set Header and line item approval status based on the Approval Required Check API.
	• Pass the line items that need to be evaluated for approvals when the line items and header information is created or updated.
	APIs to be invoked:
	  • Apttus_Approval.ApprovalsWebService.isApprovalRequired (customobjectAPIName, customobject.Id)
	  • Apttus_Approval.ApprovalsWebService.CheckIfApprovalRequired2 (headerIdStatus, childIdStatusList, modifiedChildObjectIds)
	Approval status at the line item as well as header should be set to Approval Required for preview / submit to work.

Create Approvals Page	Create the Approvals Page using Conga *Approvals Management* package as reference. Link the Approvals page to a button of the primary page header, for example Approvals. This page will be used by a requestor to preview approvals, submit the object for approvals, or cancel or recall approvals while the object is going through approvals. This page will also be used by requestors to view approval history.

To do...	How to define...
Create My Approvals Page	Create the My Approvals Page using Conga *Approvals Management package* as reference. Link this page to a button of the primary object header such as My Approvals. This page will be used by Approvers to view all header and line items that require approval. Approvers can also use this page to take ownership of any queues that they are part of.
Setting up Email Templates and Alerts	While provides default email templates for approval requests, if you want the emails to have attachments or have your own specific content, you must customize the templates.
Setup Approval Process	Configure Approval Process • Use the Approval Required status as process entry criteria. • Create additional pick list values for Approval Status etc. • Setup approval rules on the child context object to be used for the step of type Child Process. • Setup approval rules on the primary context object to be used for the step of type Subprocess. • Approval rules of type Condition can also be defined if the approval rule entry criteria is complex expression. • Setup the initial and final submission actions.
Setting up Approval Rules	Create Approval Rules for each Sub Process and Child Process if you have multiple approvers for a process. Configure Approval Rule criteria. Select Approval Rule Type as Condition to use Auto Re-approval conditions. For each step in sub-process or child process, create approval rule entry, step label, Approval Condition, Auto Re-approval Condition, and Assignee, and step dependency.
Setting up Backup or Delegate Approvers	Backup approvers are required to 'catch' approval requests whose assignees cannot be identified and ensure approval processes progress as expected.

To do...	How to define...
Setting up the Backup Administrator	The backup admin user is responsible for handling routing issues that may occur during the approval process, while admin profiles enable you to associate users – via their roles – with admin level access to Approvals.
Setup Page Layout for Primary Object	Add Approvals and My Approvals buttons to primary object layout.

Global Settings

Optional Configuration

Auto-escalating Approval Requests	When an approval request is not approved, rejected, or reassigned within the allotted time, it can automatically be reassigned to a new approver so the approval process can continue.
Changing Approval Rule Step Behavior	Typically steps that cannot identify an assignee are sent to the backup administrator; however, you can change that behavior and have those steps skipped instead.
Changing the Assignee Search page results list	This controls the number of assignees that can be displayed on a single search page. Typically the default value of twenty is suitable for your organization, but you can increase or decrease the value
Creating Approval Matrices	Matrices are used with *Global Discount Policies* to route approvals to users based on system-wide default discount percentages.
Approval Request History	When an approval process is initiated – by submitting or resubmitting an approval – for an agreement, term exception, or opportunity, an approval history record is created. The Approval request history Object comprises information for old approval requests such as, the approval status, the date of the request, who it was assigned to, and who the actual approver was.

Submitting a Request with Attachments	Enables you to submit your approval request with an attachment. To submit a request with attachment, ensure that you have created custom email notification templates and specified the name of each of the templates in the Approval Process of the object.
Approval Submission Comments	Approval submission comments enable you to add personalized comments when you submit an approval request. You can have a single comment at the process level or up to three comments at the step level.
Child Filter Objects	From 7.0 onwards you can use a child process to configure approvals on child objects. However, if you wish to create approval steps based on child object fields, you can use custom child object fields. Create an child filter expressions to filter the records based on child object fields.
Opportunity records and ACM	When Advanced currency management (ACM) is enabled in your org, you cannot have any Opportunity record fields of data type currency in your email templates or approval summary page.

Performance and Scalability

Area	Detail
Multiple Approval Processes	When to use multiple approval process? When you have Region Based Approval variations, Business Unit Based Approvals variations, Channel Based Approvals variations. How do I configure multiple approval processes? Define mutually exclusive process entry criteria so that only one process is selected for a given primary object instance. Why do I configure multiple approval processes? Having multiple approval processes reduces the number of approval steps/criteria system needs to execute

Area	Detail
Approval Rule Entry Criteria	Specify minimum approval conditions in Approval Rule entry criteria for Sub Process or Child Process steps to avoid evaluation of all approval rule entries every time.
Approval Required Check	Provides users visibility to whether approval is needed for header or line items without executing the entire approval process. Provides a mechanism to avoid approval process evaluation for a large number of scenarios.

Mandatory Configuration for Custom and Agreement Objects

This section outlines the mandatory settings to setup Approvals for your objects.

- Approval System Properties
- Global Settings for Custom Objects and its Related Objects
- Setting up the Admin Profiles
- Configuring Email Templates
- Setting up Intelligent Approval Flag

Configuring Email Templates for Custom Objects

Typically the easiest way to create your own custom template is to clone one of the email templates provided in the unmanaged Conga samples package and then edit it. After that has been done you can then override the default email templates for all of your approval processes using APTS_ApprovalConfig or on an ad-hoc basis for specific approval requests.

The Conga Approvals package provides email template examples for currently supported objects, as well as for Salesforce Cases, which can subsequently be used as the basis for creating email templates for custom objects you've configured yourself. Also note that Opportunity Product is also used to provide templates for further customization.

You have to specify the email templates in the Approval Process and in the Approvals Custom Config custom setting.

As well as providing samples for templates and the summary page, the pages that are used to provide approval functionality on the custom object record page are also included.

Note

As this package is unmanaged, it cannot be upgraded. If you want to move to the newest version, you must uninstall the existing package and then install the new package. It is recommended you uninstall the samples package after you have created your custom templates and pages.

The Conga Approvals package contains email templates, components, Visualforce pages and Apex controllers which can help you build you templates and approval summary pages for custom context objects.

Note

The samples contained in the package should not be used 'as is' and are only to be used in the development of your own bespoke templates and pages, and should not be installed in production.

Configuring Email Templates

For a process with header and child process steps, use the Conga Approval Email Templates. Notification templates are used to create the notifications sent to approvers to alert them to pending approvals and to allow them to approve or decline a request without logging in to Salesforce.

The following approval email notification templates are available as a part of the standard Approvals Management package. These are generic email notification templates that can be used for approvals on any standard or custom objects without any change. However, these notification templates can be customized as needed. Clone these templates if you want to customize your own template.

Sample	Component Type	Description
Conga My Approvals Notification (Assignment)	Visualforce Email Template	This template is used to notify users they must decide on an approval request.
Conga My Approvals Notification (Reassignment)	Visualforce Email Template	This template is used to notify users that an approval request task initially assigned to them has been reassigned to someone else.

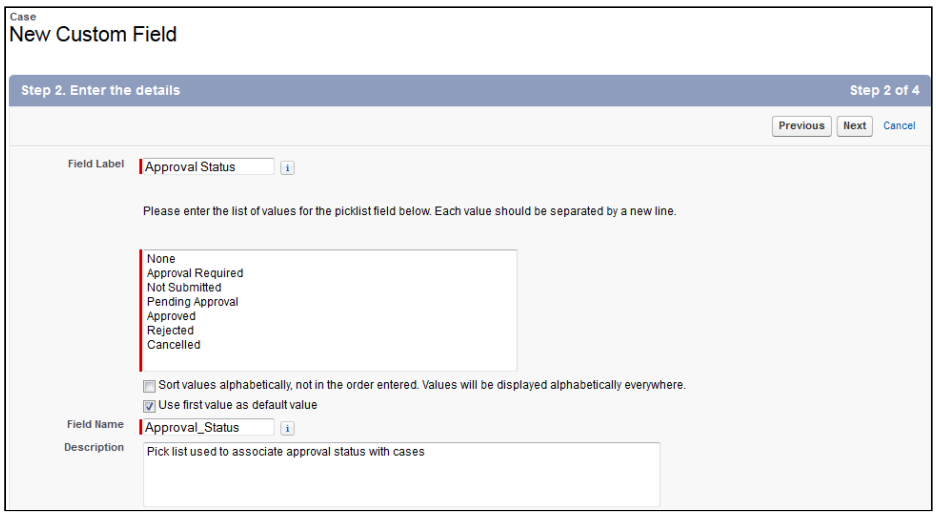
Sample	Component Type	Description
Conga My Approvals Notification (Escalation)	Visualforce Email Template	This template is used for auto-escalation. Configuration is required to activate this feature.
Conga My Approvals Notification (Cancellation)	Visualforce Email Template	This template is used to notify users that an approval request has been cancelled.
Conga My Approvals Notification (Notify Only)	Visualforce Email Template	This template is used to notify users only.

Adding Approval Status to Objects

After the picklists have been set for objects, it's time to set the object picklist field so that Approval Status is available on the object record.

To add a picklist field for approvals to objects

1. Go to **Build > Customize > {Object Name} > Fields**.
2. In the **Custom Fields & Relationships** section, click **New**.
3. Select **Picklist** data type and click **Next.**
4. On the Details page, enter the following to ensure approval statuses will be available for the custom object and click **Next**.



5. Select which users will be able to see the picklist and click **Next**.

6. Select which page layouts will include the picklist and click **Save**.

The **Approval Status** picklist is added to the **Custom Fields & Relationships** related list and the API Name *Approval_Status__c* is automatically added to the picklist.

Lookup Relationship with Approval Request Object

To have visibility on Approval Requests that are associated with the custom object in a related list or in Approval Center, you need to create a look-up relationship between the Approval Request object and your custom object.

Adding Values to Approval picklists

When creating an approval process, to be able to associate it and its various components with a custom object, you must have the custom object available from approval picklists.

Any part of the approval process creation that includes linking approvals to a context business object, needs to have the custom object added to it. This includes the approval process, approval rule dimension, approval rule, and search filter.

To add a new Object to Approval Process picklist

1. Go to **Build** > **Create** > **Objects** > **Approval Process** and click **Object Label**.
2. In the **Picklist Values** section, click **New**.
3. Enter `Object Name` in the text field and then select **Approval Process**.
4. Click **Save**.

Object is added to the Picklist Values list for the Approval Process.

The custom object API name must be added to the pick list.

To add API name to Approval Process picklist

1. Go to **Build** > **Create** > **Objects** > **Approval Process** and click **Object Type**.
2. In the **Picklist Values** section, click **New**.
3. Enter `Object API name` in the text field and then select **Approval Process**.

4. Click **Save**.

The Object is added to the Picklist values and can now be associated with an approval process during the creation process.

To add API name to Approval Rule Dimension picklist

1. Go to **Build > Create > Objects > Approval Rule Dimension**.
2. In the **Custom Fields & Relationships** section, click **Business Object**.
3. In the **Picklist Values** section, click **New**.
4. Enter `Object API Name` in the text field.
5. Click **Save**.

The Object can now be associated with dimensions during the creation process

To add API name to Approval Rule picklist

1. Go to **Build > Create > Objects > Approval Rule**.
2. In the **Custom Fields & Relationships** section, click **Business Object**.
3. In the **Picklist Values** section, click **New**.
4. Enter `Object API name` in the text field.
5. Click **Save**.

The Object can now be associated with approval rules.

To add API name to Search Filter picklist

1. Go to **Build > Create > Objects > Search Filter**. In instances when there are multiple options, be sure to select the *Search Filter* associated with Conga *Approvals Management*.
2. In the **Custom Fields & Relationships** section, click **Business Object**.
3. In the **Picklist Values** section, click **New**.
4. Enter `Object API Name` in the text field.
5. Click **Save**.

Configuring My Approvals and Approvals Pages

Pre requisite:

Before you start configuring the new buttons on your context object, ensure that you have created a record with the API name of the context object in the **Approvals Custom Config** available at **Setup > Develop > Custom Settings**.

For example, if you want to configure Approvals on Agreement object and its Child objects, create a record for each of the objects in the **Approvals Custom Config** and perform the following steps. So, to configure Approvals for Agreement and Agreement Clause objects, create 2 records with the API names, Apttus__APTS_Agreement__c and Apttus__Agreement_Clause__c respectively. For more details, refer to Configuring Global Settings for Objects.

To enable Approvals for your context object, you must configure the following 2 buttons and place them on your object record's page layout.

- **Preview & Submit Approval** - For actions such as **Preview**, **Submit**, **Recall for Submitter**.
- **My Approvals** - For actions such as **Approve**, **Reject**, **Add Ad hoc Approver**, **Reassign**.

Configuring the My Approvals button

The My Approvals page is the page within Salesforce where you can act on the approval requests assigned to you.

The My Approvals page is primarily used by the requester for the following tasks.

- View the list of items that need approval from the current logged in approver
- To take ownership of the approval items that are assigned to queues or roles, if the current user is a member of the queue / role
- To approve or reject one or more assigned approval items
- To add ad hoc approver to the list of approvers

For configuring **My Approvals** button for your context object, follow any one of the following procedures:

- Creating a custom button for your context object

1. Go to **Setup > Create > Objects** and search for your object.
2. In the **Buttons, Links, and Actions** section, click **New Button or Link.**
3. In the Label field, enter **My Approvals**.
4. In the **Display Type**, select *Detail Page Button*.
5. In the **Behavior** drop-down list, select *Display in existing window without sidebar or header*.

6. In the **Content Source** drop-down list, select *URL*.
7. Enter the following formula in the Advanced Formula section,
   /apex/Apttus_Approval__MyApprovals?id={!context-object-api-name.Id}

   Ensure that in the **context-object-api-name**, you enter the API name of your context object.
   In our example for Agreement object, enter Apttus__APTS_Agreement__c as context-object-api-name.
8. Click **Save**.
9. Go to a record of your object, click the **Edit Layout** link.
10. From the Buttons list, drag and drop the newly created button on your page layout and click **Save.**

- Creating a custom formula field for your context object

1. Go to **Setup > Create > Objects** and search for your object. In our example, select Agreement object of Contract Management package.
2. In the Custom Fields & Relationships section, click **New.**
3. Select **Formula** as the field type and click **Next**.
4. In the Field Label, enter **My Approvals**, select *Text* as **Formula Return Type** and click **Next**.
5. In the Advanced Formula text box, enter the following formula:
   IF(OR((ISPICKVAL(Approval_Status__c,"Pending Approval")), (ISPICKVAL(Approval_Status__c,"Approved"))), HYPERLINK("/apex/Apttus_Approval__MyApprovals?Id="&Id, IMAGE("/resource/[button-image]", "My Approvals"),"_self"),NULL)

   Ensure that you enter the API name of the field you use for Approval Status of your context object, which you defined before configuring this button.
6. After checking the syntax of your formula field, click **Next**.
7. Select visibility for your required profiles and click **Next**.
8. Select appropriate page layouts on which you want to display the new field and click **Save**.

Configuring the Preview & Submit Approvals button

The Approvals page enables you to preview the approval requests assigned to you. The sample files provide the base for customizing the information to display on the page. However, you must ensure you keep the Approval Action code on the page so that users can

approve or reject requests assigned to them. The Approvals page is primarily used by the requester for the following tasks.

- Preview a list of all approvals that will be required before they are submitted for approval. You can preview and submit approvals for complex processes.
- View a list of all approvals including those that have already been submitted and approved or rejected, or are in another status such as Assigned, On Hold, or Not Submitted. You can also view a large number of requests in the my approvals page without view state errors.
- Allow the submitter to recall all approvals for the current context object.

You can follow any one of the following procedures to configure **Preview & Submit Approvals** button for your context object:

- Creating a custom button for your context object:

1. Go to **Setup > Create > Objects** and search for your object.
2. In the **Buttons, Links, and Actions** section, click **New Button or Link.**
3. In the Label field, enter **Preview & Submit Approvals**.
4. In the **Display Type**, select *Detail Page Button*.
5. In the **Behavior** drop-down list, select *Display in existing window without sidebar or header*.
6. In the **Content Source** drop-down list, select *URL*.
7. Enter the following formula in the Advanced Formula section, **/apex/Apttus_Approval__PreviewSubmitApprovals?id={!context-object-api-name.Id}** Ensure that in the **context-object-api-name,** you enter the API name of your context object.
   In our example for Agreement object, enter *Apttus__APTS_Agreement__c* as **context-object-api-name**.
8. Click **Save**.
9. Go to a record of your object, click the **Edit Layout** link.
10. From the Buttons list, drag and drop the newly created button on your page layout and click **Save.**

- Creating a custom formula field for your context object:

1. Go to **Setup > Create > Objects** and search for your object. In our example, select Agreement object of Contract Management package.
2. In the Custom Fields & Relationships section, click **New.**
3. Select **Formula** as the field type and click **Next**.

4. In the Field Label, enter **Preview & Submit Approvals**, select *Text* as **Formula Return Type** and click **Next**.
5. In the Advanced Formula text box, enter the following formula:
   IF(AND(NOT(ISPICKVAL(Approval_Status__c,"Pending Approval")), NOT(ISPICKVAL(Approval_Status__c,"Approved"))), HYPERLINK("/apex/Apttus_Approval__PreviewSubmitApprovals?Id="&Id, IMAGE("/resource/[button-image]", "Preview & Submit Approvals"),"_self"),NULL)

   Ensure that you enter the API name of the field you use for Approval Status of your context object, which you defined before configuring the My Approvals, Preview, and Submit button.
   Along with **ID** as a mandatory parameter, you can also append the following optional parameters to capture more details by a button click.
   These parameters are applicable only for CPQ.

Parameter name	Datatype	Use of the parameter
ctxChildSObjectId	ID of the child context object	To view subset of approvals required for the specified child context object.
returnId	ID of the object	To return to detail view of the object referred to by this identifier.
returnPage	string	To return to the specified page.
returnButtonLabel	string	To use the specified label for the return button. This is useful when this page is invoked from multiple locations or pages.
autoPreviewIndicator	boolean	To force the system to perform Preview before displaying approval requests.
hideSubmitWithAttachments	boolean	Indicator to instruct the system to hide **Submit with Attachments** button.
headerSObjectId	Id of the standard object	The ID to be rendered as a link in the header of the page, if different from the context object. This is used to display the Quote link instead of Cart or Product Configuration link.

Parameter name	Datatype	Use of the parameter
headerSObjectTitle	string	The title or label to display as a link in the header next to the ID, if one wants to display a value other than the **Name** field value of the context object.

Parameter name	Datatype	Use of the parameter
CancelPendingProcess (Deprecated from 7.2.370 onwards. Use Enable Resubmit instead)	boolean	• true:Cancel the previous approval requests on the object record and submit a new approval request for the object every time you click Submit. To configure this behavior, in the My Approvals Formula button for your context object set the CancelPendingProcess parameter to true. For example,/apex/ApprovalContextPreview?sObjectType=Apttus__APTS_Agreement__c&sObjectId={!Apttus__APTS_Agreement__c.Id}&CancelPendingProcess=true. • false:The existing Approval Process will not be cancelled. To configure this behavior, in the My Approvals Formula button for your context object set the CancelPendingProcess parameter to false. For example, /apex/ApprovalContextPreview?sObjectType=Apttus__APTS_Agreement__c&sObjectId={!Apttus__APTS_Agreement__c.Id}&CancelPendingProcess=false.

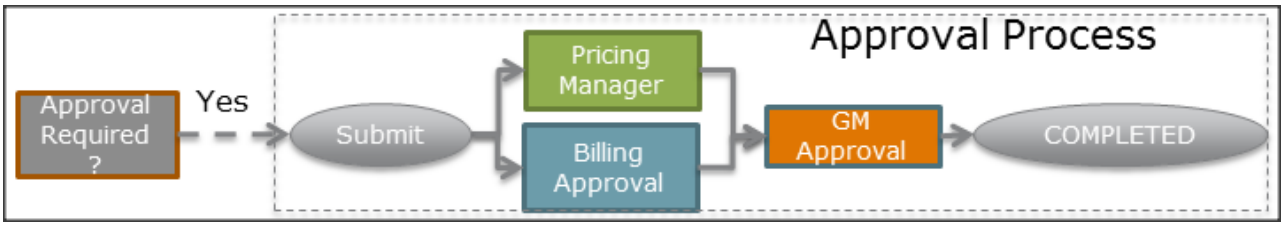
Parameter name	Datatype	Use of the parameter
		• If you do not pass any parameter in the formula field, the approval requests on the context object are cancelled and a new approval request is submitted.

6. After checking the syntax of your formula field, click **Next**.
7. Select visibility for your required profiles and click **Next**.
8. Select appropriate page layouts on which you want to display the new field and click **Save.**

Approval Required Check

Approval Required check is performed to determine whether a custom object, Conga object, standard Salesforce object or its line item requires an approval and if an approval process needs to be executed. Approval Required check sets the approval status of the object or line item to Approval Required. Approval Required check does not evaluate the approval process, but rather performs a simple check outside of the approval process definition. Approval Required Check sets the approval status of the header or line items to Approval Required to make it easy to check whether approval is needed. The purpose of the Approval Required check is to reduce the overhead of performing detailed process evaluation to determine if approval is needed. While defining the Entry criteria of the Approval Process in conjunction with Approval Required check, you should use the *Approval Required* approval status as one of the criteria.

For example, Approval Required check is invoked prior to the execution or evaluation of the approval process. Approval criteria/conditions are defined using **Search Filters(Approvals)** of type *Entry Criteria*. Approval Required Check is primarily used with CPQ approvals, but can also be used with approval on other objects, such as Agreement and its related objects.



Steps to Set up Approval Required check

- Adding Approval Status and Approval Preview Status fields to an object
- Adding the custom object to the search filter
- Setting up the Approval Required check trigger
- Setting up an entry criteria for the Approval Required check

Workflow Rules: An alternative to Approval Required Check

If you do not wish to configure and use Approval Required Check, you can use the Salesforce functionality of Workflow Rules to set the Approval Status of your header or line items from *Not Submitted to Approval Required.* While defining Workflow rules, you must ensure that every time an update is made to the existing header or line item records, the Approval Status is set based on the Entry Criteria of Search Filters(Approvals).

For details on how to create and configure Workflow Rules to match your business requirements, see Workflow Rules.

To create an approval required check trigger

Approval Status picklist enables triggers in the Approval Required check process. Ensure that you have added Approval Status picklist for your context object by navigating to **Build > Custom object (Object Name) > Fields > New** and adding the values: *Not Submitted (default), Approval Required, Pending Approval, Approved, Rejected, Cancelled, Closed*.

1. Use the following two global methods of the ApprovalRequiredCheck class to create your Approval Trigger or Workflow Rule.
    a. doCheck(Id objId) – This API takes up the IDs of all the records of your context object and checks if these records satisfy the Entry Criteria defined at the Search Filter (Approvals). If the Entry Criteria is satisfied, this API returns true, otherwise false. For example, if you have approval processes defined for your Agreement header fields and Agreement Line Items, this API checks if your Agreement record and Agreement Line Items satisfy the Entry Criteria (RecordType == NDA) returns True or False, and sets the Approval status of filtered records to Approval Required. Also, if there is a change in the value of your Agreement header or Agreement Line Item, and the entire approval request is re-submitted, this API considers all record IDs for Approval Required Check and re-evaluates them.
    If a child object record, such as an Agreement Line Item is already Approved, this API does not consider these Line Items. For rest of the Approval Statuses, such as Cancelled, Rejected, Not Submitted, this API is useful.

This API is invoked as follows:
Apttus_Approval.ApprovalRequiredCheck checker = new
Apttus_Approval.ApprovalRequiredCheck()
Boolean result =checker.doCheck(objId)

b. doCheck(SObject object) – This API returns a map of approval status values keyed by object ids for your context object. The first entry in the map corresponds to the header context object. The Key-Value pair indicates what Approval Status should be set for a specific record of your context object. For example, if you make any change in one of the field of Agreement header, such as Agreement End Date and the Approval Status for your Agreement record is *Cancelled.*
This API is invoked in a trigger before update as follows:
Apttus_Approval.ApprovalRequiredCheck checker = new
Apttus_Approval.ApprovalRequiredCheck()
Boolean result = checker.doCheck(object)

2. Add the custom object to a search filter criteria by clicking **Create Objects > Search Filter (Approvals) > Business Object**. Add the API name of your custom object to the pick list.
3. Create a trigger that sets the Approval Required check for your custom object.
4. Navigate to **Conga Approvals Management > Search Filters (Approvals)** tab and set up your entry criteria in this section.

After you setup the Search Filter Criteria for your object, define the Approval Process for it. If the approval criteria is fulfilled, the approval status of the object changes based on the trigger you configure.

Creating Search Filters (Approvals)

The two Filter Type options (Child Filter and Entry Criteria) serve two different purposes.

Entry Criteria	This is required for approvals to be initiated for a custom object. To improve performance, the system will check if there are any Entry Criteria search filters that match the attributes of the custom object and set the approval status to Approval Required. You can have the entry criteria at the line item to set the approval status to Approval Required or Approval Triggered.

Child Filter	This is required to have a header-level approval-step entry criteria that makes use of the line item fields for the header custom object. After you create a child filter, you then need to create step-entry criteria using child filters that mirrors the expression used in the child filter.

Configuring and Creating Search Filters

Search filters of type *Entry-Criteria* are used to check whether the object requires approval before it can be finalized. If you want approvals to be triggered at the *header* level, then you need to configure search filters based on the context object. If you want approvals to be triggered at a Line Item level, then you would configure search filters for the Line Item context object. Additionally, if you want consolidated approvals for the whole object configuration, then you would create both. You must follow the procedure listed in the following section to configure and create search filters.

Configuring Search Filters - this adds custom objects to the Search Filter object picklist so you can create search filters that use these objects for their entry criteria.

Creating Search Filters - the Search Filters are used to determine if the products in the object require approval.

To configure Search Filters for Approvals

1. Go to **Setup > Build > Create > Objects** and select **Search Filter (Approvals)**.
2. From the **Custom Fields & Relationships** section, select **Business Object**.
3. From **Picklist Values**, click **New** and enter the API names for the Header object and the line items: `Header Object API name` for Header level approvals and `Child Object API name`for Line Item level approvals.
4. Click **Save**. The values are added to the **Picklist Values list**.
5. Click **Back to Search Filter (Approvals)** and from the **Custom Fields & Relationships** section, select **Filter Type**.
6. From **Picklist Values**, click **New** and enter `Child Filter` and `Entry Criteria`.
7. Click **Save**.

The values are added to the picklists, which are required to create search filters.

You can create *Search Filters (Approvals)* for controlling which objects are checked to see if they need approval.

To create Search Filters for Approvals

1. Click **All tabs** and select **Search Filter (Approvals).**

2. Click **New Search Filter (Approvals)** and select one of the following options.
    - **Header Object Name** - select this if you want the filter to be at the context object level.
    - **Child Object Name** - select this if you want the filter to be at the Line Item context object level, which are the individual line items within the Header object.
3. Click **Next**. The Configure Filter page appears.
4. From **Filter Type**, select one of the following options.
    - **Child Filter** - This enables you to have step entry criteria within your approval process that reference the child record fields of the main context object. This enables you to use line item fields with your Header Object Name approval process. The line item child Filters you create here enable you to have them available for approval step entry criteria.
    - **Entry Criteria** - Indicates that the search filter will be used for the Approval Required check, which checks whether the header or header line item values of an object required to undergo an approval. Essentially, you have and an additional level of entry criteria for approvals here, before the system searches for matching approval processes.

    > **Note**
    >
    > You must have set at least either the Entry Criteria or the Child Filter for approvals to be available from the parent object.

5. Enter a required **Filter Name** and **Sequence**. The sequence is used to determine the order in which the search filters will check if approvals are required.
6. Enter the **Filter criteria**, which is used to determine if the object requires approval.
7. Click **Save**.

The Search Filter is now active. If it is an Entry Criteria filter, the system will determine if a header object requires approval.

You must create an Approval Process that can be used with the header object.

Approval Process for Custom Objects

Conga Approvals offers the ability to setup approval processes that are currently not supported by Salesforce. It allows the customer to configure their various approval processes with greater flexibility and visibility, helping in overall compliance.

Approval steps assign approval requests to various users and define the chain of approval for a particular approval process. Each approval step specifies the attributes a record must have to advance to that approval step, the user who can approve requests for those records, and whether to allow backup assignees of the approver to approve the requests. Regardless of their complexity, when anyone in the process rejects an item, the entire process is considered rejected.

- Creating the Process Definition
- Checking the Approval Status of the Document
- Creating Initial Submission Actions
- Creating Final Actions
- Creating Approval Steps
- Setting up Auto-escalation
- Creating Expressions
- Optional Configuration

Performance and Scalability

Area	Detail
Multiple Approval Processes	**When to use multiple approval process?** When you have Region Based Approval variations, Business Unit Based Approvals variations, Channel Based Approvals variations. **How do I configure multiple approval processes?** Define mutually exclusive process entry criteria so that only one process is selected for a given primary object instance. **Why do I configure multiple approval processes?** Having multiple approval processes reduces number of approval steps / criteria system needs to execute
Approval Rule Entry Criteria	Specify minimum approval conditions in Approval Rule entry criteria for Sub Process or Child Process steps to avoid evaluation of all approval rule entries every time.
Approval Required Check	Provides users visibility to whether approval is needed for header or line items without executing the entire approval process. Provides a mechanism to avoid approval process evaluation for a large number of scenarios.

Types of Approval Steps and when do I use them?

- **Standard** - This is the existing method for creating header level approval steps.
- **Sub Process** - This is used when you want to select approval rules that are associated with the header level context object of the approval process. The assignee types are derived from the Approval Rule. Create a Sub-Process when you have multiple approvers for a set of criteria instead of defining multiple Standard steps for each criteria.
- **Child Process** - This is used when you want to select approval rules that are associated to child objects of the approval process header level context object such as you want to define an approval process for Proposal and Proposal Line Items. The assignee types are derived from the Approval Rule. Use a child process instead of a Subprocess and Standard step type when you want to populate the Approval request with the child object line items to be approved, such as proposal line items.

It is recommended that you use conditional criteria instead of Approval Dimensions and Approval Matrix while creating your approval rules and processes.

Licensing Intelligent Approvals for Custom Objects

For each user that needs to preview or submit approvals for a custom context object, they must have a license. This is handled through the *Conga Approvals Custom Management* package.

To license users

You must have installed the license package.

1. Go to **Build > Installed Packages** and click **Manage Licenses for Conga Custom Approvals Management**.
2. Click **Add Users** and search for who you want to license to use approvals with custom objects.

All users in the Licensed Users list can use Approvals with Custom Objects.

Configuring a Custom Approval Action for Non-Salesforce or Conga profiles

The Approval Action that is available by default as part of the Approval Request object, contains references to default Salesforce and profiles which your organization may not use. If you decide to create a custom object to use instead of the default Approval Action, you must ensure you handle `$User.id` correctly when creating the formula.

`$User.id` is limited to 15 characters, while some of the fields you compare it to may contain 18 characters. To accommodate this, you must ensure you concatenate any fields that are compared to `$User.id`, such as in the following example:

```
(OwnerId=$User.Id Left(Apttus_Approval__Assigned_To_Id_c,15)=$User.Id)
```

Appendices

Limitations and Exceptions

Supported Mobile Browsers for Approvals

Platform	Supported Actions
SF 1	IWA is supported on all device browsers certified by SFDC and only on CPQ approvals pages on SF1. Support for Approvals and MyApprovals pages for CPQ only on SF1. Approve or Reject with comments, and Take ownership actions only.

Windows 10 browser support

Pages	Chrome	Firefox	IE	Edge
IWA Admin like create approval process	Supported (v 49.0)	Supported (v 45.0)	Supported	Supported (v 25.0)
ngFlow	Supported (v 49.0)	Supported (v 45.0)	Supported	Supported (v 25.0)

For supported Salesforce browser editions, refer Supported Browser Versions.

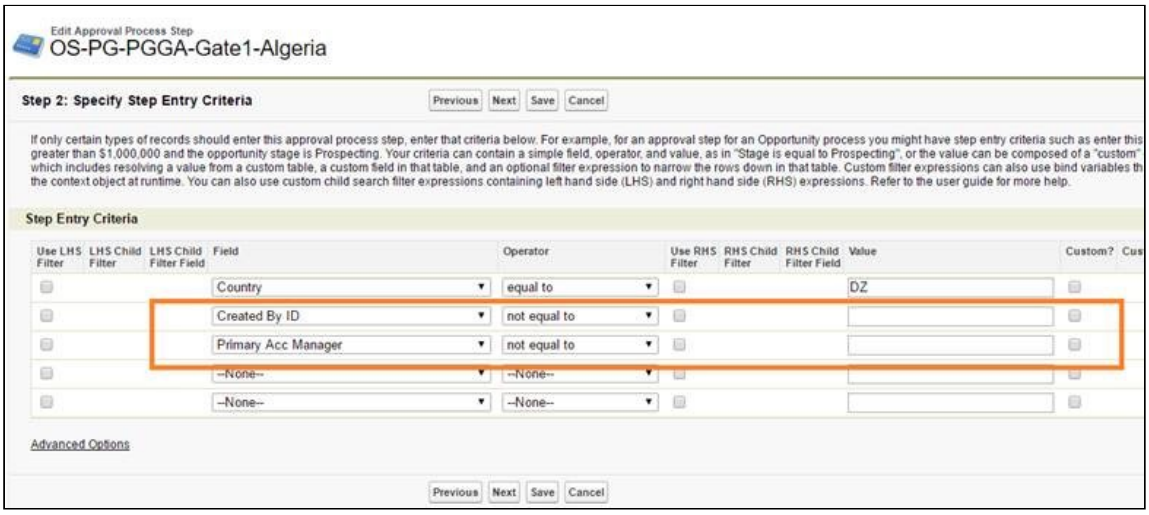
Handling a large number of Approval Rules

If an org has a large number of approval rule item records, and the approver is set as a formula field, the system cannot recognize the related user ID due to a large number of approval rules and Approval rule entries defined on the opportunity screening object. The approver cannot preview or submit the records.

Workaround

Define the related users in the rule entry or approval process step entry. for example. *Created By_ID not NULL*. The system is able to pick these values up and resolve it to the required assignee.

Update both related user values in the 1st step of the approval process as described below. If there are any more related users they should be defined on an approval process step (or in an approval rule main entry criteria). preview and submit the records.



Lightning and SF1 limitations

- Lightning Experience Enable_Custom Buttons are not displaying on Agreement Summary Page.
- Lightning Experience Enable_Activate and Deactivate button is not working for Backup-Delegate Approver.
- Lightning Experience Enable_Blank Screen Display error when you click "Save" after Agreement Creation.
- Approval Rule name modification does not reflect on UI when Lightning Experience Feature is enabled.
- Lightning Experience Enable_Manage Entry Button does not display after Approval Rule Creation.
- SF Feature: Lightning: Cancel button does not work in the Approvals related list.
- Lightning Experience Enable_Custom Buttons does not display on Proposal Summary Page.

Supported SOQL Limits

When your quote for which you trigger header and line item approvals contains a certain number of line items, you may face an SOQL limit error.

Suppose a quote comprises three level of approvals, and comprises 20 line items to be approved. an SOQL error is thrown.
An SOQL Limit is reached in the following scenarios.

Approval process	Number of line items supported
Total Number of Approval Requests supported per unique user	Up to 35 Approval Requests
Total number of line items supported that **need approvals**. (Assuming each line item will need all 5 levels of approvers in sequential format)	Up to 35 line items
Total number of approval requests supported per quote	Up to 225 Approval Requests
Total number of child process (line item level) approval requests supported per quote	Up to 175 Approval Requests
Total number of approval requests assigned to the user after which the user can Reassign or add an Ad hoc approver successfully.	Up to 40 requests.
Recall and Resubmission limits	35LIs with 5 level approvals in a sequential format. 4 levels are approved. Recall the quote, SOQL error on Resubmit.

Upgrading Intelligent Workflow Approvals From an Older Version to 7.2

When you upgrade the IWA package from versions lower than 7.2.370, ensure that you do not use the ChangePendingProcess query parameter in the Preview & Submit Approvals button as described on Approvals and My Approval Pages. Use the Enable Resubmit check box instead as described in Mandatory Configuration.

When you upgrade the IWA package from versions lower than 7.2.350 to the latest version, ensure that you make the manual changes listed in this section. Ensure that you make the changes after you install the IWA 7.2 package. To use the 7.2 features and capabilities ensure that you execute all the upgrade steps.

Inflight Approvals - Group-level approval status for Sub-Process and Child Process approvals will be blank for the records in which approvals are already in progress at the

time of upgrade. The status will be correctly displayed for the records that are submitted for approval after upgrade.

If you want to have consistency across all approval requests and want them to reflect the new enhanced approval status, run the following script and update the approval status.

> To run this script, all users must be logged off, and all custom triggers related to the child context object and Approval Request must be switched off.

Execute the following batch job using the Salesforce Developer Console (Submitter will be notified when the job completes):

Apttus_Approval.ApprovalStatusUpdateBatchJob updateJob = new Apttus_Approval.ApprovalStatusUpdateBatchJob();

ID statusUpdateJobId = Database.executeBatch(updateJob, batch-size);

> The above upgrade updates the Approval status database values as well.

Upgrading Intelligent Workflow Approvals 6.5 to 7.1

When you upgrade the IWA package from 6.5 to 7.1, ensure that you make the manual changes listed in this section. Ensure that you make the changes after you install the IWA 7.1 package. To use the 7.1 feature and capabilities, ensure that you execute all the upgrade steps.

If the assignee type is Related User in Approval Rule entry, it should refer to a User lookup field. Related user assignee is now enhanced to show the relevant User lookup fields in the picklist. The following are applicable for a related user assignee:

- Not supported: Text field for an assignee.
- Related User assignee type field value needs to be of type user lookup for the context object you create a rule for.
- If your current implementation does not include the user field as a look-up field, ensure that you define a user look up field for a related user. For more information about configuring related user look-up fields refer To configure user related fields.
- Change the existing approval rule and standard step entries to refer to the new user look up fields for assignee type Related User.

The table below highlights the picklist values you have to add to custom fields of certain custom objects.

> **Note**
>
> You must add these picklist values only if you have Apttus CPQ package installed. If your org does not use CPQ, these changes are not required.

Object	Field	Add to Picklist
ApprovalRule__c	BusinessObject__c	Apttus_Config2__LineItem__cApttus_Config2__ProductConfiguration__c Apttus_Proposal__Proposal__c
ApprovalRule__c	RuleType__c	Condition
FormulaField__c	BusinessObject__c	Apttus_Config2__LineItem__cApttus_Config2__ProductConfiguration__cApttus_Proposal__Proposal__c

The table below highlights the values you have to add to record types of custom objects for selection:

Custom Object	Record Type	Add to Selected Values in Picklist
ApprovalProcess__c	Approval Process	Context Type:Click Edit and add Single and Multiple to Selected Values.
ApprovalProcess__c	Step	Step Type: Click Edit and add Standard, Subprocess, and Child Process to the Selected values. Set the default value as Standard.

After upgrading IWA from 6.5 to 7.1, ensure that you provide appropriate step labels for the previous and existing approval processes defined. When the users click Preview or view the list of approval steps in 7.1, the appropriate step label names appear. For steps where the label value is null, the step name or the subsequent subprocess number is displayed.

To configure related user values

For the object that you want to define an Approval Rule for, ensure you create a field of type Look up relationship related to the User object. The steps below enable you to create a look up relationship for the Product Configuration object.

1. Navigate to Setup > Create >Objects > Product Configuration.

2. Click New > (Datatype) Lookup Relationship > (Related To) User.
3. Specify Field Name and click Next.
4. Select the profiles that can access the fields and click Next.
5. Select the Page Layout on which the field is visible and click Save.

When you define an approval rule for Product Configuration, and within the rule entry you select the Assignee Type as Related User, the lookup field added to the object are available in the Value drop-down list.

Related User Assignee Backward Compatibility Support

This section lists the enhancements added to support backward compatibility for a Related User.

You can assign a Related User as an approval assignee at the following places:

- Approval Rule
- Standard Step
- Auto-Escalation for a standard step

Configure a Related User in the following ways:

- To view the Related Users in a drop-down list, the Related User field should be a lookup to the User field.
- To specify the Related Users in a text field, the Related User field should be a text or a formula field.
- If Related User is a text field which requires inputs, specify the user id of the user in the text field.For defining a Related User assignee, you have to specify the API name of the field at the context object.

System Behavior when you specify a Related User:

- If the Related User field is a User Lookup field, then the system retains the selection even when the user switches from look up to formula field. However, if the Related User field is not a User lookup field, then switching the option will remove the selection.This enables the backward compatibility with Related User behavior on Approval Rule assignee prior to 7.X release.

Recommended Configurations for Intelligent Workflow and Approvals 7.1

In addition to the upgrade steps, we recommend that you configure the following when you upgrade to 7.1 or a later release.

- Approvals page: Provides an enhanced page that can be used by requestors to preview approvals, submit approval requests, view approval status, and recall approvals.
- My Approvals page: Provides a page to enable approvers to view the required approvals, approve, reject assigned items or add ad hoc approvals.
- New email notification templates: New email notification templates enables an approver to approve or reject all assigned requests by responding to an email or by accessing the My Approvals page from the email notification.
- Approval Required Check - This is a required step to use the Approvals and My Approvals pages.
- Ensure that you enable Bypass Sharing in Custom Settings > Approval System Properties.

CPQ Cart Approvals Preview from Quote

This enables you to launch a detailed approvals preview page based on the current shopping cart (Product Configuration), from the Quote/Proposal record. It includes Return, Submit, and Cancel actions.

This functionality is the same as initiating approvals from within the shopping cart; however, instead of using the product configuration you are actively working on, the most recent product configuration from the Configurations related list on the Quote/Proposal record is shown in the preview.

Configuring the Preview Page

You can use the Custom Approvals Preview page for Custom Preview instead of standard approvals preview page. This enables you to add custom information along with the information available in the standard page. Custom information sections can be added either at the top of standard screen or bottom of standard screen.

Two files are required to customize Previewing approvals: AptsCartApprovals.page, and AptsCartApprovalsController.cls. The code provided for pages and classes should be copy or pasted into the new objects in Salesforce, from where you can customize it.

Note

Please read the content of the page and class. There are warnings in the controller class that must be followed to ensure this feature works as expected.

To configure the Preview page and Controller class

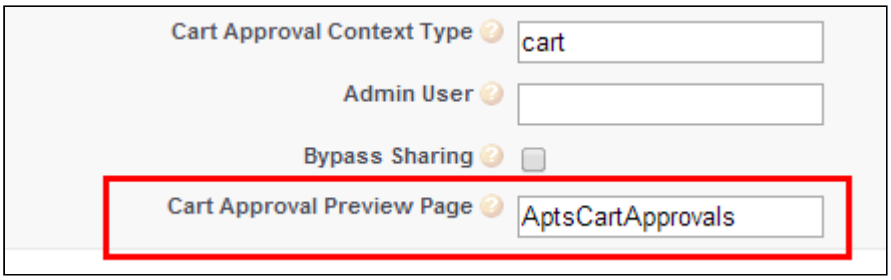
You must have created the utility classes.

1. From Setup, go to Develop > Apex Classes, click New and ensure the name of the new class is AptsCartApprovalsController. You can usefieldsetnameto include additional columns to display more information on the preview page, by default the Sequence, Step Name, and Assigned To fields are displayed.
2. Save the class and then go to Pages and click New.
3. Copy and paste the requisite code into the Visualforce Markup window:
4. Ensure the name is AptsCartApprovals and click Save.

The components required to display the preview page and enable an approval process to be submitted are ready.

You must ensure the button that will reference the AptsCartApprovals Visualforce page is displayed on the Quote/Proposal page. With a new installation this should happen automatically, but if you are upgrading you may need to add the button to the Page Layout.

After you have created a custom page, to use it, go to Develop > Custom Settings > Approvals System Properties and reference it using the Cart Approval Preview Page field.



Quote/Proposal Package

To be able to associate *Quote/Proposals* or *Proposal Line Items* with Intelligent Workflow and Approvals, you must install the Apttus Quote/Proposal-Approvals Management package.The Intelligent Workflow and Approvals package you install to get Intelligent

Workflow and Approvals feature functionality, works with *Opportunities*, *Agreements*, and *Agreement Term Exceptions*. The Quote/Proposal package provides all the required components, such as email templates and custom links, to use *Quote/ Proposals* and *Proposal Line Items*.

Migrating Approval Data

This section outlines the migration of Approvals the source (Dev/Test) to any other target (Dev/Test/Prod) environment.

Tools used for Migrating Data

Dataloader.iMio (to export/ import data with reference ids) www.dataloader.io

Naming convention

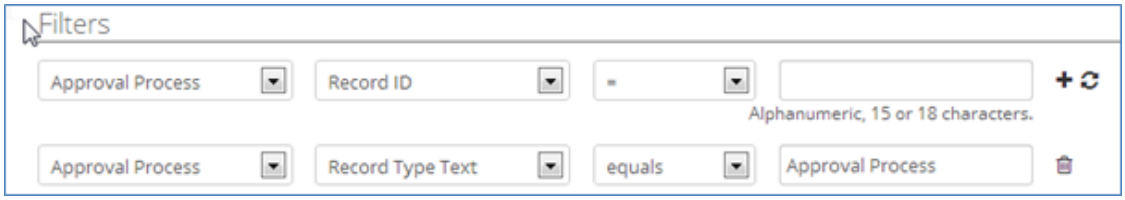
- Source Environment > SrcEnv
- Target Environment > TrgEnv

Execute the following steps

1. Ensure all other metadata is migrated beforehand. For example, any new fields created on Quote/Agreement objects which are used in Approval Process.
2. Create External ID fields for the objects, if not created.Text field with External Id flag checked, give any appropriate name.
    a. Approval Process
    b. Approval Rule Dimension
    c. Approval Rule
    d. Approval Rule Entry
3. In the object Approval Process create a formula field Record Type Text with return type Text. Formula : RecordType.Name.
4. For all this objects, copy Id for each of the records into External ID Fields in the SrcEnv. For this you will have to export the data from SrcEnv, update the data in the CSV and import the data back into the SrcEnv. For example, ID and External ID should be the same for all these records in source environment.

Exporting Data

- For Object Approval Rule Dimension export all records from SrcEnv.
- For Object Approval Rule export all records from SrcEnv.
- For Object Approval Rule Entry export all records from SrcEnv
- For Object Approval Rule Assignee export all records from SrcEnv
- For Object Approval Process from SrcEnv (*copy paste the filter values*)
    - Export records with filter Record Type Text = Approval Process and a specific Process Name.



1. Manually update Backup Admin field in the CSV with the user id from TrgEnv.
    a. Export records with filter Record Type Text = 'Entry Criteria'
    b. Export records with filter Record Type Text = 'Initial Submission Action'
    c. Export records with filter Record Type Text = 'Final Action'
    d. Export records with filter Record Type Text = 'Step Group'
    e. Export records with filter Record Type Text = 'Step'
2. Blank out all data from Step Assignee ID column in CSV.
    a. Export records with filter Record Type Text = Step Filter Criteria.

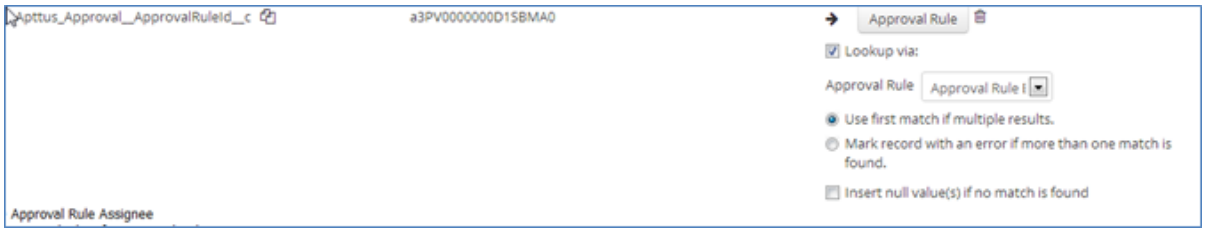
There are **11** CSV files in total.

Importing Data

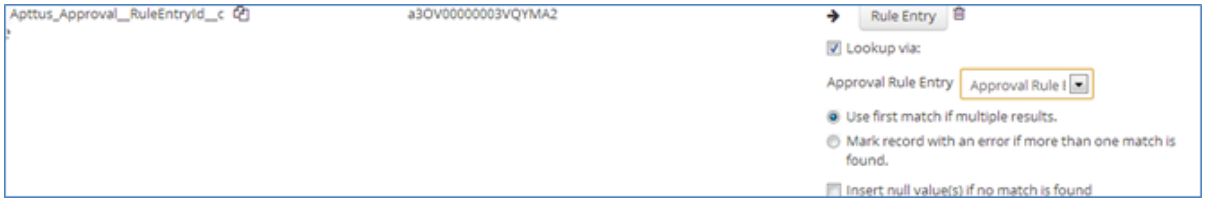
> ⓘ Before importing data it is recommended that you sort the spreadsheet by record name (for advanced logic purposes).

1. Import 'Approval Rule Dimension' csv in TrgEnv. No Lookups are required.
2. Import 'Approval Rule' csv in TrgEnv.
3. Use lookup to External ID for all **dimension** fields.

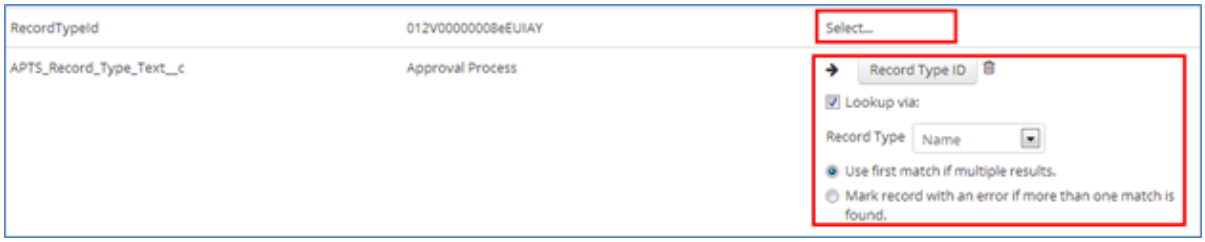
4. Import the Approval Rule Entry CSV in TrgEnv.
5. Use lookup to External ID for **Approval Rule** field.



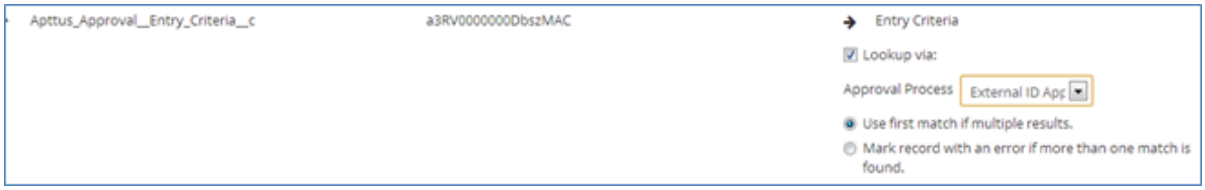
6. Import the Approval Rule Assignee CSV in TrgEnv.
7. Use Lookup to External ID for **Rule Entry** Field.



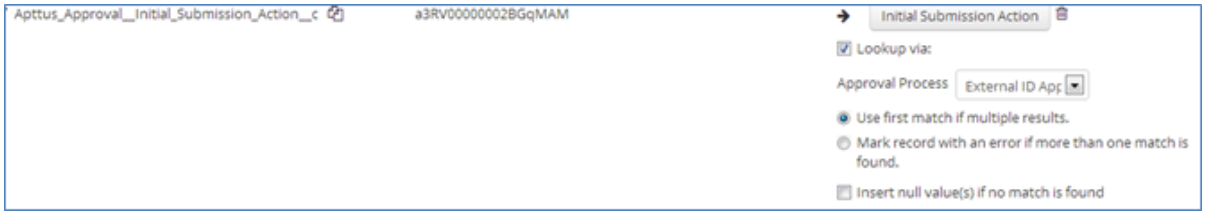
8. Import Approval Process CSV in TrgEnv.
9. Remove Mapping for RecordTypeId
10. Add Mapping to the custom field APTS_Record_Type_Text__c, to Record Type ID, Lookup via: Name.



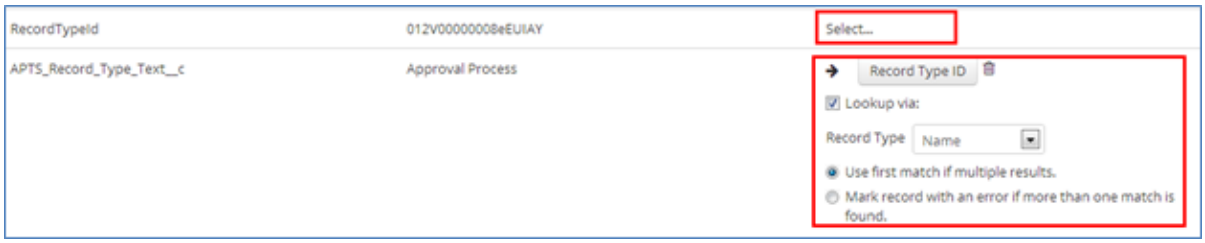
11. Import the Entry Criteria csv in TrgEnv.
12. Use the lookup to External ID for **Entry Criteria** field.



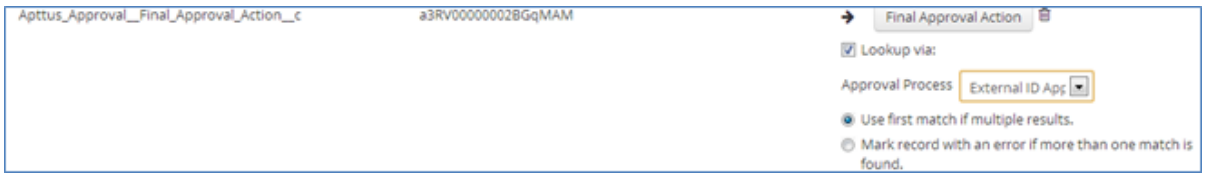
13. Remove the mapping for RecordTypeId
14. Add the mapping to custom field APTS_Record_Type_Text__c, to Record Type ID, Lookup via: Name
15. Import Initial Submission Action csv in TrgEnv.
16. Use the lookup to External ID for **Initial Submission Action** Field.



17. Remove Mapping for the RecordTypeId
18. Add Mapping to the custom field APTS_Record_Type_Text__c, to Record Type ID, Lookup via: Name
19. Import Final Action csv in TrgEnv.



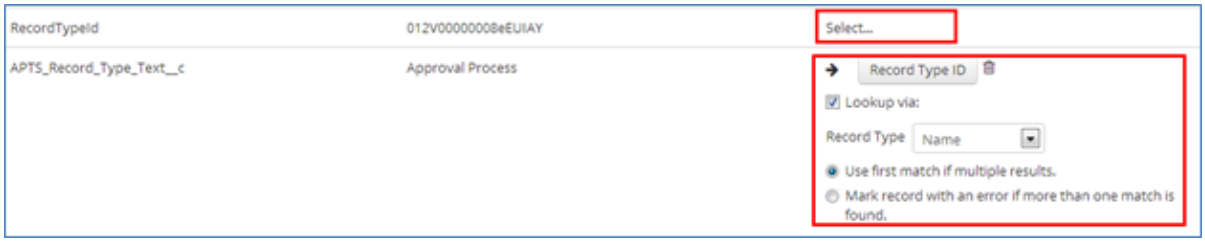
20. Use the lookup to External ID for **Final Approval Action** Field.



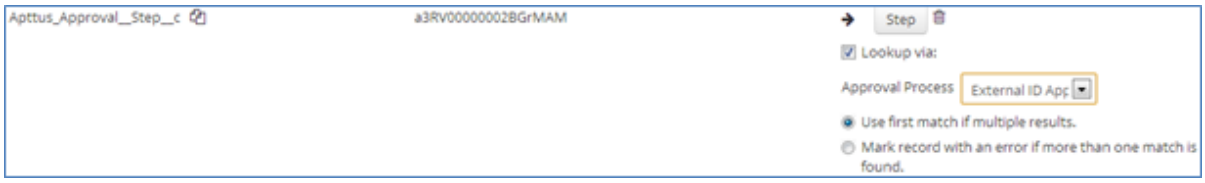
21. Remove Mapping for RecordTypeId.
22. Add a mapping to custom field APTS_Record_Type_Text__c, to Record Type ID, Lookup via: Name.
23. Import Step Group csv in TrgEnv.

24. Use lookup to External ID for **Step Group** Field.

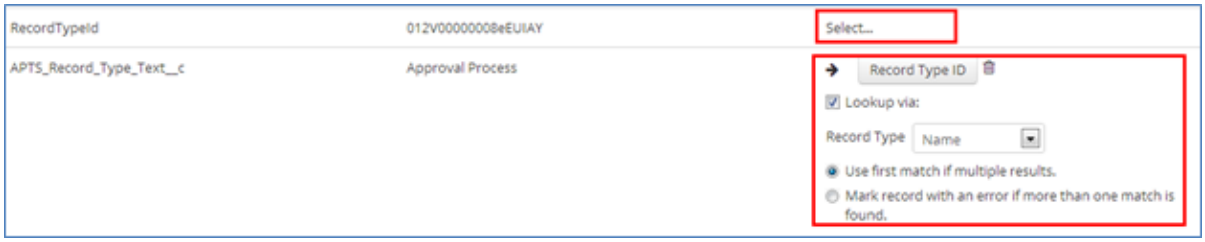
25. Remove the mapping for RecordTypeId.
26. Add the mapping to custom field APTS_Record_Type_Text__c, to Record Type ID, Lookup via: Name.



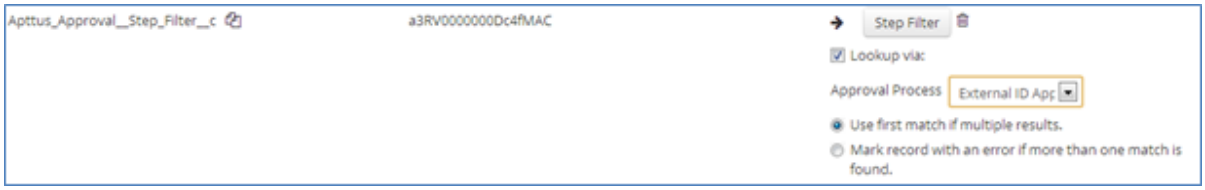
27. Import 'Step' csv in TrgEnv.
28. Use Lookup to External ID for **Step** Field.



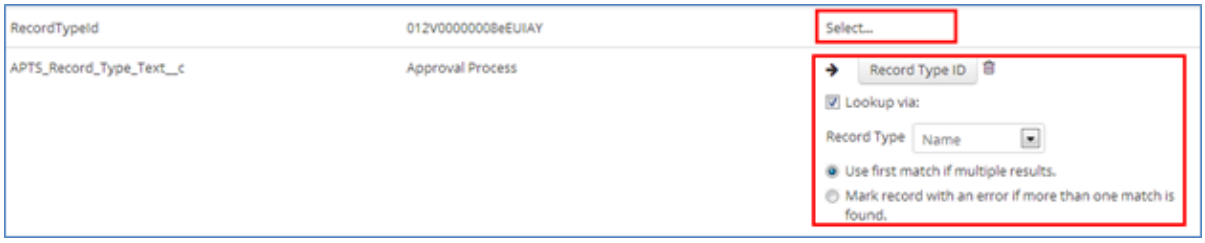
29. Remove the Mapping for RecordTypeId
30. Add the Mapping to custom field APTS_Record_Type_Text__c, to Record Type ID, Lookup via: Name



31. Import the Step Filter Criteria csv in TrgEnv
32. Use Lookup to External ID for **Step Filter** Field.



33. Remove Mapping for RecordTypeId.
34. Add Mapping to custom field APTS_Record_Type_Text__c, to Record Type ID, Lookup via: Name.



There is an issue with Advanced filter logic in Step Filter Criteria as there is no sequence number stored in the export files. If you are using Advanced Filter Logic, verify it manually after importing all the data.

Advanced Filter Condition
1 OR (2 AND 3) OR (4 AND 5) OR (6 AND 7) OR (8 AND 9)

For example, please also note if the user ID is not the same across the orgs, it may cause issues, and the user ID might have to be populated manually.

FAQs

- How to determine version numbers?
- How to find the merge server end point?
- How to determine the salesforce.com organization ID?
- How to grant login access?

How to determine version numbers?

Login to salesforce.com with the affected login credentials.

1. Go to **Your Name** > **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.

How 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** and the **Merge Webservice Endpoint** setting is displayed. The *https://mergeserver.apttus.net:9876/* portion of the setting is what will be helpful to customer support.

How to determine the salesforce.com organization ID?

1. Go to **Your Name** > **Setup** > **Administration Setup** > **Company Profile** > **Company Information**.
2. From the Organization Detail pane, provide the Salesforce.com **Organization ID**.

How to grant login access?

1. Go to **Your Name** > **Setup** > **Personal Setup** > **My Personal Information** > **Grant Login Access**.
2. From the **Apttus Support** picklist, select an option for access duration.
3. Click **Save**.

Glossary of Approval Terms

Approval Matrix	Approval matrix determines the assignees based on given criteria. For example, approver authority may change based on the discount being offered on a sales contract. So different levels of approvers can be assigned based on approval authority on a given item.
Approval Process	The process definition that describes approval processes within your organizations. You can define approval processes for Opportunity, Agreement or Term Exceptions objects. You have to define entry criteria, initial submission actions, step groups (steps, step filters) and final actions within an approval process.

Approval Rule (Rule)	Approval rule uniquely identifies an approver based on combinations of various logical conditions per business process and policies. For example, an agreement needs to be sent for approval to CFO if contract value is above $1M OR (geography is emerging markets AND contract value is above $250K).
Approval Summary Page	When an approver clicks on the link of an approval notification email, he/she is directed to an approval summary page where he/she can approve or reject that item. This page contains important business information for the person to make decision about approval. This page may be configured per your business requirements.
Assignee Type	The system supports the following: Approval Matrix, Auto Approval, Custom Queue, Custom Role, Custom Rule, Related User, Custom User, Queue, Role, Rule, User
Assignment Email Template	Email template used when an item is reassigned to another user. When an item is assigned to a user for approval, he/she always has an option to reassign the item to another user.
Auto Approval	This Assignee Type can be selected for Approval Rule Entry Criteria when you want the item in an approval process to be automatically progressed. For this to work, in an Approval Process step which references that Approval Rule using Auto Approval, the **Assigned Approver > Assignee Type** must be *Subprocess*.
Auto Complete	Selecting Step Auto Complete checkbox denotes that this step may be auto completed without taking any manual approving actions. This functionality is most commonly used when submitter is also an approver and does not need to manually approve this agreement since its already reviewed it before submission for further approvals.For auto complete to work as expected, ensure that the step you set **Set Auto Complete** for is independent from another step.
Backup Approvers	When a user is out of office, all approvals can be automatically assigned to backup approvers. The user may choose to reassign in process approval requests to backup approvers as well.

<table>
<tr><td>Cancellation Email Template</td><td>Email template used to notify that an approval process has been cancelled.</td></tr>
<tr><td>Consolidated Approvals</td><td>If some approval process results in a situation where the same user is required to approve the same object record/line items, these multiple approvals can be consolidated in one approval. In other words, the approver has to approve or reject it only once to approve or reject all of their approval requests for a single approval process.

Consider the following approval process with three steps:

<table>
<tr><th>Approval Step</th><th>Approver 1</th><th>Approver 2</th><th>Approver 3</th></tr>
<tr><td>Sales Level 1</td><td>John</td><td>Mary</td><td>Dave</td></tr>
<tr><td>Sales Level 2 (Dependent on 1)</td><td>James</td><td>John</td><td>Suze</td></tr>
<tr><td>Executive Sign-off</td><td>Art</td><td>-</td><td>-</td></tr>
</table>

When the object record is submitted for approval and the approvers are selected based on the criteria in the approval process, the approver John is included in two steps, including one dependent step.

Once Mary and Dave have approved their requests for the Sales Level 1 step, John's approval request in the Sales Level 2 step will be Assigned to him, while his approval request in Sales Level 1 is on Hold. This enables him to approve both of his approval requests via a single action.</td></tr>
<tr><td>Consolidate Notifications</td><td>If some approval process results in a situation where same user is required to be notified for the same item, these multiple notifications can be consolidated in one notification. In other words, the approver has to approve or reject it only once to approve or reject all consolidated approvals.</td></tr>
</table>

Custom Assignee	Custom assignee option allows you to configure custom code to be called to evaluate an assignee. If none of the other options are meeting the assignment requirements, custom code can be developed to evaluate the approver based on given criteria.
Custom Queue	This assignee type points to a field which contains a queue name in any custom object with a filter to narrow it down to a single row. i.There are two ways to specify a filter: 1. Example 1: Select the row where the 'Region__c' field has a value of 'Asia'. 2. Example 2: Select the row where the 'Region__c' field has a value which equals the value of 'MyRegion__c' field in the associated business object
Custom Role	This points to field which contains a role name in any custom object with filter to narrow down to a single row. For example: Custom_Object__c.Approver_RoleName__c
Custom User	This points to a user lookup reference field in any custom object with filter to narrow down to a single row. The custom object may have filter criteria enclosed in parenthesis like the Region filter in this example: Custom_Object__c(Region__c = '**Asia**').Approver_UserId__c
Depends On	This field defines the dependency of a step on other steps. For example, if a 'Level 3' step depends on Level 1 and Level 2, Level 3 approver will receive the item only after getting approved by Level 1 and Level 2 approvers. These dependencies automatically define these approval processes to be 'sequential approval processes'. The absence of dependencies may mean that the approval processes are parallel approval processes.
Dimensions	Dimension allow you to map fields from a business object such as Opportunity, Agreement Term Exception etc and may be used in dimension-based selection.

Email Approval Response	The email approval response feature gives users the ability to approve or reject email approval requests by replying to the email. The first line of the email body may contain one of the following words: • approve • approved • yes • reject • rejected • no Periods and exclamation marks are also accepted at the end of the word. You can also optionally add comments in the second line of the email body. Users can still click a link in the email to access the approval page as well. This feature is especially useful for organizations with users who receive approval requests on mobile devices.
Entry Criteria	You can define certain logical conditions as part of entry criteria for an approval process. If these conditions evaluate true, the approval process is evaluated. This criteria allows you to make sure that all agreements (or opportunities) do not process through given approval process.
Field Update (Constant)	If you select 'Field Update (Constant)' as Initial Submission Action Type or Final Approval Action in an approval process, the system would update the given field with a constant value. For example, setting the field to fixed status type. For example, changing the Agreement Status field to 'Pending Approval'.
Field Update (Value)	If you select 'Field Update (Value)' as Initial Submission Action Type or Final Approval Action in an approval process, the system would update the given field with a value, given by the field. For example, setting the approval submitter to current user ID (CurrUserId).
Final Approval Action	The action which need which will be performed on the record whenever the record is approved in final stage.
Initial Submission Action	The action which needs which will be performed on the record whenever the record is submitted for approval.

Notify Only	Selecting Step Notify Only checkbox designates this step as a notification only process step. Only notifications will be sent for information purpose only and no approvals will be sought as part of this step.
Notify Only Email Template	Email template used when a user has to be just notified as part of the process. No approval is requested as part of this email template.
Parallel Approval Process	This occurs when there are multiple approval steps in the same approval process and no step dependencies have been created.
Queue	You can assign the agreement to a queue (a collection of users) instead of a particular user, e.g. you can create a queue CXO in which to include CEO, CFO and COO. Assign the final approval to this queue, allowing any one of these members to approve the agreement.
Reassignment Email Template	Email template used when an item is reassigned to another user. Either specifically delegating to another user or automatically reassigned to a backup approver if the originally assigned user is out of the office.
Reassign	When an item is assigned to a user for approval, he/she always has an option to reassign the item to another user.
Related User	This points to a user lookup reference field in the associated business object (i.e. Agreement or Opportunity). For example: Approver_UserId__c
Role	You can assign an agreement to a group of users who has a particular role. Any user from that group can approve the agreement at that level. For example, an agreement can be approved by any user who has role of 'Finance Approvers'.
Sequential Approval Process	An approval process is a sequential approval process when multiple steps are involved in a given approval process and they have to be processed in defined sequence. This is dictated by 'Depends On' field, i.e. presence of step names in 'Depends On' makes given process sequential approval process. And absence of the step names in 'Depends On' field signifies that as a parallel approval process.

Sequence Number	Indicates the order in which approval actions are run.
Step	A step is a business action of approval which will be performed by a user to approve or reject the record.
Step Assignee Type	The type of assignee to which the record will be assigned for approval.
Step Notify Only	The check box will be used if the user want to notify the step assignee about the approval process.
Step Filter	After a step is created and saved, step filters need to be created to define the filter criteria for given approval process.
Related User	A user lookup field in the context object (i.e. Agreement) or in its relationship path. This is useful when the approver data is available on the record.
Transfer In-Flight	If Transfer in-flight checkbox is checked, all in-process approval requests will be automatically transferred to backup approvers on effective date. This is useful if user prefers to hand over current approval activities to backup approver. If this is left unchecked, in-process approvals will not be transferred to backup approvers and main approver is expected to respond after coming back to office.
User	This assignee type refers to a named user of the system. Use this assignee type when you want to dynamically assign an approval request to a user.

Approvals for Users

This section describes how Conga Approvals works and how to trigger an approval request for any object and send an email notification to the concerned stakeholders for approval.

Topic	Description
What's Covered	This section provides users with information on approval processes. It also covers the most common use cases for users and assumes a level of familiarity with basic Salesforce.
Primary Audience	Sales representatives, End Users
IT Environment	For information pertaining to the requirements and recommendations, see System Requirements and Supported Platforms Matrix.
Updates	For a comprehensive list of updates to this guide for each release, see the What's New in Approvals Documentation topic.
Other Resources	• See Approvals for Administrators for basic admin tasks and end-user experience. • See Approvals for SOAP API Developers for SOAP APIs provided to work with Approvals objects.

This section describes the following tasks:

- Approvals Functional End User Workflow
- Setting up Approval Process
- Approval Center

Before using Approvals, 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:

- Logging in to Approvals
- Approval Functional End User Workflow
- Creating Approval Processes
- Setting up Approval Rules

Logging in to Approvals

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

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

Before logging in to Approvals, make sure you meet the following criteria:

- You have installed all required Approvals packages (included packages for other integrated Conga applications).
- You have administrative privileges.
- You have login credentials provided by Conga.

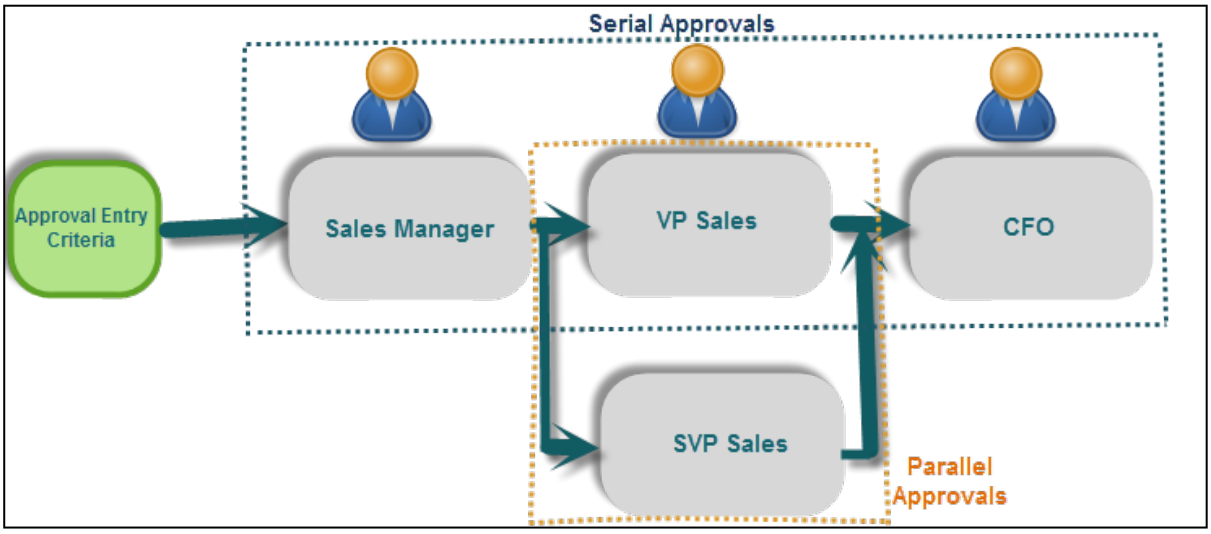
To log in to Approvals

1. Go to http://www.salesforce.com.
   Or
   If your organization is using a sandbox or test environment to access Conga Approvals (for example, if you are doing user acceptance testing), go to https://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 the Conga Approvals:
   - In Salesforce Classic: Click the App Menu and select **Conga Approvals Management**.
   - In Salesforce Lightning Experience: Click the App Launcher and select **Conga Approvals Management**.

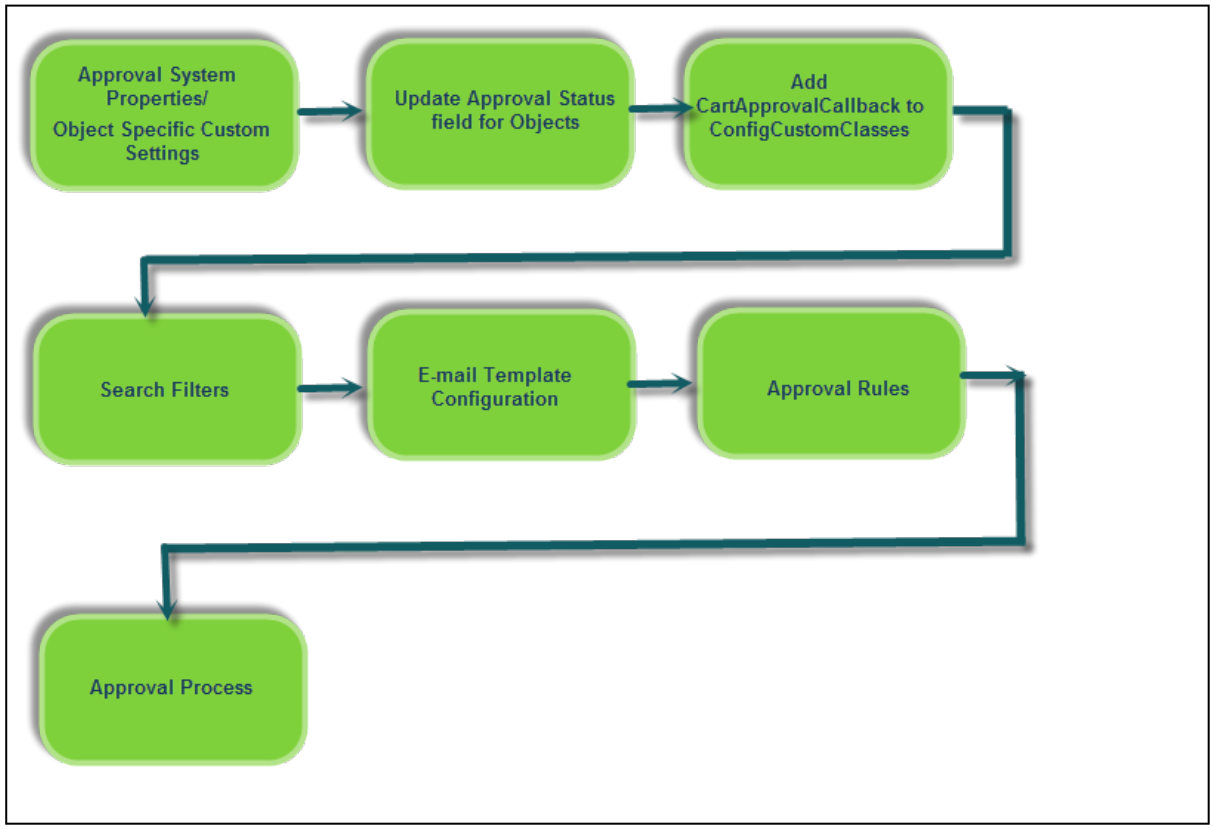
> For using Lightning Experience in your Salesforce org, you must enable My Domain in your organization. To know how to enable My Domain in your org, refer to Salesforce Help page My Domain.

Approval Functional End User Workflow

Conga Approvals enables you to create Approval Criteria, such as Discounts, based on which Approvals are triggered. You can use Approvals in scenarios where you want to trigger an approval before a quote or agreement is finalized. You can also trigger approvals for custom objects based on certain criteria. Approvers can also dynamically add an approver based on their discretion.



CPQ Workflow



Approval Center

Approval Center provides a holistic view of all your approvals. At a glance, you can see the number of approval requests assigned to you, the number of approved requests, and rejected requests. You can also click the Knowledge Performance Indicator (KPI) tiles to filter out the approvals. If you click the tile *Pending Approval,* only the approvals with Status as Pending Approval are shown. Approval Center displays approvals for Agreements, Product Configurations, Opportunities, and custom objects. To work with custom objects on the Approval Center, ensure that administrators have configured custom objects for approvals. For information on configuring custom objects, refer to Configuring Approvals for Custom Objects.

Important

To use Approval Center, you must install the **Conga Approval Center** and **Conga Grid** into your Salesforce org. **Conga Grid** is a prerequisite. Note that Conga Grid can only be installed and used for Approvals Center capabilities in Approvals.

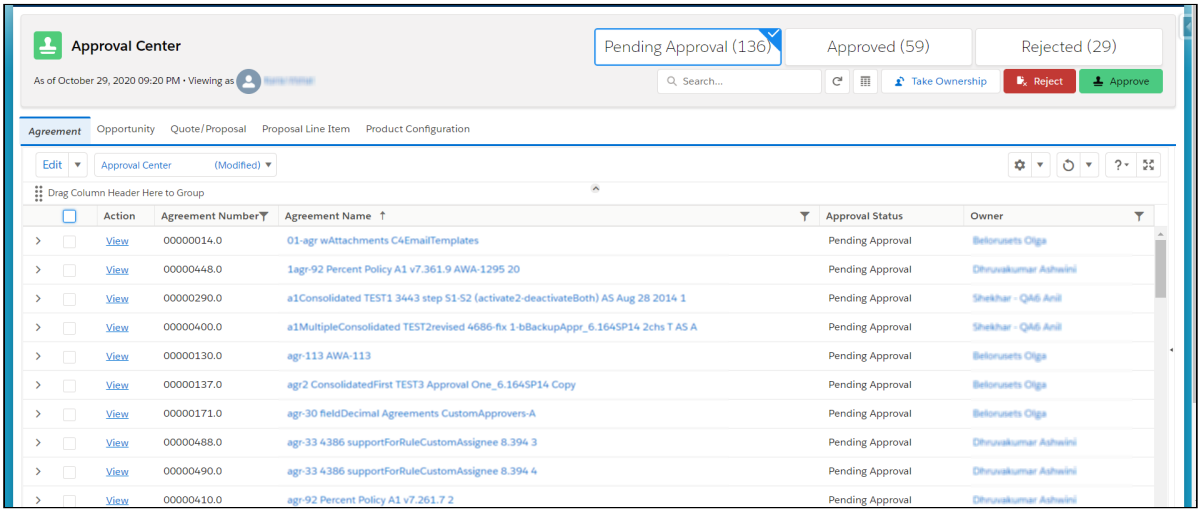
Accessing the Approval Center

Prerequisites for Accessing the Approval Center

- Access Permissions:
  Administrators need to provide access rights before you can access the Approval Center. For more information, refer to Permission Sets for Accessing Approval Center.
- Adding Objects to Approval Center
  Administrators need to add the API name of context objects in Approval System Properties → **Dashboard Objects** setting to view the object tab on Approval Center.

Navigating to Approval Center

You can access the Approval Center by navigating to **All Tabs → Approval Center.**

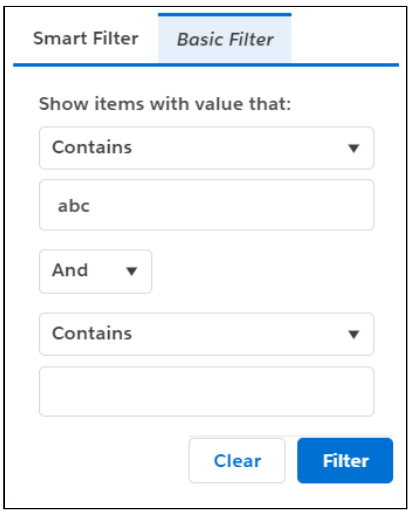


Using the Approval Center, you can:

Search for Approvals

Approval Center provides global search as well as field search. You can use the search bar located below the KPI tiles to search for approvals based on a keyword. You can also use the smart search to search for a keyword in a particular field. For example, if you want to search for an agreement name containing the keyword, ABC. You can achieve this by

clicking the filter icon next to the column, **Agreement Name,** and entering ABC in the smart filter search bar.



Approve Requests

You can select from multiple approval requests assigned to you and approve them simultaneously. To approve requests,

1. Select your approval requests.
2. Click Approve.
3. Add your comments and Click **Approve.**

Approval requests assigned to you are now approved. For information on Approving Approval Requests, refer to Approving a Request.

Reject Requests

You can select from multiple approval requests assigned to you and reject them simultaneously. To reject requests,

1. Select your approval requests.
2. Click Reject.
3. Add your comments and Click **Reject.**

Approval requests assigned to you are now rejected.

Take Ownership

For approval requests assigned to a queue, you can take ownership of them and approve or reject them. To take ownership,

1. Select the approval requests.
2. Click Take Ownership.

The approval request is now assigned to you. You can approve or reject it.

> Under the Assigned To column, look for pending approval steps currently assigned to a Role or a Queue. You can only take ownership of a pending approval that is currently assigned to a Role or Queue. You cannot take ownership of a pending approval step that is already assigned to a specific user. You can take ownership of a request only if the approval request is applicable to you.

Group Approval Requests

You can also group and filter approval requests based on different fields and field values. You can group approvals by dragging the related column to the area above the approval columns. Grouping approval requests is useful when you are working with a large number of approval records. Consider a scenario where you have 1000 approval requests. 400 approval requests for ABC organization, 200 approval requests for XYZ organization, and 400 for PQR organization. You want to view the rejected approval requests belonging to the ABC organization. You can group the approval requests belonging to the same organization together and use filters to further narrow down the results.

Create and Save a Custom View

You can create multiple custom views and save them for later use. You can customize the view by:

- Creating new filters
- Adding or removing fields
- Grouping related agreements

After you customize the Approval Center, click the down-arrow next to the **Edit** button. Click **Save** or **Save As.** Provide a unique name for your view and save it.

For more information on creating custom views and modifying the layout, refer to Conga Grid documentation.

Approving a Request

Based on the Approval Process configuration, if an approval request is initiated and you are the assigned approver for the request, you receive an approval notification or an

email. You can choose to approve the request using your Salesforce org credentials or from the mail itself. You can approve and reject multiple requests for complex processes.

Approving a request from Salesforce: Login to your Salesforce org and from the Approval Requests tab, you can choose to approve or reject an individual request. You can also choose to approve a single request at a time.

Approving a request from your email: If an Approval request is initiated, and you are the assignee for the process, you receive an email regarding the approvals assigned to you. You can Approve or Reject a request from the email itself by:

- Use the Approve / Reject key words in the very first line of the body
- Add comment in the immediate second line
- End comments with a blank line.

> Queue email address cannot be used for notifications. If Queue has **Send Email to Members** box un-checked - email of current user will be used for notifications (not queue email address). This is due to Salesforce limitations which we cannot bypass.

Submitting a Request with Attachments

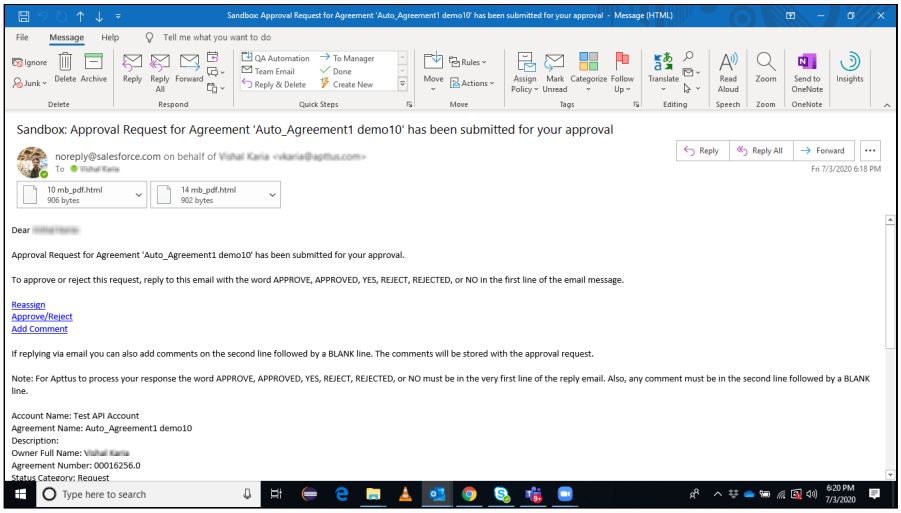
You can submit an object for approval along with an attachment. You can also submit processes with large numbers of auto approve and notify only requests. In the Approvals and My Approvals when you click Submit Approvals (with Attachments), a list of attachments appear. Select the attachment you want to submit with the approval request and proceed. The attachment is available in the Notes and Attachments section of the object.

You can choose attachments for child object records (Out-Of-The-Box or a custom object) when sending for approval. For example: Consider a custom object as a parent object and Agreement as the child object. When you submit for approval, the attachments from the child object (in this example, Agreement) is also included for approval. This feature is based on the custom setting *IncludeChildObjAttachmentsForSubmit__c* in ApprovalsCustomConfig custom setting.

To submit a request with attachment, ensure that you have created custom email notification templates and specified the name of each of the templates in the Approval Process or Custom Configuration of the object. Ensure that you have specified custom templates for Assignment, Cancellation, Escalation, NotifyOnly, and Reassignment. Even if one of the templates is not created and specified, the Submit with Attachments feature does not work.

While submitting an approval request with attachments, ensure you adhere to the file size limits defined by Salesforce. For more information about file size limits, refer File Size Limits in Salesforce.

If the combined file size of all the attachments exceed 5 MB, the attachments are sent as deep links. The recipient can use the links to download the attachments.



There are two ways to attach files with your approval requests; Notes and Attachments or Salesforce Files. By default, Notes and Attachments is used for attaching files to approval requests. To enable Salesforce Files, select the *Use Files Instead of Notes & Attachments* from the Approvals Custom Config setting. When enabled, the approvals engine works with attachments from Salesforce Files attached or uploaded from the desktop for the context object instead of Notes & Attachments related list.

IMPORTANT

You must enable or disable the *Use Files Instead of Notes & Attachments* custom setting independently for each object used for approvals.

If a custom setting for a context object is not found, the system defaults to working with files from the Notes & Attachments related list for the content object.

For example, if you are using Contract Management with approvals, enable the *Use Files Instead of Notes & Attachments* custom setting for Apttus__APTS_Agreement__c, Apttus__Agreement_Clause__c or other agreement related objects. In case of CPQ, enable the custom setting for Apttus_Config2__ProductConfiguration__c or quote proposal objects.

Submitting Version Specific Agreement Attachments

For documents with **Enable Document Versioning** set as *True* and agreements with **Version Aware** set as *True*, you can select and send a specific version of agreement documents for approval. For information on enabling document versioning, refer to *Enabling Contract Document Versioning topic in Contact Management Administration Guide.*

When selecting an attachment to submit for approval, you can select the attachment from Document Versions along with Files or Notes and Attachments:

	Version Aware	Attachments
Files	False	All attachments are available for selection from Files.
	True	If Show All Document Versions is set as False: • All attachments from Files are available for selection. • The latest version from Document Versions is available for selection from Document Version Related List. If Show All Document Versions is set as True: • All attachments from Files is available for selection. • All document versions from Document Versions are available for selection from Document Version Related List.
Notes and Attachments	False	All attachments are available for selection from Notes and Attachments.

	Version Aware	Attachments
	True	If Show All Document Versions is set as False: • All attachments from Notes and Attachments are available for selection. • The latest version from Document Versions is available for selection from Document Version Related List. If Show All Document Versions is set as True: • All attachments from Notes and Attachments are available for selection. • All document versions from Document Versions are available for selection from Document Version Related List.

To submit version-specific agreements for approvals

1. Go to Custom Settings → Approvals Custom Config.
2. Select **Show all Document Version Details.**
3. Click Save.

All the versions of the agreement documents are now available when you select the agreement attachments to send for approval. Select the version that you want to send from the Document Version Related List and click **Submit.**

If you do not set **Show all Document Verison Details** as *true,* only the latest version of the agreement attachment is available for you to select and send it for approval.

Approvals and My Approval Pages

The approvals can be previewed directly from the *Object* record.

Go to an Object record that has values configured for it and click **Approvals**.

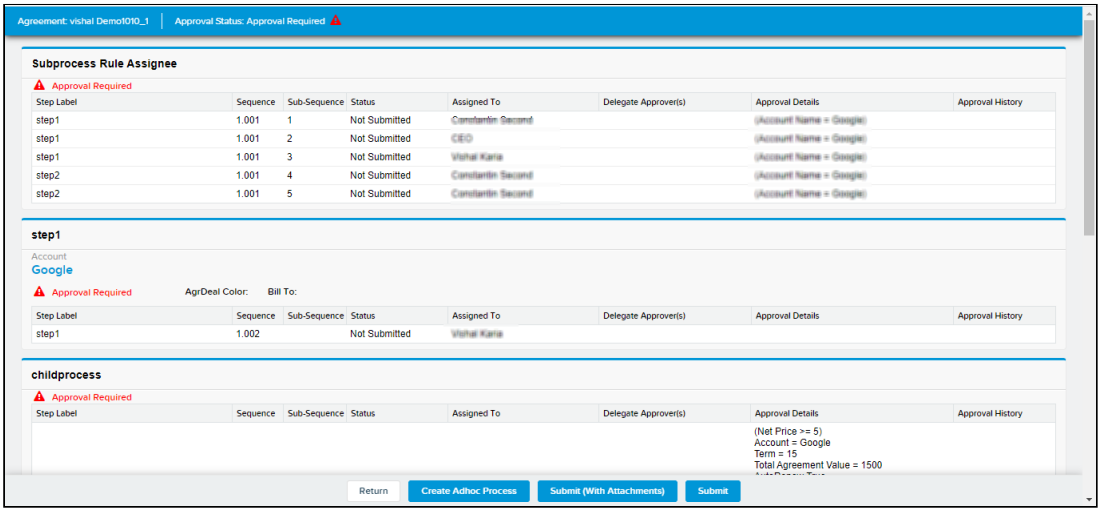
If there is an approval process having an entry criteria that matches the attributes of the object, the preview page will display the approval steps and enable you to submit the approval.

The Preview page also includes Display Fields you have selected to be shown when you created your approval step.

Approvals Page

The Approvals page is primarily used by the requester for the following tasks.

- Preview a list of all approvals that are required before they are submitted for approval. You can preview and submit approvals for complex processes
- View a list of all approvals including those that have already been submitted and approved or rejected, or are in another status such as Assigned, On Hold, or Not Submitted.
- View the Backup/Delegate Approver name.
- View approval sequence and sub sequence. The sequence and sub-sequence are captured from the Approval Request object. The sequence number denotes the sequence of approval request and the subsequence denotes the sequence of child processes.
- Allow the submitter to recall all approvals for the current context object.

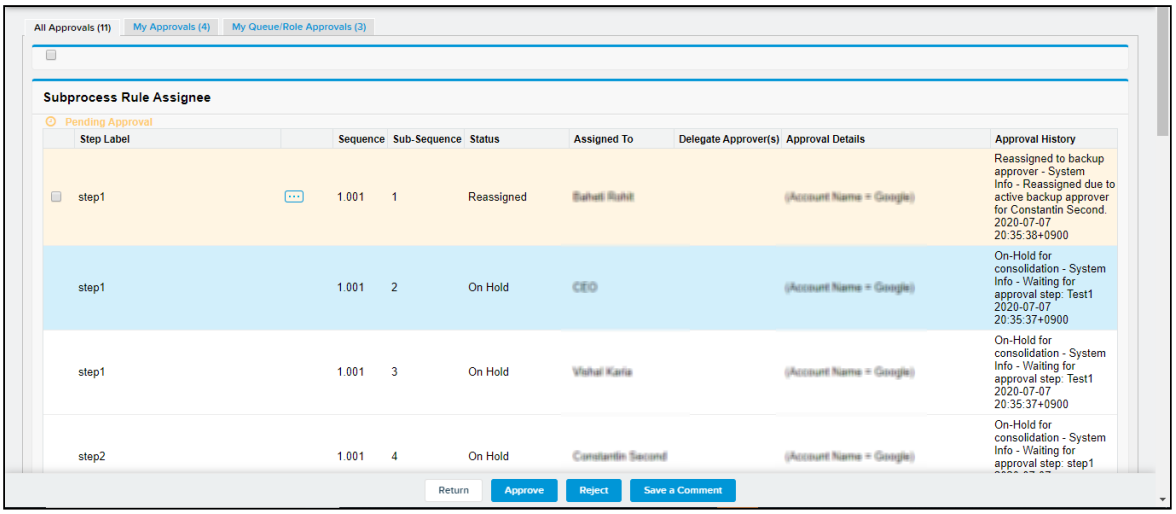


My Approvals Page

The My Approvals page is primarily used by the requester for the following tasks.

- View the list of items that need approval from the current logged in approver.
- View a large number of requests in the my approvals page.
- To take ownership of the approval items that are assigned to queues or roles, if the current user is a member of the queue/role.
- To approve or reject one or more assigned approval items.
- To add an ad hoc approver to the list of approvers.
- To remove an adhoc approver from the list of approvers.
- View approval sequence and sub sequence. The sequence and sub-sequence are captured from the Approval Request object. The sequence number denotes the sequence of approval request and the subsequence denotes the sequence of child processes.
- View backup/delegate approval name.

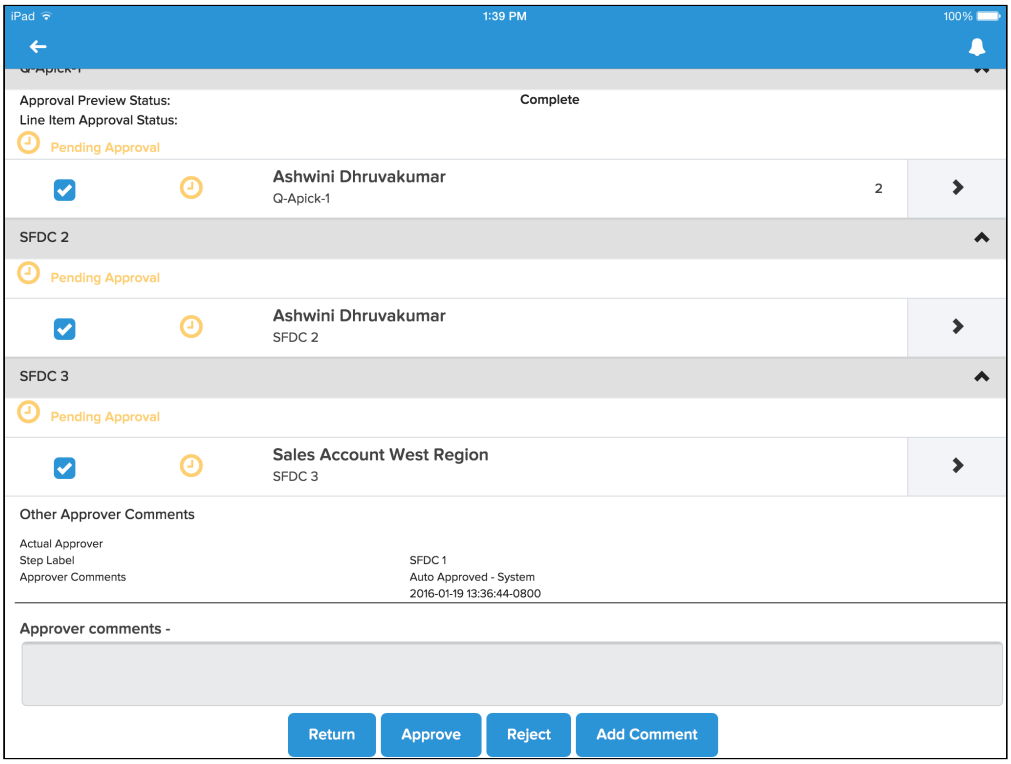
You can use the check box available at the top-left corner to select or deselect multiple approvals at once. This check box allows you to approve or reject approval requests in bulk.



My Approvals Page for SF1

The My Approvals page is primarily used by the requester for the following tasks.

- View the list of items that need approval from the current logged in approver.
- To take ownership of the approval items that are assigned to queues or roles, if the current user is a member of the queue/role.
- To approve or reject one or more assigned approval items.



> Add Adhoc Approver, Reassign approval request, View Delegate Approver features are not available on SF1.

Continue Pending Approvals on Reject

This feature enables you to continue approval processes, even if an approver rejects the approval request. Previously if a specific approval request resulted in a rejection, the progress of the entire process would halt.

This option is selected at the specific approval process level.

> **Important**
>
> Before you configure Continue Pending Approvals on Reject, ensure that you understand the behaviour of Continue Approval Policy on a Reject, since both these settings are dependent over each other.

Continue Pending Approvals on Reject Scenario

In a situation where you have a typical three step approval process, where the third step is dependent on the previous two steps, rejecting an approval request doesn't negatively impact the remaining steps.

In that instance the approval process continues as if the initial approval request was approved.

Approval Step	Approver	Status (With Continue Pending Approvals Setting Selected)	Status (Without Continue Pending Approvals Setting Selected)
Sales Level 1	Deal Desk	Rejected	Rejected
Sales Level 2	Regional VP Sales	Assigned	Canceled
Executive Sign-off (Dependent on previous two steps)	SVP Sales	Not Submitted	Canceled
Approval Status record		Pending Approval	Rejected

To set Continue Pending Approvals on Reject

1. In an approval process, go to *Step 3: Specify Consolidation Settings*.
2. Select **Continue Pending Approvals on a Reject** and save the approval process.

By default, this option is not selected.

The following lists the results on approval requests when selecting and not selecting the option.

Selected	• *Rejected* request status will be *Rejected* • Request that is dependent on *Rejected* request will go in *Cancelled* status • All other requests in *Assigned* status will continue to remain in that status • Requests in *Not Submitted* status will remain in that status • Overall object record approval status will continue to be in *Pending Approval*
Cleared	• *Rejected* request status will be *Rejected* • Rejection of any one approval request will cancel all other *Assigned* requests • Requests in *Not Submitted* status will remain in *Not Submitted* status • Overall object record approval status will be *Rejected*

You can return to the approval process later to change the option, which will subsequently impact new approval processes, but will not change the behavior of in-flight processes.

Auto Reapprovals

Auto reapproval enables you to return to a specific step in the approval process, after restarting the approval, allowing you to avoid having to make approvers reapprove requests they had already completed.

The auto reapprovals occur at the step level,when a resubmit of the approval process happens. This allows you to select where the approval process continues, at a granular level. When you have a Standard step type you configure the reapproval for the step; however, if you have a Subprocess or Child Process step type, the reapproval is configured in the referenced approval rule.

> For CPQ approvals, if the cart is approved, the cart status should be in Ready for Finalization status for auto reapprovals to work.

Using Approvals you can set auto-reapproval criteria, using which a request once approved by certain assignees are auto-approved based on the auto-reapproval criteria.Auto Reapprovals Scenario

Use case 1: Suppose your approval process comprises three steps that are dependent on each other.

- In step 1, the configuration is such that all records comprising a discount of a value between 30- 40 % should be approved by John.
- Once step1 is approved, in step 2, all records comprising a discount of 41-50 % should be sent to James.
- Once step 2 is approved, in step 3, all records above 50 % should be sent to the CFO.

Suppose you have provided a discount of 52 % which is approved by John and James, and then rejected by the CFO. You do not want to reduce the amount and restart the entire approval process from John. To do so set up and Auto-Reapproval criteria such that,

1. For any Discount where the Current value = A constant value of 50 it should be approved.
2. For any Discount where Current value <=Prior Value it should be approved.

You can create an Advanced Condition such that if either 1 or 2 is satisfied, the request is auto-approved. Once you reduce the discount, step 1 and 2 is auto-approved, and the request is forwarded for approval only to the CFO.

Use case 2: Enabling auto re-approvals for Proposal and Proposal Line Items

You can submit the approval requests for a Quote/Proposal and Proposal Line Items for re-approval. In a scenario where a Quote/Proposal is approved but the Approvals Manager decides to change the discount from 10 to 15% on a Proposal Line Item, you can resubmit the approval request for this for re-approval.

The system honors the auto-reapproval criteria, using which a request once approved by certain assignees are auto-approved.

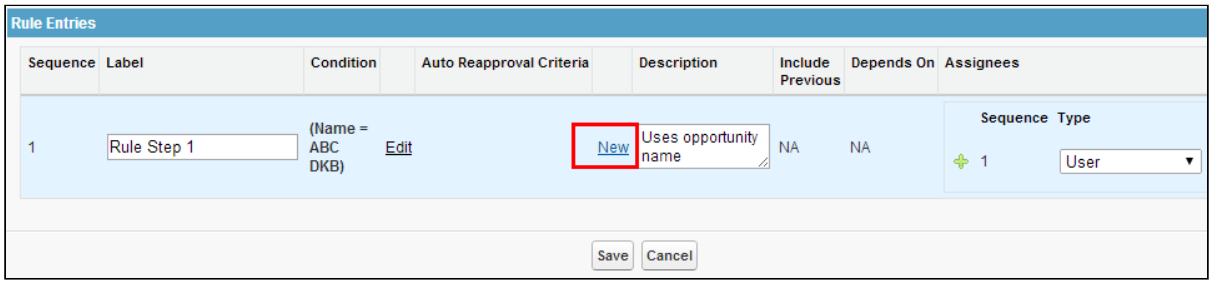
The updateApprovalData API re-evaluates all the child record IDs (Proposal Line Item IDs in our case) for reapproval. For details, refer to Conga SOAP API reference guide. Note that

this API is available in the base Approvals package and can be used with the optional unmanaged custom package "Conga Proposal Auto-Reapprovals" available on request with Conga Release Management. This package contains the complete sample code that can be used along with the API in the base package to make auto-reapprovals work with Proposals and Proposal Line Items.

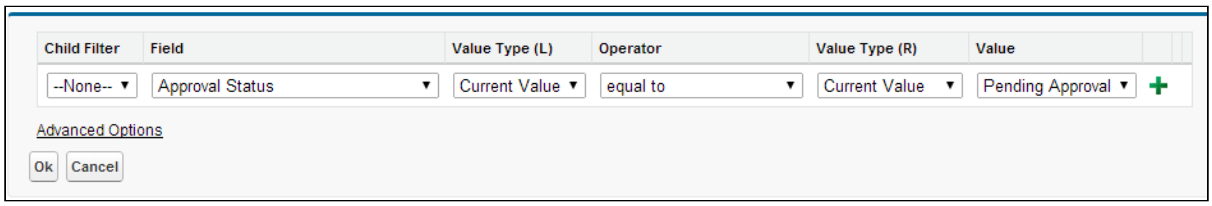
To configure reapprovals for approval rules

The approval rule must be of type Condition. You cannot have reapprovals for Dimension types.

1. Go to an approval rule and click Manage Entries.



2. Click **New** below Auto Reapproval Criteria to display the criteria options.



3. Select the fields and operators to create your expression:

Option	Description
Child Filter	This field value will be *None*, unless there are child custom objects associated with the approval rule's *Business Object*. If there is a child filter available, selecting it will populate the *Field* list with fields from the child object. Child filters do not support auto reapproval. Whenever auto reapproval is required, you should use child processon line items or child objects.

Option	Description
Field	The fields available are related to the *Business Object*, or a *Child Filter* if one is selected. If you have defined Search Filter Approvals as Entry Criteria on your Line Item or Child object, all the corresponding fields of your Child object are populated here.
Value Type (L)	*Current Value* will use the current value of the field to determine if it should be auto reapproved. *Prior Value* uses the value of the field before the approval process was resubmitted to determine if it could be auto reapproved.
Operator	Common sense needs to be applied when selecting operators. For instancea field value for *Approval Stage* cannot be *greater than* another field. If you select *in* or *not in*, the *Value* list will become a multi-select list and you can choose multiple values, if applicable. For example if the field is *Approval Status*, you could select multiple values associated with it, such as *Not Submitted* and *Pending Approval*. Ensure that you set a sequence number for the entry criteria to view the values.
Value Type (R)	*Current Value* and *Prior Value* are the same as Value Type (L). *Constant Value* means whatever you subsequently enter in the Value field is used for the expression, regardless of at what point during the approval process the field had that value.
Value	The value that is compared to Field/Value Type (L) to determine if the expression is true or false.

Click + to add another line to the expression and make it an *AND* relationship, where each line must be true.

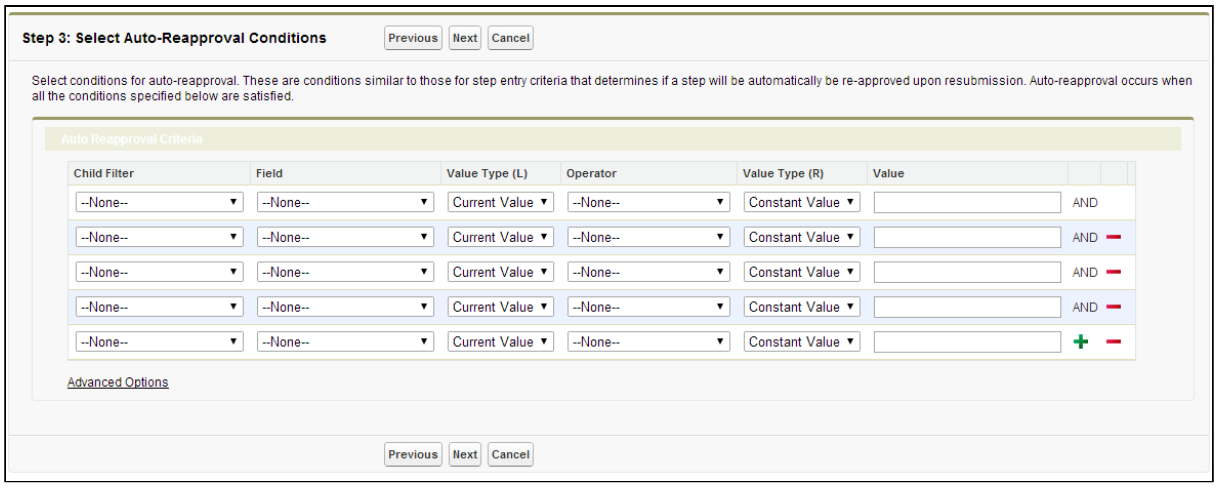
4. Click **Advanced Options** if you want to have a combination of *OR* and *AND* relationships between each of the lines of the expression.
5. Click **OK** and the expression is added to the Auto Reapproval Criteria column.
6. Click **Save** to update the approval rule.

When you have an approval process that contains an approval step that references the approval rule, auto reapproval may occur, based on the configured settings. You can add more auto reapproval criteria for that approval rule entry or for any other entries for that rule.

To configure reapprovals for process steps

The approval process step type must be Standard.

1. Select an approval process and click New Approval Step.
2. Ensure you select **Standard** from the **Step Type** list and proceed to the Select Auto-Reapproval Conditions page.



3. Select the fields and operators to create your expression:

Option	Description
Child Filter	This field value will be *None*, unless there are child custom objects associated with the approval rule's *Business Object*. If there is a child filter available, selecting it will populate the *Field* list with fields from the child object. You do not need to select a value for this. Child filters do not support auto reapproval. Whenever auto reapproval is required, you should use child processon line items or child objects.
Field	The fields available are related to the *Business Object*, or a *Child Filter* if one is selected. If you have defined Search Filter Approvals as Entry Criteria on your Line Item or Child object, all the corresponding fields of your Child object are populated here.

Option	Description
Value Type (L)	*Current Value* will use the current value of the field to determine if it should be auto reapproved. *Prior Value* uses the value of the field when it was last approved by the approver during a previous iteration of the approval process, which may not be the most recently changed value of the field.
Operator	Common sense needs to be applied when selecting operators. For instance, a field value for *Approval Stage* cannot be *greater than* another field.If you select *in* or *not in*, the *Value* list will become a multi-select list and you can choose multiple values, if applicable. For example, if the field is *Approval Status*, you could select multiple values associated with it, such as *Not Submitted* and *Pending Approval*. Ensure that you set a sequence number for the entry criteria to view the values.
Value Type (R)	*Current Value* and *Prior Value* are the same as Value Type (L). *Constant Value* means whatever you subsequently enter in the Value field is used for the expression, regardless of at what point during the approval process the field had that value.
Value	The value that is compared to Field/Value Type (L) to determine if the expression is true or false.

The Step Auto Reapproval Criteria field used to store Reapproval Criteria accommodates 32000 characters.

4. Click + to add another line to the expression and make it an *AND* relationship, where each line must be true.
5. Click **Advanced Options** if you want to have a combination of *OR* and *AND* relationships between each of the lines of the expression.
6. Click **Next** and proceed to finish configuring the approval step.

When you resubmit the approval process, and the auto reapproval criteria are met, the process will automatically reapprove the appropriate steps.

Consolidated Approvals

Consolidated approvals have been enhanced to handle situations where the same approver is included in multiple dependent steps. The dependencies of the steps are respected and the approver is assigned approval requests sequentially.

The consolidation feature is available for Approval Requests and Notifications both. This is useful when a single user is a part of multiple approval requests from a given approval process.

For example, Sam has the following responsibilities:

- Sam is a Senior Finance Manager
- He is a part of Deal Desk queue
- He is a part of Legal Review Team queue
- He plays a role of Senior Legal Analyst

For a given Quote/Agreement, Sam has to perform the following:

- Approve the payment terms.
- As a Senior Finance Manager, oversee the discount provided.
- As a Senior Legal Analyst, approve renewal of a legal contract, being a part of the Legal queue.

In such cases, when there are multiple approval requests being assigned to a single user, Consolidated Approvals and Notifications is useful. When implemented, the approver does not receive multiple emails, but a single email consolidating all the approval requests for a particular type. The type can be **User**, **Queue**, **Role**, **Related User**, **Custom User**, and more.

For example, when the discount is applied over different line items which require approval, Sam will receive individual emails consolidating the following approval requests:

- 2 individual requests related to payment terms and billing frequency
- 2 individual notification steps related to payment terms
- 2 requests assigned to legal review team
- 2 requests assigned to the legal analyst role
- 2 notification steps assigned to the legal analyst role

Assignee Type	Agreement/Quote header fields	Number of Requests
Senior Finance Manager User	Billing Frequency = Net 30	1 Notification
Senior Finance Manager User	Payment Term = Quarterly	1 Notification

Legal Review Team Queue	Managed Services = Yes	1
Legal Review Team Queue	NDA Signed = False	1
Senior Legal Analyst Role	Account Status = Red	1
Senior Legal Analyst Role	Approved Reseller = No	1
Senior Legal Analyst Role	Payment Term = Quarterly	1 Notification
Senior Legal Analyst Role	NDA Signed = False	1 Notification
Deal Desk Queue	Percentage discount > 20%	1
Deal Desk Queue	Total Amount > $500000	1

Sam can then go to each request or approve the entire Quote/Agreement.

Note:

1. System does not consolidate approval requests across multiple quotes, agreements or other custom objects.
2. System considers the **Assignee Type** (User, Role, Queue, and more) and the Request Type (Assignment and Notification) while consolidating the approval requests.

In the above example, in case if you do not check **Consolidated Approvals**, Sam will receive **10** emails for a given Quote/Agreement.

When **Consolidated Approvals** is checked, separate emails will be sent for each of the responsibility of Sam, because there are different approval steps configured for each such responsibility.

In total, Sam will receive **5** email notifications, one for each of his responsibility.

- 1 email, consolidating 2 **User** notification requests assigned to Sam, which are related to payment terms and billing frequency.
- 1 email, consolidating 2 Requests, which are assigned to Legal review Team **Queue**.
- 1 email, consolidating 2 Requests, which are assigned to the **Role**: Senior Legal Analyst.
- 1 email, consolidating 2 notifications steps, which are assigned to the **Role**: Senior Legal Analyst.
- 1 email, consolidating 2 requests assigned to the Deal Desk **Queue.**

The consolidated notification/email goes to the approver only when all the items, which require approval are ready for approval and have entered the approval process. Till the

time the last approval request enters the approval process, the requests which have already entered the process are kept in the *On Hold* status.

For scenarios with **Consolidated Approvals** and **Consolidated Notifications** checked at the process level,

All approval requests assigned to a user will be in an on-hold status and only the final approval request will be in an assigned status. An email is sent out to the user with all the available approval requests. Only one request will be in the Assigned status while all other will be in an On-Hold status. It depends on the user to approve/reject the approval requests by one of the following options:

- Approve/reject all the approval requests at one go from the email. The decision taken here is applied to all the requests, which means, when approved, all the approval requests will have the *Approved* status.
- Approve/reject one request which is in assigned status from MyApprovals page. The decision taken here is applied to all the requests, which means, when approved, all the approval requests will have the *Approved* status.

For information on where Consolidation is used for Approvals, refer Creating an Approval Process under an Approval Process.

Delegate Approver for an approver

You can assign a delegate user to the current approver. The delegate user can approve requests on behalf of the assigned approver. All approval requests and email notifications are routed to the current approver and the delegate approver.

The following features and limitations are applicable with the delegate user functionality:

- The delegated assignee receives all approval requests/email notifications assigned to the context approver.
- The delegated assignee can execute all the functionality that the context user can such as approve, reject, add comments, add adhoc approver, or take ownership of a queue.
- The delegated assignee cannot be a queue. If the Primary user is a part of a Queue, an email notification is directed to the delegate assignee and they can take ownership of the queue.
- If the Primary user is a custom user or is a related user then the all the email notifications and access should be assigned and sent to the delegate assignee.

- If the context user is a part of a Queue and receives queue email notifications, then the delegate assignee also receives those email notifications and can take ownership of the queue. (queue within a queue).
- If the same user is assigned as a Backup Admin to an approver and as a parallel Ad hoc approver, the approval request is shown in both the My Approvals tab as well as in the Take Ownership tab.
- If the primary user has requested notification on the submission of an approval request, the delegate/backup approver will receive the notification.
- **Consolidation**: Consolidation is based on the original approval process. If the delegate user is a part of the approval process, they receive a separate notification for the requests directly assigned to them as a primary user. Delegate approvers also receive a delegate email notification based on the consolidation rule of his primary user. For example, If consolidated notifications is turned on and If the Primary user has requests in the approval process, they will be consolidated together and available to the primary user (and the delegate approver) when all the requests are in an assigned status. If the delegate approver has his own separate requests in the process, they will be consolidated together and available to the delegate approver when all the requests are in an assigned status.
- **Delegate Setup**: Non-Time bound delegation.
- **Suppress notifications options**: Option to suppress email notifications to delegate users via a checkbox.
- **Re-assign**: After you re-assign a request, the new users delegate and the new users receive the approval request.

Setting up Backup or Delegate Approvers

Backup or Delegate approvers enable you to delegate approval request tasks and help ensure approval processes do not get delayed waiting for someone to make a decision. From the August 2016 release onwards you can add up to three delegate approvers for a single user.

You can also add a custom link to the user layout so you can assign a backup or delegate approver for a user from the page layout itself.

Workflow rules needed to be added to the system to ensure the feature works as expected and that the users you select to be backup or delegate approvers receive delegated approval requests.

Note

- If the backup approver workflow rule is already set up, do not set up separate workflow rules for a Delegate approver as the object used is same.
- If the backup approver workflow is not set up and if the customer wants to use the Delegate feature, then set up workflow rules similar to the effective workflow rule for a Backup approver.
- When choosing users to act as backup or delegate approvers, you can use the Backup/Delegate Approvers tab or the Set Backup/Delegate Approver button, if it has been added to the User page layout to designate a proxy.
- You can either assign a backup approver or a delegate approver at a time.
- The delegate approver receives only one email for which they have been set up as a delegate for. The delegate approver does not receive emails about any previous assignments made to the Primary user, however, the delegate approver does have access to approve/reject these records.
- From the May release onwards:

  The current backup/Delegate approver field *Apttus_Approval_Backup_User_c* has been deprecated.
  Any custom backup/delegate field/list should map to the new field *Apttus_Approval_DelegateUserIds_c*.

To create Set in Effect on Effective Date rule

Build > Create > Workflow & Approvals > Workflow Rules

1. Click New Rule.
2. Select the **Backup/Delegate Approver** object and click **Next.**
3. For Rule Name, enter`Set In Effect on Effective Date`and for Evaluation Criteria select **Created, and any time it's edited to subsequently meet criteria**.
4. Do any one of the following:
   - If you have an existing workflow rule criteria, ensure that you modify the rule. If you do not have rule criteria, ensure that you create one.

     ```
     (Backup/Delegate Approver: Effective Date NOT EQUAL to null ) AND (Backup/Delegate Approver: Is Active EQUALS True) AND (Backup/Delegate Approver: Is Delegate EQUALS False)
     ```

   - For a delegate approver specify the rule criteria as follows and click **Save & Next**:

```
(Backup/Delegate Approver: Effective Date NOT EQUAL to null ) AND
(Backup/Delegate Approver: Is Active EQUALS True) AND (Backup/
Delegate Approver: Is Delegate EQUALS True)
```

5. Click **Add Time Trigger**, select the following and then click **Save.**

6. Click **Select Existing Action** in the Workflow Actions section.
7. From the Search drop-down list, select Field Update and in the Available Actions list, select Field Update: **Set Cancellation Date to Null, Field Update: Set Effective Date to Null, Field Update: Set In Effect Flag To True.**
8. Click to move them to the Selected Actions list and click **Save.**
9. From the Specify Workflow Actions page click **Done** and click **Activate** from the Workflow Rule Detail section.

The workflow rule is active and helps ensure backup and delegate approver functionality works as expected.

To create Cancel Backup/Delegate User on Expiration Date

Build > Create > Workflow & Approvals > Workflow Rules

We do not recommend that you create a cancellation workflow rule for a delegate user since delegate users must be perpetual.

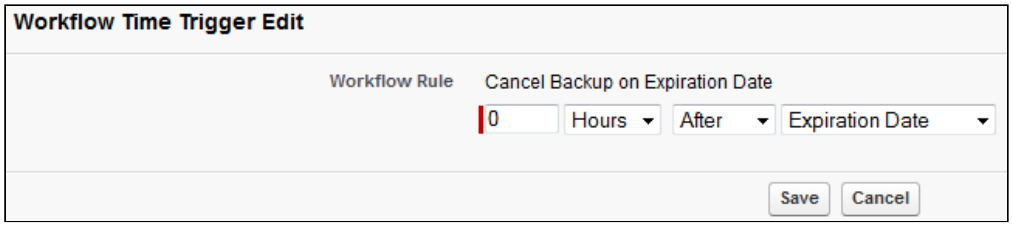
1. Click New Rule.
2. Select **Backup/Delegate Approver** Object and click **Next.**
3. For Rule Name, enter`Cancel Backup on Expiration Date` and for Evaluation Criteria select **Created, and any time it's edited to subsequently meet criteria**.
4. Do any one of the following:
   - For a backup approver specify the rule criteria as follows and click **Save & Next**:

```
(Backup/Delegate Approver: Expiration Date NOT EQUAL to null )
AND (Backup/Delegate Approver: In Effect EQUALS True) AND (Backup/
Delegate Approver: Is Delegate EQUALS False)
```

a. For a delegate approver specify the rule criteria as follows and click **Save & Next**:

```
(Backup/Delegate Approver: Expiration Date NOT EQUAL to null )
AND (Backup/Delegate Approver: In Effect EQUALS True) AND (Backup/
Delegate Approver: Is Delegate EQUALS True)
```

5. Click **Add Time Trigger**, select the following and then click **Save:**



6. Click **Select Existing Actio**n, from the Workflow Action section.
7. From the Search drop-down list, select **Field Update** and in the Available Actions list, select Field Update: **Set Cancellation Date to NOW, Field Update: Set Is Active to False, Field Update: Set Effective Date to Null, Field Update: Set Expiration Date to Null, and Field Update: Set In Effect Flag to False.**
8. Click to move them to the Selected Actions list and click **Save.**
9. From the Specify Workflow Actions page click **Done** and click **Activate** from the Workflow Rule Detail section.

The workflow rule is active and helps ensure backup approver functionality works as expected.

Associating a Backup Approver or Delegate Approver to a User

There are two ways you would want to associate a backup or delegate approver to a user.

Backup approvers can be set via the Backup/Delegate Approver tab or from a User profile. Once past the initial entry point, the process of associating a backup or delegate approver to a user is the same.

Once you have set the backup or delegate approver for a user, you must always click *Deactivate* before you begin making changes and must click *Activate* again for the saved modifications to take effect.

Behavior limitation: If you have a single user acting as the backup approver for multiple users and you have set it up to simultaneously re-assign approvals to those original users, the system may not be able to complete that action.

To add the backup and delegate approver custom link for a User

1. Click Edit for User Layout.
2. Create a new Section.
3. From the Custom Links list, drag and drop Set Backup/Delegate Approver to the Custom Links section of the page.
4. Click Save.

To add multiple delegate approvers for a User

Build > Customize > Backup/Delegate Approver > Page Layouts

1. Click Edit Layout.
2. From the Visualforce Pages list, drag and drop Delegate Approver to the new section of the page.
3. Click Save.

The Set Delegate approver VF page is now available in the Custom section.

You can now select a user you want to delegate approval requests for and complete the Associate backup approver to a user task.

Before you associate a backup or delegate approver, ensure that you remove the Backup/Delegate User field from the Backup Approver Layout.

To associate a backup approver

1. Select the Backup/ Delegate Approvers tab or click and select Backup/Delegate Approvers. - or - If the custom link has been added, you can also complete this task from Administration Setup > Manage Users > Users. Select the user you want to assign a backup approver to and go to the Custom Links section and click Set Backup Approver or Set Delegate Approver and proceed to step 4.
2. Click New to display the Backup Approver Edit page. By default, Transfer in-flight and Is Active is selected, while In Effect is cleared.
3. If you are setting a backup user, ensure that the Is Delegate check box is cleared.

4. Click the lookup icon for Current User to select the user you want to assign a backup approver to.
5. Click the lookup icon for Backup User to select the user you want to act as the backup approver.
6. Select Transfer in-flight if you want approval processes already underway to be transferred to the backup approver, otherwise clear the option and only new approval processes can get routed to the backup approver. When you select the Transfer in-flight checkbox for a backup approver, the approval request is available only to the backup user.
7. Enter the Effective Date and Expiration Date, for the period you want all approvals that are sent to the Current User to be automatically transferred to the Backup Approver. Cancellation Date does not impact and is only used for administrative purposes to flag when someone manually ended the effective period, before the expiration date was reached. It is recommended that you do not specify an Effective date and the Expiration date for a delegate approver.
8. Click Save.

The user's page is displayed and approval requests will now be delegated to the Backup Approver when the Effective Date is reached. The Activate action occurs automatically on that date. When the Expiration Date is reached, the Deactivate action occurs automatically.

You can create multiple entries for Backup/Delegate records. At a time, only one record is active. Also you can create multiple entries for Backup/Delegate records for different date ranges, as long as the date ranges are non-overlapping. All approval requests are routed to the designated Backup/Delegates during the activated time period(s).

Use Case 1: Multiple entries for Backup/Delegate record for different date ranges

Record	Primary User	Backup Approver	Effective Date	Expiration Date	Is Active
1	Sam	Jack	09/01/2019	09/30/2019	True
2	Sam	Bob	11/01/2019	11/30/2019	True

Both the records are not activated and are future dated records.If you activate the second record now (i.e current date) the following message dialog appears:Primary user Sam already has a Backup/Delegate that is scheduled. This will activate the new record immediately as well as any ones currently scheduled. Are you sure you want to continue?Click Yes to activate the second record (Sam as the primary user and Bob as the Backup Approver) immediately. The Effective Date is set to the

current date with no expiration date.On 09/01/2019, the second record is deactivated and the first record (Sam as the primary user and Jack as the Backup Approver) is activated.On 11/01/2019, the second record (Sam as the primary user and Bob as the Backup Approver) is activated and the first record is deactivated.On 11/30/2019, the second record is also deactivated.

Use Case 2: Multiple entries for Backup/Delegate record

Sam is a primary user and Jack is a backup approver. This record is activated.Now, when you create a new record where Sam is a primary user and Bob is the backup approver and activate it. The following message dialog appears: Primary user Sam already has a Backup/Delegate that is in effect. This will deactivate the existing record and set the new one in effect. Are you sure you want to continue? Click Yes to activate the new record (Sam as the primary user and Bob as the Backup Approver). The previous record (Sam as the primary user and Jack as the Backup Approver) is deactivated at the same time.

To Create Delegate Approvers

1. Select the Backup/ Delegate Approvers tab or click and select Backup/Delegate Approvers.
   - or - If the custom link has been added, you can also complete this task from Administration Setup > Manage Users > Users. Select the user you want to assign a delegate approver to and go to the Custom Links section and click Set Delegate Approver and proceed to step 4.
2. Click New to display the Backup/Delegate Approver Edit page. By default, Transfer in-flight and Is Active is selected, while In Effect is cleared.
3. To set a delegate user, select the Is Delegate checkbox. The Suppress Delegate Notification check box appears if you select the Is Delegate check box. Selecting this checkbox enables you to restrict notifications to the delegate user for every approval request assigned to the Primary approver.
4. Click the lookup icon for Current User to select the user you want to assign a backup approver to.
5. Click the lookup icon for Delegate Approver 1, Delegate Approver 2, Delegate Approver 3 to select the user you want to act as the delegated approver. The approval request will be routed to the primary approver as well as all the delegate approvers. The delegate approvers can perform all the actions as the primary user
6. Select Transfer in-flight if you want approval processes already underway to be transferred to the Delegate approver, otherwise clear the option and only new approval processes can get routed to the approver. The Transfer in-flight checkbox is always selected for a delegate approver, the approval request is then available to both the Primary user and the delegate approver.

7. Enter the Effective Date and Expiration Date, for the period you want all approvals that are sent to the Current User to be automatically transferred to the Backup Approver. Cancellation Date does not impact and is only used for administrative purposes to flag when someone manually ended the effective period before the expiration date was reached. It is recommended that you do not specify an Effective date and the Expiration date for a delegate approver.
8. Click Save.
9. Click Edit Layout and remove the Backup/ Delegate user field.

When required, you can return to the User Detail page to edit or deactivate the backup or delegate approver. If you want to edit the backup or delegate approver, you must deactivate the backup or delegate approver first, make your changes, and then activate it again. This is required because the Backup or Delegate Action (Activate/Deactivate) initiates workflows and logic required for the Current User/Backup User relationship.

We recommend that you do not select the In-Active, In-Effect, Is Delegate, or Suppress notification checkboxes from the All Records View.

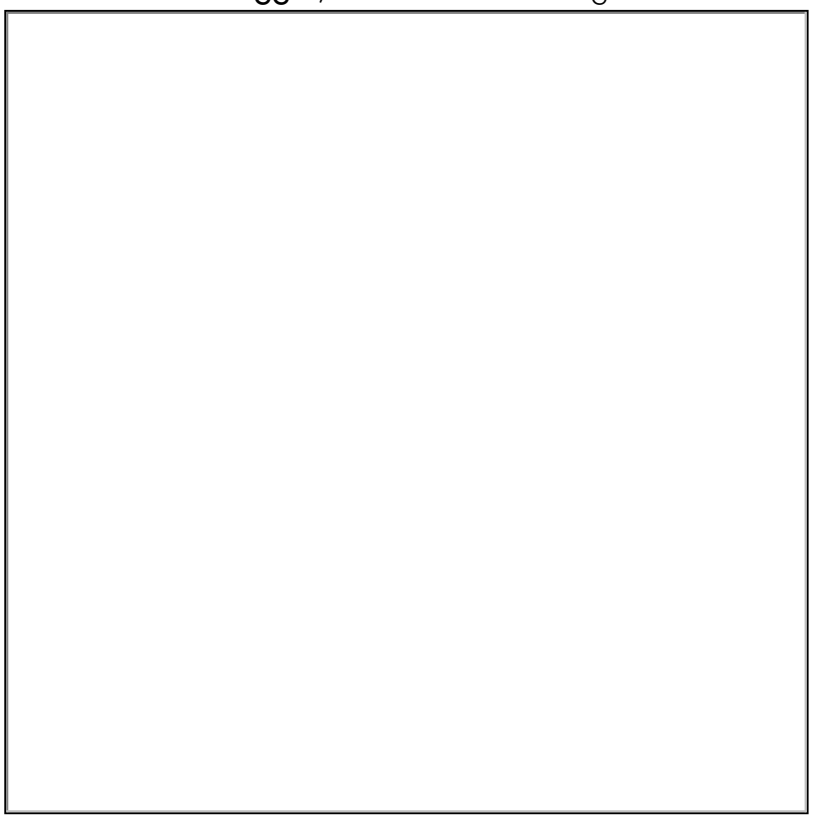
To create an email notification for a delegate user

Build > Create > Workflow & Approvals > Workflow Rules

Create an E-mail template to be associated with the to E-mail notification.

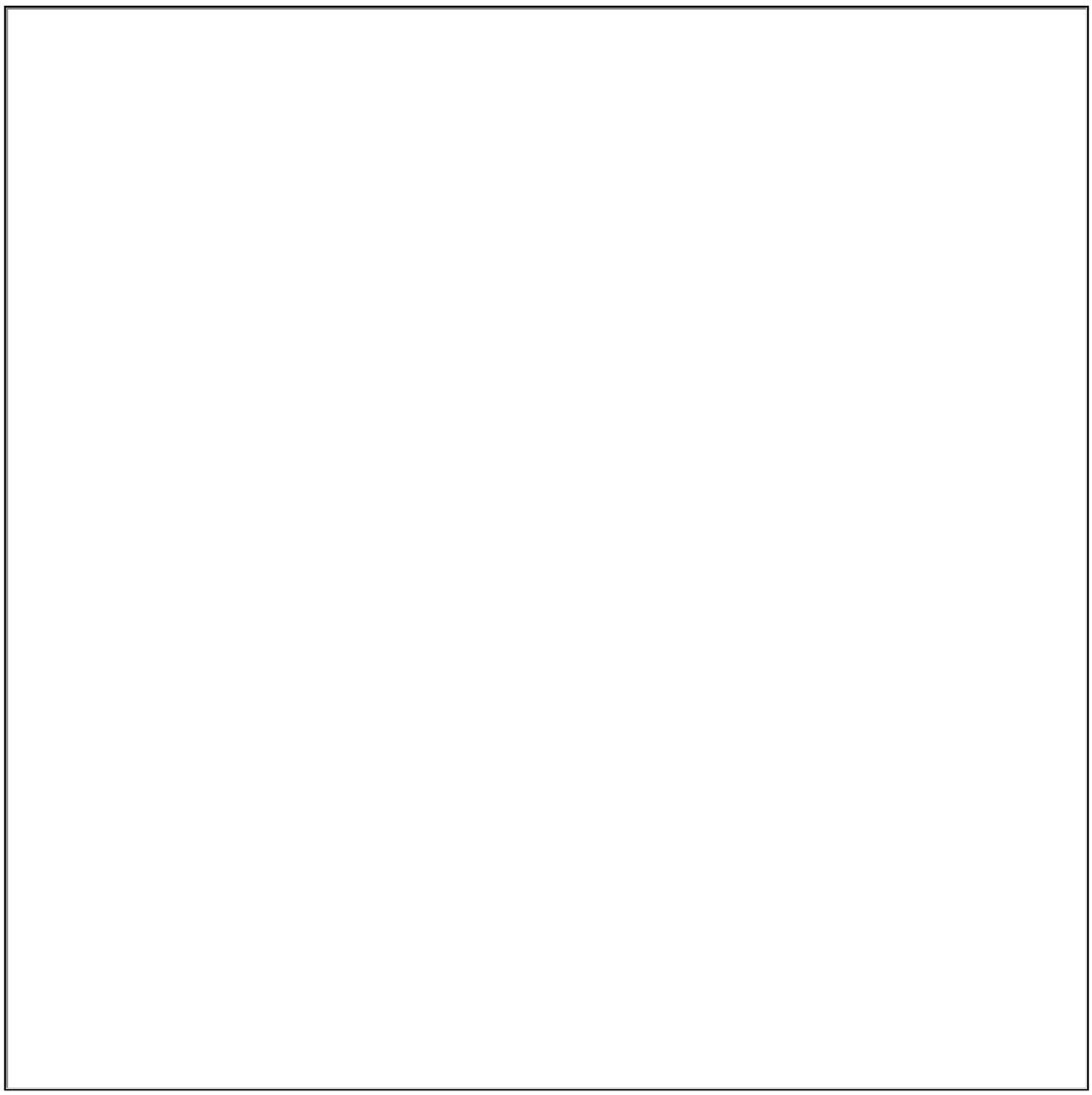
1. Click New Rule.
2. Select **Backup/Delegate Approver** Object and click **Next.**
3. For Rule Name, enter `Send Notification Delegate` and for Evaluation Criteria select **Created, and any time it's edited to subsequently meet criteria**.
4. For a delegate approver specify the rule criteria as follows and click Save & Next: `(Backup/Delegate Approver: In Effect EQUALS True) AND (Backup/Delegate Approver: Is Delegate EQUALS True)`.

a. Click **Add Time Trigger**, select the following and then click **Save:**



b. From the Workflow Action related list, click New **Email Alert**.

c. Create an e-mail Alert as follows:



d. From the Specify Workflow Actions page click **Done** and click **Activate** from the Workflow Rule Detail section.

The workflow rule is active and helps ensure delegate approver notification works as expected.

Backup Approver Search Page Configuration

The Backup Approver Search page provided as part of the Approvals package is a custom Visualforce page that needs to be correctly linked to selecting a backup user.Previously the New and Edit buttons displayed the standard Salesforce pages, which could have led to incorrect backup approvers being selected, as approval matrix levels are not considered. This enhancement directs you to the correct Backup Approver page from the User Detail

record, as well as when you create new backup approvers or edit existing backup approvers from the Backup Approvals tab.

Previously when you clicked  for backup users in the Backup Approver Details page, a lookup window would be displayed. Now, a new Backup User Search page is displayed.

To configure backup approver search pages

1. Select the **Backup Approvers** tab or click All tabs and select **Backup/Delegate Approvers**.

   > **Note**
   >
   > If the backup approvers custom link has been added, you can also complete this task from **Administration Setup > Manage Users > Users** and select the user you want to assign a backup approver to.

2. Click **New** to display the Backup Approver Edit page - or - Click **Edit** for an existing backup approver to display the same page.
3. Ensure there is someone in the **Current User** field and then click  for the **Backup User**.

A new Backup User Search page is displayed from where you can search by name and then select a valid backup user. Previously, a Lookup pop-up window would be displayed.

To configure actions for Backup Approver Search page

Build > Create > Objects > Backup Approver

1. Go to Standard Buttons and Links and click Edit beside New.
2. Select Visualforce Page and select BackupApprover [Apttus_Approval__BackupApprover].
3. Click Save.
4. Repeat Steps 1 to 3 for the Edit action.

The Backup User  link now takes you to the custom Visualforce page (Backup User Search). For existing records, ensure that you remove the Backup/Delegate User field from the Backup Approver Layout.

Setting up the Backup Administrator

The backup admin user is responsible for handling routing issues that may occur during the Approvals approval process, while admin profiles enable you to associate users – via their roles – with admin level access.

Typically, approvals will progress as expected, but in instances where an appropriate user cannot be found to handle an approval request, having designated administrators ensures an approval process can still be completed. As these users may be responsible for a number of administrative tasks, you should ensure they have wide ranging access to the objects used in your Salesforce environment.

The backup admin user configured below is used globally. You can also designate a backup admin user for each approval process, which will be used instead of the global backup admin user.

> **Attention**
>
> You must ensure that the user you select as the backup admin user is always an active user. If you are going to deactivate that specific user, then be sure to select another active user to assume the backup admin user role.

To setup the Backup Admin user

Build > Develop > Custom Settings

1. Click **Manage** beside **Approvals System Properties**.
2. Click **Edit** beside **System Properties**.
3. In the **Backup Admin User** field, enter the first and last name of the appropriate person and click **Save**.

   > **Note**
   >
   > Only one user can be designated as the backup admin.

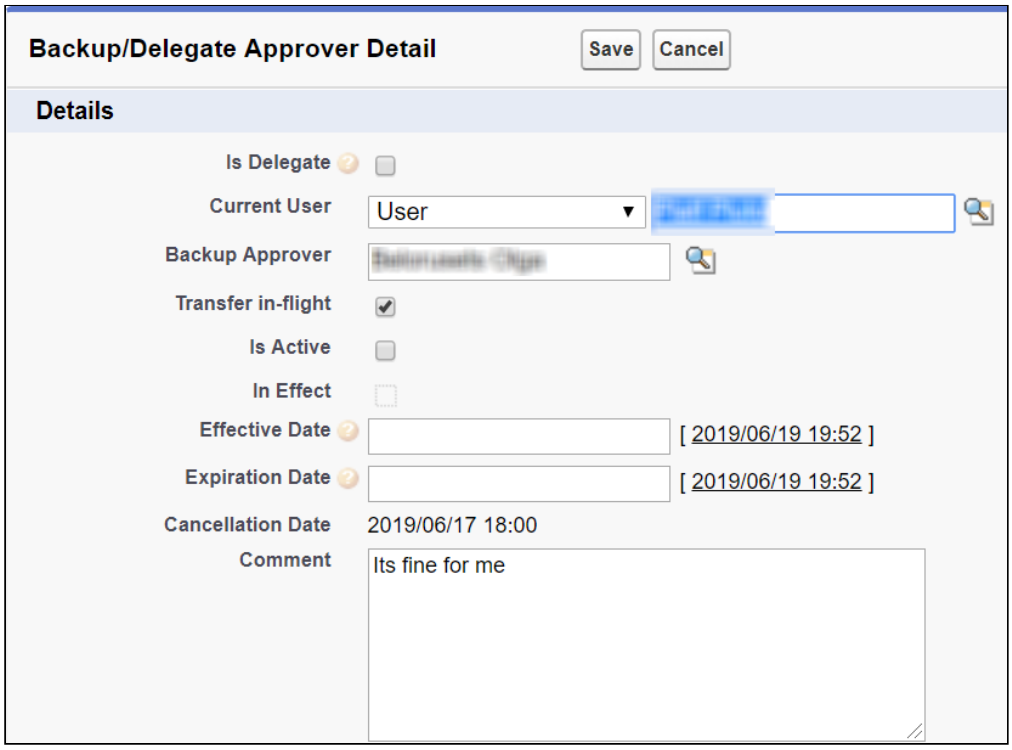
The selected person is now the backup admin user and if there are any tasks the system cannot route as expected, the backup admin user will be able to deal with them.

Setting Up a Backup or Delegate for Out-Of-Office

As an approver, you can easily set up your Out-Of-Office for all your approval requests when you are on leave or set up a backup or delegate when you are Out-Of-Office. You can select a time frame as well as assign someone to receive your notifications, approvals and general tasks.

To set up Out-Of-Office

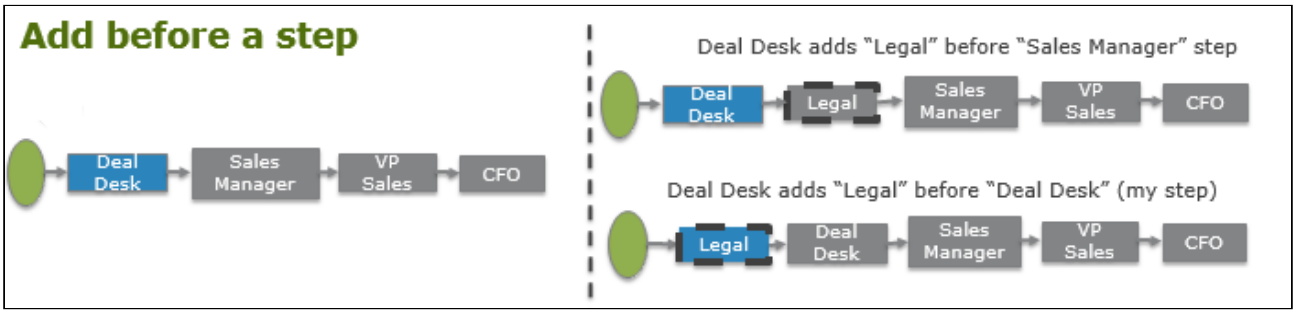
1. Go to **My Profile** > **User Detail**, under Custom Links click **Set Backup/Delegate Approver**. The Backup/Delegate Approver page appears.



2. In **Backup Approver**, type to search and select a user whom you want to set as the backup user.
3. Set the **Effective Date** and the **Expiration Date** when you will be away and you want the backup user to be active.
4. In **Comments**, you can enter a note or comment to specify the purpose of the record.
5. Click **Save**.
6. To activate the backup user, from **Backup/Delegate Action**, click *Activate*.

Adding an Ad Hoc Approver

This feature allows an approver to add a new approval step in an active approval process. For example, if, while approving a request, the Deal Desk concludes that a quote will require additional approvals from Legal and the CFO, creates those approval steps for a quote, decides the sequence for approval, and creates a new approval step.

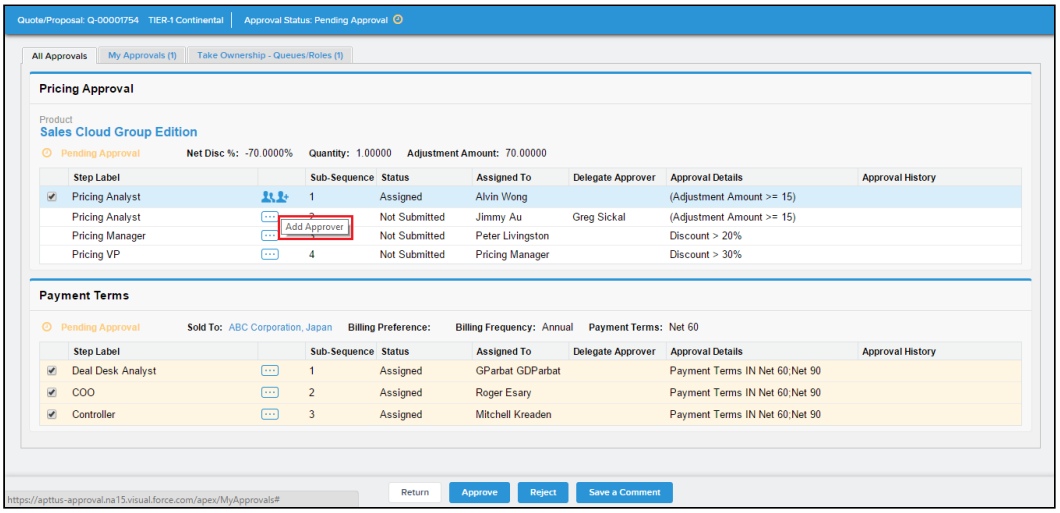


By adding ad hoc approvers, you can:

- Fast track approvals for risk scenarios that do not have predefined automated controls and approval policies for risk mitigation.
- Provide input to improve and optimize approval policies over time.

You can add ad-hoc approvers from the My Approvals screen. Approvers are allowed to add ad hoc approvers in the following points in the approval process.

- Before a selected approval step
- After a selected approval step
- Parallel to a selected approval step



The overall document must be in the Pending Approval status and the selected approval step must have one of the following approval statuses.

- Assigned
- Not Submitted
  Add Approver cannot be started from a Not Submitted step but can be started from an active step to a Not Submitted step. Approvers cannot add additional approvers before, after, or parallel to already Approved, Rejected, On Hold, or Cancelled steps.

This capability is available for headers and child object approval items. In addition, the capability is available for Conga standard objects as well as any custom object.

Note

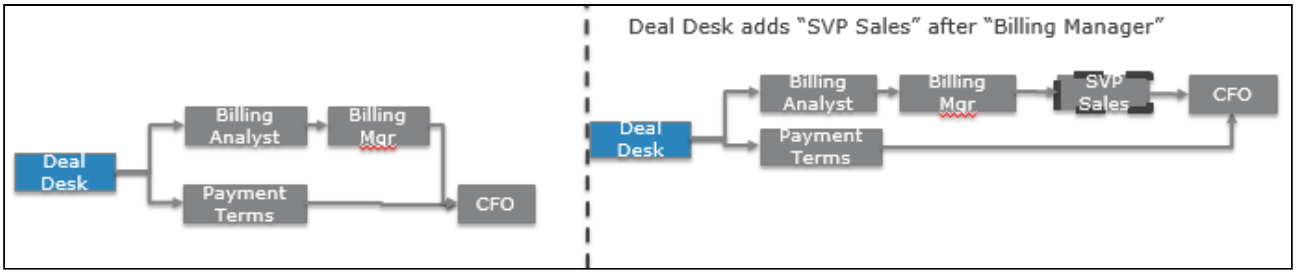
Ad hoc approval steps will be available only in the current iteration of the approval. If the document needs to be resubmitted for approval for any reason, the previously added ad hoc approver is not retained in the resubmitted approval. Users can choose to add an ad hoc approver again.
The behavior is the same even when the ad hoc approver rejects the approval request.

Adding an Ad Hoc Approver After an Assigned Approver

This section describes the scenario where you add an ad hoc approver after a currently assigned approver.

Your organization's approval workflow is designed such that, for any operational costs incurred, the first approval is assigned to the Deal desk, the second approval is routed to two assignees, the Billing Analyst and Payment Terms, in parallel. After the Billing Analyst approves the request, the next assignee is the Billing Manager. After the Billing Manager and Payment Terms approve the request, an approval request is forwarded to the CFO. From the My Approvals page, you can now add an ad hoc approver after a specific step in the process or after an existing approver.

Suppose the bills for an operation, sale, or deal exceed a certain value, the Billing Manager might want the SVP of Sales to approve the request. From the My Approvals page (where all the steps are listed), select the step where the Billing Manager is assigned a request. Click the Add Approver icon next to that step and add the SVP of Sales as an approver after the Billing Manager. The following diagram shows the scenario where you add an ad hoc approver after an assignee.



Dependencies for a new ad hoc step are as follows.

- **Depends on steps**: The new ad hoc approval is dependent on the selected step. For example, select the Billing Manager step, and add the SVP Sales as a step after the Billing Manager step. The SVP step is now dependent on the Billing Manager step.

- **Steps dependent on new step**: Steps that were dependent on the selected step are now dependent on the new ad hoc step. After you add the SVP Sales step, all the steps dependent on the Billing Manager step, such as the CFO step, are now dependent on the SVP step.

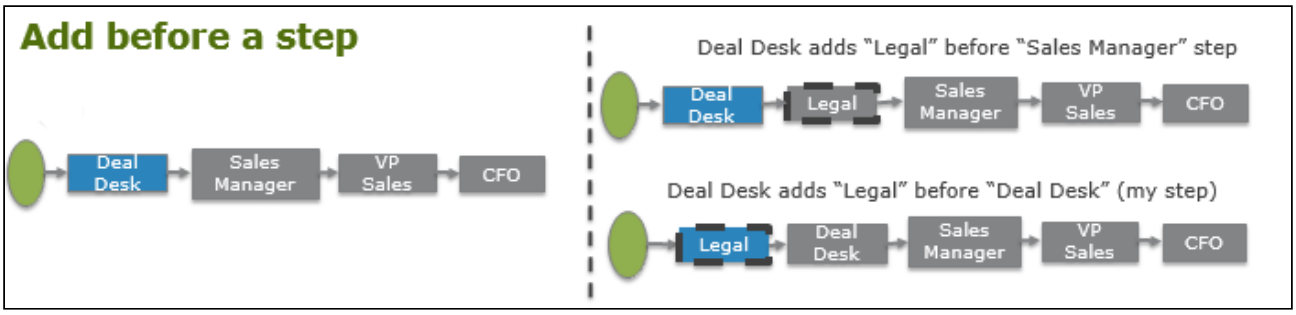
The revised dependencies for an existing selected step are as follows.

- **Depends on:** Does not change
- **Steps dependent on existing step**: New ad hoc step

Adding an Ad Hoc Approver Before an Assigned Approver

This section describes the scenario where you add an ad hoc approver before a currently assigned approver.

While you are negotiating your agreement, the customer asks you to change a clause for an early termination of an agreement. In this case, you might require an approval from the Legal Team before the request is forwarded to the CFO. Based on your approval process configuration, the request will be forwarded to the CFO after the VP of Sales has approved the request. If the CFO has not yet approved or rejected the request, you can add an ad hoc approver before the CFO from the My Approvals page. The CFO step must be in the Assigned or Not Submitted state. Similarly, you can choose to add an ad hoc approver before the current assignee. In the following process workflow, the current assignee of the approval request is the Deal Desk. As a Deal Desk owner, you can choose to add Legal as an approver before you. In the My Approvals page, each step of the approval process is listed separately. To add an ad hoc approver before a selected step, select the step you want to add the approver before and click the Add Approver icon.



Select a step with a Not Submitted or Assigned status and add the step before the selected step.The dependencies for a new ad hoc step are as follows.

- **Depends on steps**: Existing dependencies for a selected step are transferred to the new ad hoc step, step. In other words, the new ad hoc step inherits the dependencies of the step it was added from. For example, if Legal is added as a new step before the Deal Desk step, the Legal step is now dependent on all of the steps the Deal Desk step was dependent on. All the step dependencies in the Deal Desk step are now transferred to the Legal step.
- **Steps dependent on new step**: The selected step is dependent on the new ad hoc step. For example, if you select the Deal Desk step and add a Legal step before the Deal Desk step, the Deal Desk step is now dependent on the Legal step.

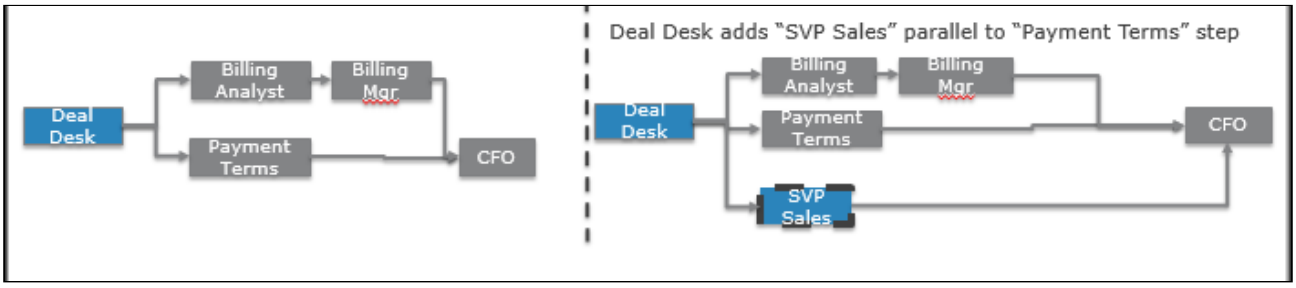
The revised dependencies for existing selected step are as follows.

- **Depends on**: New ad hoc step
- **Steps dependent on existing step**: Does not change

> **Note**
>
> **Ensure that you are done with your configuration process before you add an ad hoc approver.**

Adding an Ad Hoc Approver Parallel to an Assigned Approver

This section describes the scenario where you add an ad hoc approver parallel to a currently assigned approver.Your organization's approval workflow is designed such that for any operational costs incurred, the first approval step is assigned to the Deal Desk, and the second approval is routed to assignees, the Billing Analyst and Payment Terms, in parallel. After the Billing Analyst approves the request, the next assignee is the Billing Manager. After the Billing Manager and Payment Terms approve the request, an approval request is forwarded to the CFO. From the My Approvals page you can now add an ad hoc approver parallel to a specific step in the process or parallel to an existing approver. Suppose the bills for an operation, sale, or deal exceed a certain value, the Payment Terms team might want the SVP of Sales to approve the request. From the My Approvals page (where all the steps are listed), select the step where the Payment Terms is assigned a request. Select the add approver icon next to that step and add SVP of Sales as an approver parallel the Payment Terms step. The following diagram shows the scenarios where you add an ad hoc approver parallel to an assignee.



The new dependencies for new ad hoc step are as follows.

- **Depends on steps**: The new ad hoc approval step is dependent on the same steps the selected step is dependent on. For example, add the SVP Sales step parallel to the Payment Terms step. The SVP Sales step is dependent on all the steps the Payment terms step is dependent on. i.e. Deal Desk.
- **Steps dependent on new step**: Steps dependent on the selected step are dependent on the new ad hoc step as well as the existing step. For example, add the SVP Sales step parallel to the Payment Terms step. All the steps dependent on the Payment terms step is dependent on the SVP Sales step as well. In other words, the CFO Step is also dependent on the SVP Sales step.

The revised dependencies for an existing selected step.

- **Depends on**: Does not change
- **Steps dependent on existing step**: Remain dependent on the existing step as well as the new ad hoc step.

> **Note**
>
> **Ensure that you are done with your configuration process before you add an ad hoc approver.**

Prerequisites for Adding an Ad Hoc Approver

Before you add an Ad Hoc approver, you must configure and create the following items for the primary or child objects for which you trigger an approval.

- My Approvals
- Approvals Pages
- Approval Required Check Trigger
- Search Filters (Approvals)
- Create approval email templates
- Set the Enable Adhoc Approval flag to true from Approvals Custom Config settings.

To add an ad hoc approver

One or more approval requests must be assigned to you before you can add an approver. The approval status of the object must be Approval Pending. The approval status of the step to which you want to assign an approver to must be Assigned or Not Submitted.

1. Navigate to the object to which you want to add an ad hoc approver.
2. Click **My Approvals**.
   If you are a member of a queue or a role, you can add an approver only after taking ownership of the queue/role assignment.
3. Select the step that you want to and an ad hoc approver before, after, or parallel to.
4. Click the **Add Approver** icon. The **Add Approver** dialog opens.
5. Select one the sequences from the drop-down list. Enter a Step Label for the step.
   - **Add approver before this approver**: Enables you to add an ad hoc approver before the selected step approver.
   - **Add approver after this approver**: Enables you to add an ad hoc approver after the selected step approver.
   - **Add approver parallel with this approver**: Enables you to add an approver parallel with the selected step approver.
6. Enter a **Step Label** for the step.
7. Select an Approver using the lookup icon. The approver can be of type User, Queue, Role, or Related User.
8. Add a comment. This comment is visible to the ad hoc approver in the email notification they receive and in the Approver comments.
9. Click **Save**.

The ad hoc approver receives the email notification and can approve or reject a request using the Salesforce org credentials or from within an email. If the ad hoc approver is added before an assigned step, then the originally assigned approver will get an additional notification once the request becomes available again for assignment.

Reassigning an Approval Request

This feature allows an approver to reassign an approval step in an active approval process in the Assigned state. For example, if, while approving a request, the Deal Desk concludes that a quote needs to be reassigned to Legal to fix terms and conditions, the deal desk assignee will reassign the request to Legal.

The following limitations and features are supported with the reassign functionality:

- Allow user to filter and search the assignees of a type from the displayed list,
- Custom assignee search screen for searching and filtering the approvals,

- No support for reassigning to queues, roles, or related users.

By reassigning approvers, you can:

- Fast track approvals for risk scenarios that do not have predefined automated controls and approval policies for risk mitigation.
- Provide input to improve and optimize approval policies over time.

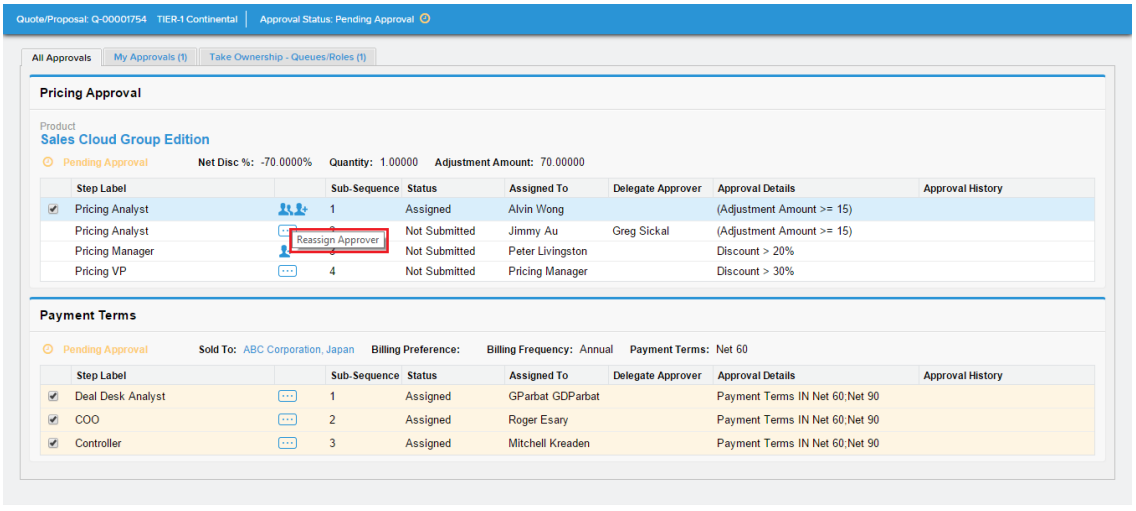
You can reassign approvers from the My Approvals screen. The overall document must be in the Pending Approval status and the selected approval step must be in the Assigned or Reassigned state.

This capability is available for headers and child object approval items. In addition, the capability is available for Conga standard objects as well as any custom object.

Prerequisites for Re-assigning an Approver

Before you reassign an approver, you must configure and create the following items for the primary or child objects for which you trigger an approval.

- My Approvals
- Approvals Pages
- Approval Required Check Trigger
- Search Filters (Approvals)
- Create approval email templates



To reassign an approver

One or more approval requests must be assigned to you before you can add an approver. The approval status of the object must be Approval Pending. The approval status of the step to which you want to assign an approver to must be Assigned or Reassigned.

1. Navigate to the object to which you want to reassign an approver.
2. Click **My Approvals**. If you are a member of a queue or a role, you can add an approver only after taking ownership of the queue/role assignment.
3. Select the step that you want to reassign an approver to.
4. Click the **Reassign Approver** icon. The **ReAssign Approver** dialog opens.
5. Select an Approver using the look-up icon.
6. Add a comment. This comment is visible to the reassigned approver in the email notification they receive and in the Approver comments.
7. Click **Save.**

The reassigned approver receives an email notification and can approve or reject a request using the Salesforce org credentials or from within an email.

Auto-escalating Approval Requests

When an approval request is not approved, rejected, or reassigned within the allotted time, it can automatically be reassigned to a new approver so the approval process can continue.

> From the March 2016 release onwards, Auto-Escalation is enabled for (Sub-Process) Approval Rules only. Do not setup Auto-Escalation on (Child Process) Approval Rules. For example, if an approval rule is setup on a child record object, such as Line Item or Agreement Line Item and this Approval rule is used as a child process, then auto-escalation is not executed for entries in that approval rule.

Once an approval request has been created, the system uses the value of *expected time to complete* (days and hours) to determine when the approval request should be auto-escalated. A background service runs that periodically checks on the status of approval requests and when one is flagged as exceeding that time, the auto-escalation is triggered.Auto-escalation is triggered at the approval step level. If you do not want to wait for the auto-escalation to occur, you can also manually escalate approval requests from the *Approvals* related list for the object that is under approval. Auto-escalations can have one of two paths – single-step or multi-step – depending on the auto-escalation assignee type.

Single-step	After an approval is auto-escalated to the next user, if that user does not decide on the approval in the designated time, the approval request is auto-escalated to the backup admin user. For example, if the auto-escalation Assignee Type is *Role*, and you have a Territory Manager role with three users assigned to it, the auto-escalation will be assigned to the first active user assigned to that role. If that user does not decide on the request within the allocated time frame, the request is sent to the backup admin user and that is the extent of the auto-escalation path.Single-step assignee types are *Backup Admin User*, *Queue*, *Related User*, *Role*, *Rule*, and *User*. When the assignee type is queue or Backup Admin user, there is no delegate user. **Note** If you select *Rule* as an **Assignee Type**, the approval request is sent only to the first user that satisfies the filter criteria in the Approval Rule. If you have multiple approvers in a sequence, only the first approver is selected and assigned.
Multi-step	The auto-escalated request can move amongst multiple users for that assignee type before being sent to the backup admin user.For example, if the auto-escalation assignee type is *User Delegated Approver*, the initial auto-escalation will send the approval request to that user's delegated approver, as configured in their user account. If the request is not decided on within the allocated time, the request would be auto-escalated to the delegated approver of that user who received the auto-escalated request. This would continue until that assignee type was exhausted, then it would be sent to the backup admin user.Multi-step assignee types are *User Approval Matrix Next Approver*, *User Delegated Approver*, and *User Manager*.

Configuring Auto-escalation

You need to configure the following for this feature.

Workflow Rules	Workflow rules that contain time triggers cannot be included in a managed package, so you must configure them manually to provide the time-based framework for this feature.

Auto-escalation Reminders	Reminder emails can be sent to approvers to let them know if they do not approve, reject, or reassign an approval request, it will be automatically escalated.
Approvals System Property	**Enable Approval Request Auto-Escalation** is the flag for the system to use the auto-escalation functionality, based on the settings configuration.
User Approver Settings	If you want to auto-escalate to User Delegated Approver or User Manager assignee types, you must configure these settings.
Email Templates	These templates are used to generate the email notifications that are sent to the auto-escalation approvers. The system comes with escalation templates for agreements, opportunities, and term exceptions. For other objects that use approvals you will need to create custom templates. **Attention** If you are going to use attachments with your emails, you MUST create custom email templates. Default templates do not support attachments.
Global Escalation Options	These options must be configured, so that if no escalation options are configured for the approval process step, the escalation process will fall back on these settings to determine when the escalation will begin and who will be involved.

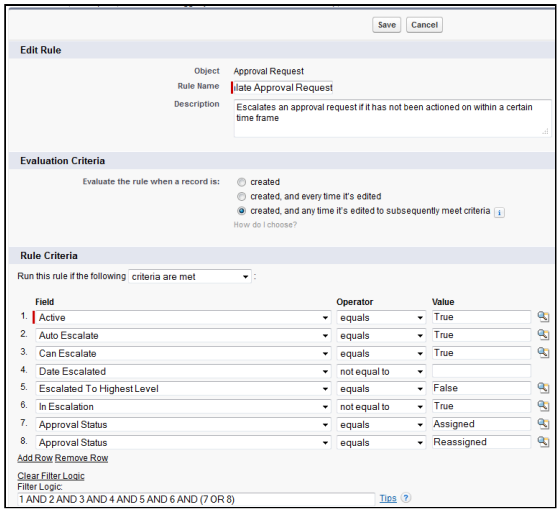
By completing this configuration, you can use auto-escalations and they will work according to the configuration; however, you will be able to override some of the options later when creating approval processes.

To configure Workflow Rules

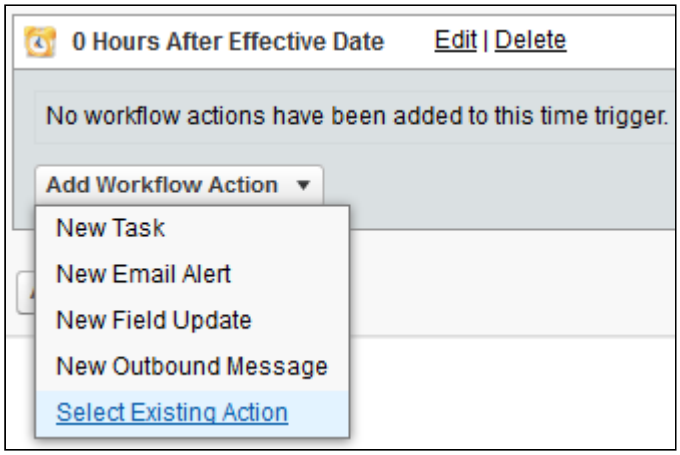
1. Go to **Build > Create > Workflow & Approvals > Workflow Rules** and click **New Rule**.
2. Select **Approval Request** from the list and click **Next**.
3. Enter *Escalate Approval Request* for the mandatory **Rule Name** and for **Evaluation Criteria** select **created, and any time it's edited to subsequently meet criteria**.
4. Enter the following **Rule Criteria**:
    - Active equals True

- Auto Escalate equals True
- Can Escalate equals True
- Date Escalated not equal to Null
- Escalated to Highest Level equals False
- In Escalation not equal to True
- Approval Status equals Assigned
- Approval Status equals Reassigned

5. Enter the following **Filter Logic**:`1 AND 2 AND 3 AND 4 AND 5 AND 6 AND (7 OR 8).`



6. Click **Save & Next** to display the Specify Workflow Actions page.
7. Click **Add Time Trigger** and set the values to `0 Minutes After Date Escalated` and click **Save**.
8. In the **Time-Dependent Workflow Actions** section, select **Add Workflow Actions > Select Existing Actions**.



9. From the Choose **Action Type** list, select **Field Update** and from the **Available Actions** list, select **Field Update: Set Can Escalate To False, Field Update: Set Date Escalated To Null, Field Update: Set in Escalation to True** and click **Add**. These actions are already included in the Approvals package and only need to be associated with the workflow rule here.
10. Click **Save** and return to the **Specify Workflow Actions** page.
11. Click **Done** on the Specify Workflow Actions page and you are returned to the main Workflow Rule page. Subsequently, activate the workflow rule.

The required workflow rule framework for auto-escalated approval requests is now ready.

You can add additional workflow triggers for informing someone their request will be auto-escalated, configure the Auto-escalate settings, and email templates.

To enable the Approval Systems Property

1. Go to **Build > Develop > Approvals System Properties** and click **Manage.**
2. Click **Edit** beside **System Properties**.
3. Select **Enable Approval Request Auto-Escalation** and click **Save**.

This enables the auto-escalation feature for the system.

Set up the required email templates and global escalation options.

To select Approver Settings

1. Go to **Administration Setup > Manage Users > Users** and select **Edit** for the appropriate user.
2. Scroll to the **Approver Settings** section and complete the following:

Option	Description
Delegated Approver	This can be any user, but you should ensure they have access to the same objects as the user they may receive delegated approval requests for.
Manager	The user's manager.
Receive Approval Request Emails	If you select Never, the user will not receive email notifications that they have had an auto-escalated approval request sent to them. They will receive notifications with any other option.

3. Click **Save**.

You can now select *User Delegated Approver* and *User Manager* as auto-escalation assignee types for this user.

Repeat this task for other users, as there is no batch action update available for these settings.

Email Templates for Auto-escalation

By default, the Conga Approvals Management package contains Escalation templates for *Opportunities*, *Agreements*, and *Term Exceptions*.

These templates can be used as-is out of the box; however, you can also configure bespoke templates for your organization. See Setting up Email Templates and Alerts and the email Appendices for more details.

Auto-escalation Behavior

This section outlines the behavior for auto-escalation in Approval Rules.

Scenarios	Behavior
• Auto escalation is active • Expected **hours** to complete =1 hour, X hours • Auto escalation assignee is a "User" **X= any other variable # of hours*	Approval request is escalated to the assignee "User" if the original assignee does not take an action within **(1 or X)** hours of generation of the approval request If the escalated user assignee does not take any action within the stipulated **(1 or X)** hours then the request should be escalated to the backup admin. Same behavior applies for all other types of assignee.
• Auto escalation is active • Expected **hours** to complete =1 • Auto escalation assignee is a "Delegate Approver"	Approval request is escalated to the assignee "**Conga Delegate Approver**" if the original assignee does not take an action within **(1 or X)** hour of generation of the approval request If the escalated assignee does not take any action within the stipulated **(1 or X)** hours then the request is escalated to the backup admin. Approval request should remain with the escalated assignee for the remaining life cycle of the request.

Scenarios	Behavior
• Auto escalation is active • Expected **hours** to complete =1 • Auto escalation assignee is a "Approval Matrix"	Approval request is escalated to the first assignee in the approval matrix if the original assignee does not take an action within the stipulated **(1 or X)** hour of generation of the approval request. • If the first assignee does not take any action within one hour, the request is escalated to next assignee in the approval matrix chain. • The escalation process continues till the last assignee in the approval matrix is assigned the request. • If the last assignee does not take any action then the approval request is reassigned to the backup admin. ⓘ The request traverses up the approval matrix at the interval defined in the auto escalation setup. For example, after every one hour the request is escalated to the next assignee if no action is taken on the approval request.
• Auto escalations is active • Expected **Days** to complete =1 • Auto escalation assignee is a "User" **X= any other variable # of Days*	Approval request is escalated to the assignee "User" if the original assignee does not take an action within **(1 or X) days** of generation of the approval request If the escalated assignee does not take any action within the stipulated **(1 or X)** hours then the request is escalated to the backup admin. Same behavior applies for all other type of assignees.
• Auto escalations is active • Expected **Days** to complete =1 • Auto escalation assignee is a "Delegate Approver"	Approval request is escalated to the assignee "**Conga Delegate Approver**" if the original assignee does not take an action within **(1 or X) Days** of generation of the approval request. If the escalated assignee does not take any action **(1 or X)** hours then the request is escalated to the backup admin.

Scenarios	Behavior
• Auto escalations is active • Expected **Days** to complete =1 • Auto escalation assignee is a "Approval Matrix"	Approval request is escalated to the first assignee in the approval matrix if the original assignee does not take an action within (**1 or X) Days** of generation of the approval request. • If the first assignee does not take any action within one day, the request is escalated to next assignee in the approval matrix chain. • The escalation process should continue till the last assignee in the approval matrix is assigned the request. • If the last assignee does not take any action then the approval request should be reassigned to the backup admin. ⓘ The request traverses up the approval matrix at the interval defined in the auto escalation setup. For example, after every one day the request is escalated to the next assignee if no action is taken on the approval request.

Auto-Escalation Limitations

- There are limits on the number of time-based workflow rules per hour. The limits are as follows:
  - Enterprise Edition =500
  - Developer Edition =50
  - Unlimited Edition =1,000
- Auto-escalation cannot be defined for rules you want to use as a child process.
- For an assigned request if the expected completion time is greater than the current time then the request will be auto-escalated.
- If an assignee in a standard step is a rule, the assignees in the rule cannot have auto escalation values.

Best Practices

1. As a best practice, escalation time should be minimum for one day. Not less than that.
2. Whenever Escalation is set up for a specific user, role, or a queue the same format should be followed at each and every step.

For example, Configure a rule entry where Jim is an assignee and Mark is the escalation assignee and the escalation time setup up is 1 day. If Jim does not respond to a request in 1 day, it will be escalated to Mark.

Given this logic, if there is any other step (standard or subprocess step) that also has Jim as an assignee, the step should also have Mark as the escalation assignee and an escalation time of 1 day.

This is also a system limitation. For example, if John has auto-escalation in one step, no auto-escalation in another step, and enables consolidated approvals, escalation does not work. In this scenario, the escalation request goes on hold.

Consolidation and Auto-Escalation Behavior

If consolidation is enabled (Approvals or Notifications), consolidation will always take precedence over escalation. For example, unless all requests are available (a user has multiple requests, at least one request should be in the assigned status) the escalation timer will not start till then.

Reassign and Auto-Escalation Behavior

Consider an approval request which has Jim as a primary assignee and Mark as the auto-escalation assignee on it. The request is scheduled to escalate one day after assignment. After some time, the request is reassigned to Karl. This will not impact the escalation time. If Karl does not take any action, the request will escalate as expected (i.e. it will escalate 1 day since it was assigned to Jim.)

Ad hoc Approver and Auto-Escalation Behavior

If an ad hoc approver is added, the end user does not have an option to add escalation value to the ad hoc assignee.

Auto Approve/Auto Complete and Auto-Escalation Behavior

Escalation should never be set up on Auto Approved Steps.

Clone Quote with its existing configuration

You can now enable an approval flag to clone header and it's line item approval status and approval requests when the line item status is Approved, Rejected, and Cancelled. Now you can clone a quote with its line items, including proposal header information.

Whenever an Account Executive has to create multiple draft options of the same quote or needs to make a change (for example, extend the quote expiry date) to a finalized quote before sending it out to the customer, they may choose clone the quote with the line items with the approval status. Each cloned version of the quote has to go through its own approval life cycle before being finalized. In an event that a quote has gone through multiple rounds of approvals before cloning. After the quote is cloned, all "Approved" approval requests get cloned to the new quote as well.

On the original quote, line items which were approved in previous iteration of the approval process flow normally have associated "Approved" approval requests in the approval request history table. When a quote is cloned, these requests will get cloned to the new quote(cart's) approval request history table **NOT** to the approval request table.

To enable the above enhancement, go to Proposal System Properties and enable **Clone with Approval Status**.

The new cloned quote will retain all the information of the parent quote, including:

- Proposal header details
- Configuration line items (if parent quote is not finalized)
- Proposal line items (if parent quote is finalized)
- Attributes of line items

The following information is not retained from the parent quote:

- Start date and End date of the Quote Header
- Activity history
- Notes and Attachments

The updated behavior when you clone a Quote/Proposal is as follows:

- If a Quote is cloned in an approved status, the line items will retain the approved status and are not considered for approval required check, unless they are

specifically modified. All approved requests related to these line items are cloned as well.

- If a Quote header is updated, all approvals related to the quote header will be retriggered. All approval requests related to the quote header will be retained. Now if the quote is modified, it may need to go through approvals again, depending on pre-defined approval rules

When you clone a quote with its line items, a new quote is generated with a unique quote number whose **Status** will be *Draft* and **Configuration** status is *Saved*. The table below highlights the behavior in the older and the current version.

Behavior in New version	Behavior in Old version
If the Line item is **Approved**, the approval status is retained in the cloned quote.	All the values were blanked out and the value in the **Approval Status** was not carried forward to the cloned quote after cloning.

The tables below outline the behavior of the Quote before and after Clone when "Clone With Approvals" flag is TRUE.

Before Clone						
Number	**Quote (Approval) Stage**	**Quote Approval Status**	**Cart Approval Status**	**Cart Status**	**Line Item -1**	**Line Item -2**
1	Draft	Not Submitted	Not Submitted	New	Blank	Blank
3	In Review	Pending Approval	Pending Approval	Pending Approval	Pending Approval	Pending Approval
4	Approval required	Approval required	Approval required	Approval required	Approval required	Approval required
5	Denied	Rejected	Rejected	Approval required	Rejected	Rejected

Before Clone						
6	Approved	Approved	Approved	Ready For Finalization/ Finalized	Approved	Approved
7	Approval required	Cancelled	Cancelled	Approval required	Cancelled	Cancelled

After Clone						
Number	**Quote (Approval) Stage**	**Quote Approval Status**	**Cart Approval Status**	**Cart Status**	**Line Item -1**	**Line Item -2**
1	**Draft**	Blank	**Configuration is not cloned**	Saved	Blank	Blank
3	Draft	Blank	Blank	Saved	Blank	Blank
4	Draft	Blank	Blank	Saved	Blank	Blank
5	Denied	Rejected	Rejected	Saved	Rejected	Rejected
6	Draft	Approved	Approved	Saved	Approved	Approved
7	Approval Required	Cancelled	Cancelled	Saved	Cancelled	Cancelled

Approval Submission Comments

Approval submission comments enable you to add personalized comments when you submit an approval request. You can have a single comment at the process level or up to three comments at the step level.

Providing comments at the process level enables you to add general instructions to all of the approvers, while adding comments at the step level enables you to tailor a specific message for each individual approver. The comments can be text only, up to 4096 characters.

Process Level comments could be used when a quote requires approval from one group and the same justification holds true for all approvers. *Step Level* comments could be used when quote needs approval from Finance, Legal, and Pricing and you want to send a separate justification for each approver group.

These comments can be viewed on:

- Approval Request Record
- Approval Request History Record
- Approval Summary Page
- Approval Email Notifications

Configuring Approval Comments

You must have custom labels available for the comments and then you must enable submission comments for an approval process, at either the process or step level.

To create comment custom labels

1. Go to **Build > Create > Custom Labels** and click **New Custom Label**.
2. Enter values for the mandatory fields. *Name* is the text you must enter in the approval process when you are selecting the label to use with the submission comments. *Value* is the label that is displayed on the approval request itself, where the submission comments will be entered.

New Custom Label

The label is now available for use with submission comments.

You can create more labels as required.With the labels available you can now create approval processes that use submission comments.If you have the **Translation Workbench** enabled in your org, you can also translate the label into multiple languages.

To configure approval comments for a process

You must have created labels.

1. Go to the **Approval Processes** tab.
2. Select an existing process or create a new one and proceed to the Step 6: Specify Submission Comment Settings page.
3. Click **Submission Comments Enabled** to display the submission comments options.
4. For **Submission Comments Type**, select **Process** and for **Submission Comment 1** Label, enter the value from the label's *Name* field you want to use. You must be sure you enter the correct value for the label, as there is no validation you have entered a valid label.

5. Click **Save**.

Complete the remaining configuration for the approval process and ensure that it is active.

When the entry criteria for this approval process is met when someone submits a record for approval, they will be able to enter submission comments at the process level.

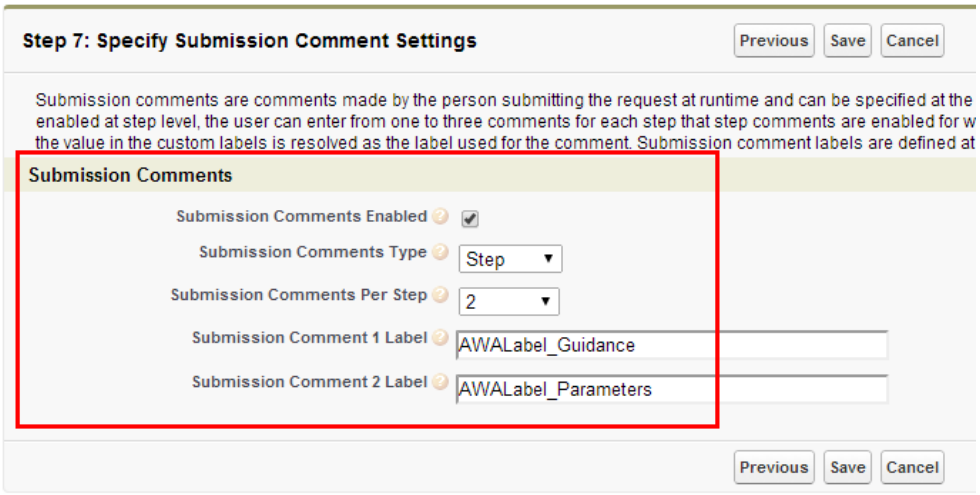
To configure approval comments for steps

You must have created labels.

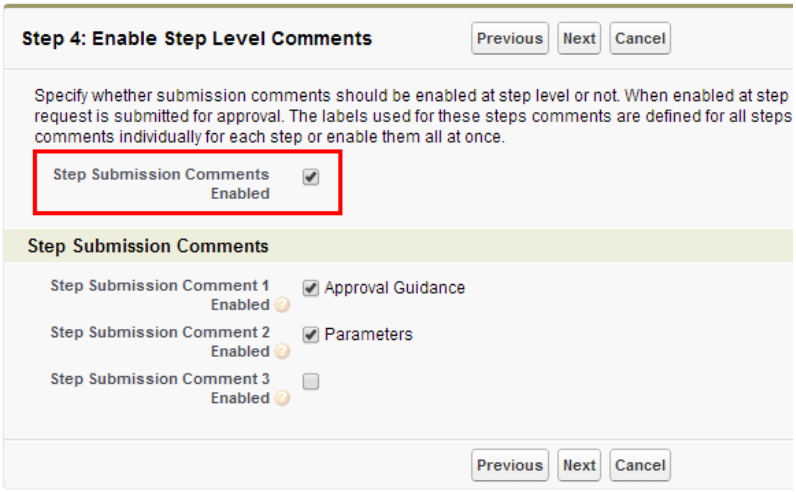
1. Go to the **Approval Processes** tab.
2. Select an existing process or create a new one and proceed to the Step 7: Specify Submission Comment Settings page.
3. Click **Submission Comments Enabled** to display the submission comments options.
4. For **Submission Comments Type**, select Step and for Submission Comments Per Step, select **1, 2,** or **3.**

> **Note**
>
> If you define submission comments at the step level, ensure that you have defined them for all the steps in the process to avoid running into SOQL error.

5. For each **Submission Comment Label**, enter the value from the label's *Name* field you want to use. You must be sure you enter the correct value for the labels, as there is no validation you have entered a valid label.



6. Click **Save**.
7. Go to the approval steps for the process and configure them as normal, and proceed to the Step 6: Enable Step Level Comments page.
8. Click **Step Submission Comments Enabled**, if you want to be able to enter submission comments for that specific step. Select which of the comments you want to use. If *Submission Comments Type* is *Process*, that overrides this setting and comments will only be available at the process level and not the step level.



9. Click **Save**.

Complete the remaining configuration for the approval process and ensure that it is active.

When the entry criteria for this approval process is met when someone submits a record for approval, they will be able to enter submission comments at the step level.

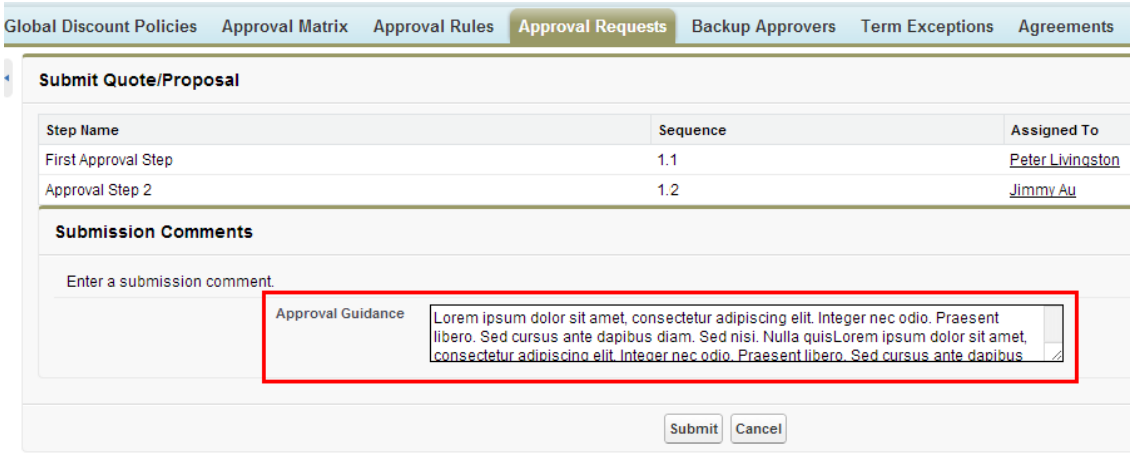
Submitting Comments

When an object record meets the entry criteria upon clicking Submit Approval, for an approval process using submission comments, the fields will be available to enter comments in.Either submission comments will be available as a single field for the entire process or 1 to 3 comment fields will be available for specific approval steps.

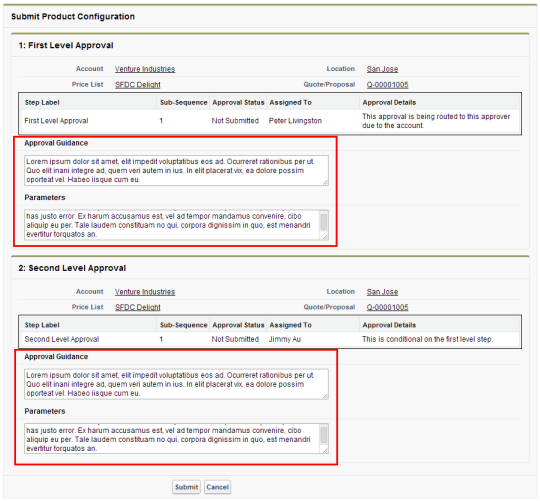
To submit comments

There must be an active approval process with submission comments enabled.

1. From the object record (such as a Quote/Proposal record), click **Submit Approvals** from the **Approval Requests** related list.
2. From the Submit page enter comments:
    a. For submission comments at the approval process level there will be a single comment field.

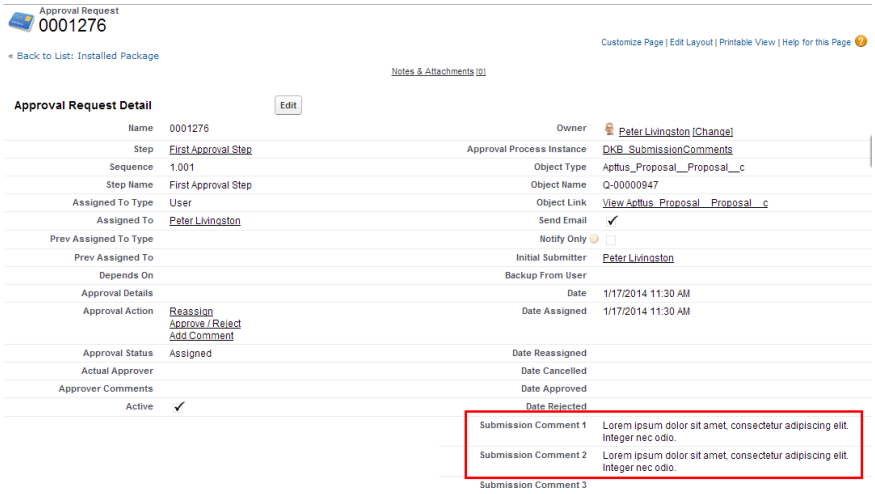


b. For submission comments at the approval step level there will be between one and three comment fields. It will also depend on how the steps were configured, whether the fields will be available for some steps or all steps. In the example below the submission comments are available for each step.



3. Click **Submit** and the approval request process starts.

The submission comments are captured on the Approval Request object and the Approval Summary Page.



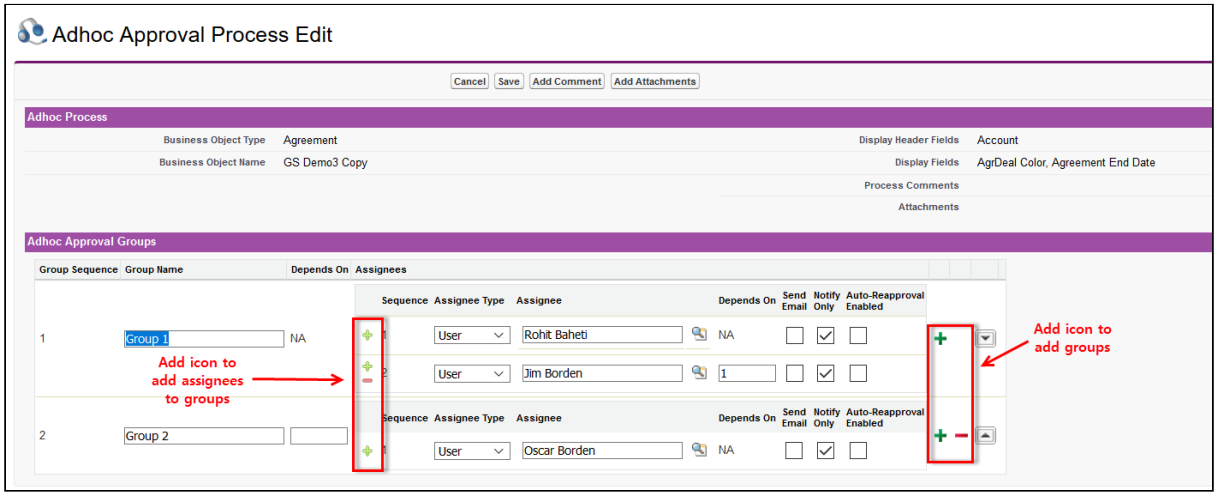
If the approval process is cancelled and a new process is started, the submission comments will be included in the Approval Request History.

Setting Up An Adhoc Approval Process in Runtime

You can create and modify an adhoc approval process in runtime. You can preview and save all approval requests with comments and attachments. You can also configure adhoc approvers - parallel or sequential. Save and preview them all at the same time on the Preview Submit Approval page. You can preview and submit approvals for complex processes.

To configure adhoc approvers from runtime

1. From the Agreement or Quote record, click the **Preview and Submit Approvals** button. The Preview Submit Approval page appears.
2. Click **Create Adhoc Process**. If you have a rule based approval process and want to switch to adhoc approval process, on the confirmation pop-up dialog, click **Yes**. The Adhoc Process Runtime page appears.
3. In the Adhoc Approval Group section, type a **Group Name**.
4. From Assignees, select an **Assignee Type**. The available assignee types are:
    - User
    - Role
    - Queue
5. Search and select an **Assignee**.
6. To send email notification to the selected assignee, set the **Send Email** flag to true.

7. To add more assignees to the group, click the Add icon next to Sequence.
8. 

9. To add more groups, select the Add icon on the extreme right of the Assignee section.
10. Click **Save**.
11. To add process level comment, click **Add Comment**.
12. To add attachment for the process, click **Add Attachments**.
13. To submit all adhoc approval request, from the Preview Submit Approval page, click **Submit**.

You can preview all these changes on the Preview Submit Approval page.

Creating Approval Processes

Conga Approvals offers the ability to setup approval processes that are currently not supported by Salesforce. It allows the customer to configure their various approval processes with greater flexibility and visibility, helping in overall compliance.

Approval steps assign approval requests to various users and define the chain of approval for a particular approval process. Each approval step specifies the attributes a record must have to advance to that approval step, the user who can approve requests for those records, and whether to allow backup assignees of the approver to approve the requests. Regardless of their complexity, when anyone in the process rejects an item, the entire process is considered rejected.

- Creating the Process Definition
- Checking the Approval Status of the Document
- Creating Initial Submission Actions
- Creating Final Actions
- Creating Approval Steps
- Setting up Auto-escalation
- Creating Expressions

- Optional Configuration

Performance and Scalability

Area	Detail
Multiple Approval Processes	**When to use multiple approval process?** When you have Region Based Approval variations, Business Unit Based Approvals variations, Channel Based Approvals variations. **How do I configure multiple approval processes?** Define mutually exclusive process entry criteria so that only one process is selected for a given primary object instance. **Why do I configure multiple approval processes?** Having multiple approval processes reduces number of approval steps / criteria system needs to execute
Approval Rule Entry Criteria	Specify minimum approval conditions in Approval Rule entry criteria for Sub Process or Child Process steps to avoid evaluation of all approval rule entries every time.
Approval Required Check	Provides users visibility to whether approval is needed for header or line items without executing the entire approval process. Provides a mechanism to avoid approval process evaluation for a large number of scenarios.

Types of Approval Steps and when do I use them?

- **Standard** - This is the existing method for creating header level approval steps.
- **Sub Process** - This is used when you want to select approval rules that are associated with the header level context object of the approval process. The assignee types are derived from the Approval Rule. Create a Sub-Process when you have multiple approvers for a set of criteria instead of defining multiple Standard steps for each criteria.
- **Child Process** - This is used when you want to select approval rules that are associated to child objects of the approval process header level context object such as you want to define an approval process for Proposal and Proposal Line Items. The assignee types are derived from the Approval Rule. Use a child process instead of a Subprocess and Standard step type when you want to populate the Approval request with the child object line items to be approved, such as proposal line items.

It is recommended that you use conditional criteria instead of Approval Dimensions and Approval Matrix while creating your approval rules and processes.

Creating An Approval Process

A process definition serves as the framework for your approval process, providing options for setting up key objects like entry criteria, email templates, and approval summary pages.

Entry criteria are used to define the conditions that must be met before an object record is routed through the approval process. For instance, you may want to specify that a certain approval process for agreements is only used when there is a discount given of greater than 10%.

While everyone may have a Salesforce dashboard set up to display outstanding tasks, having email templates associated with the approval process ensures that people are notified outside of Salesforce when they are required to approve an item, helping ensure the approval process does not stall.

Approval summary pages contain the information and action items that enable a record to be reviewed and approval actions to be taken. The default summary pages provide sufficient information for Agreements, Opportunities, Quote/Proposals, Proposal Line Items, and Term exceptions; however, the pages can be configured to provide whatever details are needed.

You can do the following using the Approval Processes tab:

- Create Rule Based Approval Process or Ad-hoc Approval Process
- Specify the Submit, Approve, and Reject actions.
- Specify Email Templates.
- Specify fields to be displayed in the approval request of an object record.

To create a rule based approval process

1. From the main menu, select **Conga Approvals Setup** and click the **Approval Processes** tab. A list of all active and inactive approval processes is displayed.
2. Set up a new rule based approval process.
3. Select the Entry Criteria which indicates when the approval process starts.
4. To override the default approval summary page and my approvals page, enter the approval summary visualforce page name and the my approvals visualforce page where the approver indicates whether they approve or reject the item and click Next. You can leave these fields blank to use the default pages. There is no auto-

lookup available for this, you must enter the exact name of the page to change the setting. The default pages are
    - OpportunityApprovalSummary
    - AgreementApprovalSummary
    - QuoteApprovalSummary QuoteLineItemApprovalSummary
    - AgreementTermExApprovalSummary
5. Specify the Backup Admin User. From **Backup Admin Approver Settings**, search and select a user who can act as the backup admin for a specific process and click Next. If an approver is deleted or is inactive, an approval request is sent by default to a backup admin. If you do not select one, the global backup admin user is used.
6. Click **Save**.

A rule based approval process is created.

To create an ad-hoc approval process

1. From the main menu, select **Conga Approvals Setup** and click the **Approval Processes** tab. A list of all active and inactive approval processes are displayed.
2. Set up a new adhoc approval process.

    > **Important**
    >
    > You must have an existing active Rule Based approval process for an Ad-hoc approval process to work.

3. Specify Consolidation Settings.
4. Select Custom Email Notification Templates.
5. Specify the Backup Admin User. From **Backup Admin Approver Settings**, search and select a user who can act as the backup admin for a specific process. If an approver is deleted or is inactive, an approval request is sent by default to a backup admin. If you do not select a backup admin, the global backup admin user is used.
6. Select Comment settings.
7. Set up fields to be displayed on the My Approvals page.
8. Set up Auto re-approval settings.
9. Click **Save**.

An adhoc approval process is created.

Setting Up a Rule Based Approval Process

1. Click **Create Approval Process**.

2. Use **Manage Approval Processes for** to select one of the following objects for which the approval is triggered and click **Create New Approval Process**.
    - Agreement
    - Agreement Line Item
    - Agreement Term Exception
    - Line Item
    - Opportunity
    - Product Configuration
    - Proposal Line Item
    - Quote
    - Quote/Proposal
3. Select **Rule Driven Process** to create rule based approval process. Specify entry criteria which when satisfied, triggers the approval process. For example, you can trigger an approval for all agreements having a Total Agreement value more than 5000 dollars.
4. Type a mandatory **Process Name** and provide a **Description** for the approval process.
5. Click **Next**.

Setting Up an Ad-hoc Approval Process

1. From the main menu, select Conga Approvals Setup, select Approval Processes.
2. Use **Manage Approval Processes for** to select one of the following objects for which the approval is triggered and click **Create New Approval Process**.
    - Agreement
    - Agreement Line Item
    - Agreement Term Exception
    - Line Item
    - Opportunity
    - Product Configuration
    - Proposal Line Item
    - Quote
    - Quote/Proposal
3. Select **Adhoc Process** to create your own approval process with the ability to select approvers/reviewers and decide their sequence of actions while viewing the object record.
4. Type a mandatory **Process Name** and provide a **Description** for the approval process.
5. Click **Next**.

Specifying Consolidation Settings

Consolidation applies to scenarios when a user is responsible for multiple approval items for a single document. In this scenario, you can use the Consolidated Notification and Approvals options to avoid sending separate approval notifications to the approver for each approval item. Consolidated Notification is the capability to send single approval email notifications to the approver for all the header or line item approvals, when all the approval items become available for approval. This enables approvers to review all their approval items for a document as a whole, rather than revisiting the same document multiple times. When all the items become available, the system changes the status of all the approval items to Assigned and then you are allowed to independently Approve or Reject the assigned approval items. You can navigate to the My Approvals page (provides the list of all the approval items for the user for that document) in the application to take approval actions. From the My Approvals page, you can selectively approve or reject assigned approval items.

The Consolidate Notifications check box when selected, enables the assignee to receive single consolidated assignment notification for all approval items. An approver then has the flexibility to approve or reject all or subset of the assigned requests. An assignee also receives consolidated e-mail notifications for cancellation and notify only requests. If an approver is defined for notify only steps, a consolidated notification email is sent after all the notify only steps are available. If an approval is cancelled, and the user is an assignee for multiple steps, the user receives a consolidated cancellation notification.

Email Approvals: In case of email approvals for consolidated notifications, when you approve or reject an approval using email, all the items that require approval are Approved or Rejected simultaneously. Email approval does not enable you to approve or reject individual approval items. You can then either respond to the email approval notification to approve or reject all the items assigned to the user or can navigate to the "My Approvals" page (provide the list of all the approval items for the user for that document) in the application to take approval actions. Within the application, from "My Approvals" page, user can selectively approve or reject assigned approval items. System will consolidate the approval requests as per the behavior specified above and send single notification for all the approval items that require approval from a given user when all the items become available, rather than sending separate notification for each approval item. The Consolidate Approvals check box when selected, enables the assignee to receive single consolidated assignment notification for all approval items. An approver can then either Approve All or Reject All the requests that require their approval.

1. To bundle multiple approvals into a single approval action, select **Consolidate Approvals**. Consolidated Approvals only works when Consolidated Notifications is selected.
2. To bundle multiple notifications into a single notification both via email and in Salesforce, select **Consolidate Notifications**. Consolidated Notifications can work independently.
3. To continue the approval process even when the approver rejects an approval request, select **Continue Pending Approvals on a Reject**. The Continue on Reject flag means "Continue to assign all approval requests in the chain as if all requests before it were approved regardless of whether they were or not." This flag at the approval process level does not have any effect on the flag at sub process (Process Rule) level. This flag at the approval process level and at sub process level do not necessarily have to be the same.
4. Click Next.

For more details and scenarios, see Consolidated Approvals, for more details.

Specifying the Custom Notification Email Templates

Specify the Custom Notification templates that are used when an approval request is sent to an approver or a reviewer via an email and click Next. You can specify the following email templates or leave the fields empty to use the default templates:

Custom Notification Templates	Description
Assignment Email Template	This is used for emails that are sent to someone who has been assigned a task, based on the approval workflow process. The default is Conga Agreement Approval Notification (Assignment).
Reassignment Email Template	When an existing task is reassigned to someone else, this is used for the email sent to that person.. The default is Conga Agreement Approval Notification (Reassignment).
Cancellation Email Template	This is used to notify people when an approval process has been cancelled. The default is Conga Agreement Approval Notification (Cancellation).
NotifyOnly Email Template	This is used when the Notify Only option is selected in one of the steps. The recipient of this email does not have any action items. The default is Conga Agreement Approval Notification (Notify Only).

Custom Notification Templates	Description
Reminder Email Template	This is used when reminders are set up. The default is *Conga Agreement Approval Notification (Reminder)*.
Escalation Email Template	This is used when auto-escalation is enabled and an outstanding approval task has reached the point where it is auto-escalated. The default is Conga Agreement Approval Notification (Escalation).

⚠ **If you submit approvals with attachments, the email templates you use need to be custom templates.**

Select Comment Settings

Submission comments are comments made by the person submitting the request at runtime and can be specified at the process or step level. When enabled at process level, the user can enter a comment that applies to the entire process when the request is submitted for approval. When enabled at step level, the user can enter from one to three comments for each step that step comments are enabled for when the request is submitted for approval. Submission comment labels specified below should be the actual name of the custom label and cannot contain spaces. At runtime the value in the custom labels is resolved as the label used for the comment. Submission comment labels are defined at the process header level but can be enabled for each step individually.

1. To customized comment that applies to the approval process or step when the request is submitted for approval, select **Submission Comments Enabled**.
2. To enable mandatory comments while approving an Approval process, select **Approval Comments**. This flag impacts an Approval on Run time, Approval Summary page as well as Approval dashboard. When the Approval comment flag at process level is enabled, an approver has to add comments while approving an approval request.
3. To enable rejection comments in an approval, select **Rejection Comments**. These comments can be entered by the submitter or an approver while approving or rejecting the request.
4. Click Save.

Selecting an Entry Criteria

If your approval process is a Rule-based process, specify entry criteria for the approval process. When the Entry Criteria is met, the approval process is triggered for an object record. You must ensure you select an appropriate operator for the field value, as there is no validation of operator-value combinations. If no criteria is selected, all records will require approval once the approval process has been activated.

1. Select the fields and operators to create your expression:

Option	Description
Field	The fields available are related to the Business Object, or a Child Filter if one is selected. If you have defined Search Filter Approvals as Entry Criteria on your Line Item or Child object, all the corresponding fields of your Child object are populated here.
Operator	Common sense needs to be applied when selecting operators. For instance a field value for Approval Stage cannot be greater than another field.If you select in or not in, the Value list will become a multi-select list and you can choose multiple values, if applicable. For example, if the field is Approval Status, you could select multiple values associated with it, such as Not Submitted and Pending Approval. Ensure that you set a sequence number for the entry criteria to view the values.
Value	The value that is compared against Field to determine if the expression is true or false.

   You can add another line to the expression and make it an AND relationship, where each line must be true. You can change the combination to OR and AND relationships between each of the lines of the expression.
2. Click Next.

Setting Up Display Fields for preview and My Approvals Page

Select fields to display for preview and My Approvals pages. A maximum of six fields are allowed per step.

1. Header Display Field lists the fields that appear in the approval request header. From **Header Display Field**, select a field that appears on the My Approvals page of the record.
2. Select display fields from the **Unselected Items** list and move to **Selected Items** list to be displayed on the My Approvals Page. To add more sections to the Approvals Summary Page, click **Add Section**.
3. Click **Next**.

Setting Up Auto Re-approval Settings

1. To set auto re-approval for the Approval process, select **Enable auto re-approval**.
2. To enable deletion of previously approved requests before the request is auto-approved., select **Allow approved requests to be deleted**.
3. Click **Next**.

Checking the Approval Status of the Document

Approval required check is performed to determine whether a document requires approval and if approval process needs to be executed.

CPQ Approvals: For CPQ Approvals, **Approval Required check** is a default system behavior and an approval process is not evaluated unless the Approval status is set to Approval Required by the approval required check. Preview, Submit for Approval / Request Approval actions are only enabled when approval status is set to Approval Required. For CPQ Approvals, entry criteria filters need to be defined on Product Configuration object at minimum. In addition, the entry criteria filters can also be defined on the Line Item object if approval is needed based on line item parameters.

Approval Required Check on Pricing: For CPQ Approvals, system auto executes Approval Required check on Reprice, if Approvals is used.

Approval Required Check on Quote Update: For CPQ approvals, system can also execute Approval Required Check on Quote update if Quote Header parameters are used in Approval Required criteria. You can use Quote Header parameters in CPQ Approval Required check using **Formula Fields (Approvals)** in Entry Criteria Search Filters.

Approval Required check does not evaluate the approval process, but checks the entry criteria of the approval process. The purpose of the approval required check is to reduce the performance overhead of performing detailed process evaluation to determine whether approval is required.
Approval criteria or conditions are defined using Search Filters (Approvals) of type Entry Criteria.

Approval Required Check is primarily used with CPQ approvals, but can also be used with approval on other objects.

Use the following guidelines while defining entry criteria filters:

Entry Criteria Filters: Search Filters (Approvals) of type Entry Criteria are used in Approval Required Check.

Entry Criteria filters on the Primary object: These entry criteria filters are defined for the approval process context object. For example, if approval process is defined for Quote/Proposal then Entry Criteria filters need to be defined on Quote/Proposal object.

Entry Criteria filters on the Child Object: If the approval conditions are also based on the parameters from the child objects of the approval context object then entry criteria filters are also defined on the related child context object. For example, if an approval process is defined Quote/Proposal, but the approval conditions are also based on the proposal line item object then Entry Criteria filters needs to be defined on Quote/Proposal line items object also.

Entry Criteria using parameters related Parent Object: Entry criteria filters can also be defined on a given object using reference fields from the related parent object. For example if an approval process is defined on Quote/Proposal then Entry Criteria filters need to be defined on Quote/Proposal object. However, if approval conditions are also based on fields from Opportunity object (parent of Quote/Proposal) then you can define **Entry Criteria** fields using Formula Fields (Approvals). In the above example, you will define Formula Fields (Approvals) on Quote/Proposal object using fields from Opportunity object. Once these Formula Fields (Approvals) are defined, the fields appear in field selection list for defining Entry Criteria filters on the Quote/Proposal object.

Guidelines for Defining Entry Criteria Filters: You can define multiple Search Filters (Approvals) of type Entry Criteria on a given object. System will evaluate all these entry criteria filters.

Creating Final Actions

Final actions enable you to update record fields in the object according to whether the approval process was completed successfully or it was rejected.

As final actions can only be used to update fields, the order in which they are created is unimportant.

> **Note**
>
> There is a single reject policy embedded in the system. If there is at least one rejection during the approval process, the entire process is rejected.

To create final actions

1. Click **Add New** in either the **Final Approval Actions** or **Final Rejection Actions** section of the approval process. To use Auto Re-approvals, set the cart status to **Ready for Finalization** in the final submission actions rather than Finalized to ensure that a new cart is not created every time.
2. Select the **Field Name** for the field in the object record you want to update.
3. Enter a valid **Field Value** for the field and include a description. If the field is a pick list, the **Specify New Field Value** list is displayed, to select from.
4. Click **Save** to add the action and return to the main approval process page. - or - Click **Save & New** to add the action and create another action of the same type.

The items are added to the appropriate final actions section. The actions will update those object record fields upon the last approval or rejection action for that approval process.

Creating Initial Submission Actions

Initial actions update object record fields when the approval process begins for that record.

The only conditions considered before the initial actions occur is the entry criteria for the process definition, which starts the approval process. There is no additional logic involved. Therefore it is best to create initial submission actions for the fields that will always require updating when the approval process begins. This ensures that all fields are handled programmatically and do not require someone to manually change them.

> **Note**
>
> As with all fields that can be updated as part of the approvals process, there is no validation that the values entered for the *Field Value* are correct while you are creating actions and approval steps. You must know what the field type is and should thoroughly test your process before using it in a live environment.

To create initial submission actions

There must be an approval process.

1. Click **Add New** in the **Initial Submissions Actions** section.
2. Select **Field Update (Constant)** from the **Field Action Type** list and select the field name. Enter the **Field Value**, ensuring it is an appropriate value type for the field. - or - Select **Field Update (Value)** and select the user lookup field for the object record you want to update with the Current User ID. The current user is the person who is logged into Salesforce and makes a change to the object record that initiates the approval process.
3. Click **Save** to complete the task - or - Click **Save & New** to add another action.

The item is added to the **Initial Submission Actions** section.

Creating Approval Steps

Approval steps provide the logic for routing object record approvals to the correct people based on specific criteria.

A minimum of one approval step is required to activate an approval process; however, a process can contain many steps. When you create steps it is important to understand the approval workflow you want to have. Along with the step criteria, step dependencies should be used when organizing your approval workflow. Any steps that are not dependent on another step will be executed in parallel.

Approval Step Enhancements

Approval steps have been enhanced to offer three different step types to the user, as well as new description fields, and the ability to select which context object fields should be displayed when an approval request or preview is sent.

Approval Step Types	There are three step types you can choose from when creating an approval step: • **Standard** - This is the existing method for creating approval steps. • **Sub Process** - This is used when you want to select approval rules that are associated with the context object of the approval process, which also replace the assignee types that are used with a Standard step. • **Child Process** - This is used when you want to select approval rules that are associated to child objects of the approval process context object, which also replace the assignee types that are used with a Standard step. It enables the *Context Object* and *Approval Rule* list.
Description Fields	There are description text fields available for adding information about a step, as well as for conditions of the step entry criteria. These fields enable you to provide useful information to the approvers about these items.
Display Fields	The fields enables you to select the header level fields of the context object you want to show to users when previewing an approval or when an approval request has been sent and the approver is reviewing the request information. Any Related Fields you create for the context object is available as display fields.

To create approval steps

There must be an approval process.

1. From the Approval Process page, click **New Approval Step** in the **Approval Steps** section.
   - or -
   At the end of creating a process definition, select **Yes, I'd like to create an approval step now** and click **Go**.
2. Enter a **Step Name, Step Label** and **Description** for the step.
3. Specify the appropriate **Step Type** and click **Next**.
4. Select the **Entry Criteria** for the step and click **Next**. See Creating Expressions for details on setting the criteria.

5. Specify the conditions for setting up Auto-Reapproval. When these conditions are satisfied, the current step will be automatically re-approved.
6. Select **Step Dependencies**, if necessary, by highlighting steps in **Unselected Items** and click ☐ to add them to the **Selected Items**. Reorder the steps as required and click **Next**. Step dependency is based on their order in **Selected Items**.
7. Select **Step Auto Complete** if you want this step to automatically be approved when it is assigned to the approver. For auto complete to work as expected, ensure that the step you set Set Auto Complete for is independent from another step. You can set up any or all standard steps in an Approval Process for auto-completion. Entry Criteria is still enforced for the selection of the step, but will be marked as auto-completed if flagged. Approval Comments are not enforced even if the comments are flagged as mandatory. Fact that the Step is auto-completed is captured in the audit trail.

8. Select the process approvers, by selecting from **Assigned Approver > Assignee Type.**

Option	Description
Approval Matrix	Use this to dynamically assign an approval request to a user who has the given approval level in the approval matrix for the user submitting the request. For example, approver authority may change based on the discount being offered on a sales contract. So different levels of approvers can be assigned based on approval authority for a given item.
Custom Queue	This points to a field which contains a queue name in any custom object with a filter to narrow it down to a single row. This can be specified in two ways: • Select the custom field in the custom object you want to use for the custom filter expression and manually enter a constant value it will be compared to. • Select the custom field in the custom object, but select Bind Value, which enables you to select a field in the associated business object to compare to.
Custom Role	This is a field which contains a role name in any custom object with a filter to narrow down to a single row. The behavior is the same as Custom Queue.
Custom User	This is a user lookup reference field in any custom object with a filter to narrow down to a single row. The behavior is the same as Custom Queue.

Option	Description
Queue	Use this to dynamically assign an approval request to a queue. If email notifications are set for the queue, notifications will be sent to all members of the queue, otherwise, a single notification will be sent to the queue email.
Related User	This is a user lookup reference field in the associated business object (i.e. Agreement or Opportunity).
Role	A role can be based on your company's hierarchy and can have one or more users assigned to it.
Rule	Use this assignee type when you want to dynamically assign an approval request based on an approval rule. If you select this option, the approval request is sent only to the first user that satisfies the filter criteria in the Approval Rule. If you have multiple approvers in a sequence, only the first approver is selected and assigned.
User	Use this assignee type when you want to dynamically assign an approval request to a user.

9. Select **Send Email** to notify the assignee by email that they must approve or reject the object record and select **Notify Only** to only inform the assignee by email that the approval process has reached this step, without any actions to take.
10. Select **Skip Unresolved Assignee** to optionally skip an unresolved approver from the list of sequential approvers.
11. Choose **Step Auto Escalate** if you want to escalate the current step if no action is taken on the approval request within a specified period. Enter the number of days in **Step Expected Days To Complete and** the number of hours in **Step Expected Hours To Complete** by which you want to escalate the approval request.
12. Select a mandatory **Assignee Type** where you can escalate the approval request to a User, Role or any of the assignee type mentioned in Step 8. Click **Next**.
13. Create Reminders for approval requests that send a reminder notification to the assignee at the specified duration unit and intervals. For example, you can remind the assignee to take action on the request 7 days after it was assigned, and then again a week later. To activate reminders, select **Step Reminders Active**.
    Reminders are sent to the original assignee only. Reminders are not sent if you reassign or escalate the approval request.

> As a prerequisite, you must schedule reminder jobs in the respective environment to run a reminder flow. Run these jobs in the development console as per requirement. The default time interval between each job is 15 minutes.
> Example jobs:
> system.schedule('Approvals Reminder Job 1', '0 00 * ? * * 2019', new Apttus_Approval.ReminderJob());
> system.schedule('Approvals Reminder Job 2', '0 15 * ? * * 2019', new Apttus_Approval.ReminderJob());
> system.schedule('Approvals Reminder Job 3', '0 30 * ? * * 2019', new Apttus_Approval.ReminderJob());
> system.schedule('Approvals Reminder Job 4', '0 45 * ? * * 2019', new Apttus_Approval.ReminderJob());

14. If you have enabled submission comments at the process level, at each step, you must enable submission comments.
15. Select the Header and Display Fields which will appear at each approval step, by highlighting steps in **Unselected Items** and click ☐ to add them to the **Selected Items**. Reorder the steps as required and click **Next**. Step dependency is based on their order in **Selected Items**. For example, the selected Display fields appear on the Approvals page if you have configured Approvals for CPQ.
16. Select CC (carbon copy) assignees who are copied on email notifications for all approval requests assigned for each Approval Step. The CC assignees are set up regardless of the approval step type. This works for standard, sub-process, and child process. CC can be a Named User. A CC assignee is only copied on email notifications, they cannot approve a request by replying to the email. Their reply email is bounced back. In case of consolidation, CCs from all request are concatenated.

    > ⚠ Custom Users and Queues are not supported.

17. Click **Save**.

The step is added to the **Approval Steps** section for the Approval Process. You can now **Activate** the approval process.

To create an Approval Rule to use with a Sub-process or Child processes

Condition must be added to the *Rule Type* picklist.

Go to **Setup > Create > Objects > Approval Rule**, scroll to **Custom Fields & Relationships** and click **Rule Type**. From **Picklist Values** section, click **New**.
Enter `Condition` and click **Save**.

1. Go to **Approval Rules** and click **New**.
2. Select the Business Object and click **Next**. To use the approval rule with a sub-process it must match the context object of the approval process. To use it with a child process, it must match the child object associated with the approval process.
3. Enter a **Rule Name, Sequence,** and from **Rule Type**, select **Condition**.
4. Click **Save** and then click **Manage Entries** from the *Approval Rule Detail* page.
5. From Rule Entries, click **New**.

6. Enter the condition you want to use to route an approval request to the assignee and click **OK**.
7. Now you can fill out the rest of the rule entry as normal, including descriptions for both the condition and the assignee.
8. When selecting assignees, the **Type** you select impacts what you need to enter for the **Value**:
    - *User*, *Role*, *Queue* - Enter the *Label* for the specific object record you are referring to.
    - *Related User* - For custom fields enter the *API Name*, such as `QuoteOwner___c`. This needs to be the *User Lookup* field on the *Line Item* or *Product Configuration* object record. If it is a standard field, add `Id` to the field name and use that. For instance, with the Line Item field *CreatedBy*, you would enter `CreatedById`.
    - *Custom User*, *Custom Queue*, *Custom Role* - This can be a formula, using a format as follows: `Custom Object (Custom Object Field 01 = "Value" AND Custom Object Field 02 = "Value") User`. For more details, see Custom Assignee Example.
9. Add additional conditions and assignees as desired and click **Save**.

> **Note**
>
> The *Assignee Type* is *Subprocess* regardless of whether you created a Subprocess or Child process type.

When the approval rule is *Active* it can be used with an approval process and depending on the context object of the approval process, the approval rule will be available when you select a *Step Type* of *sub-process* or *child process*.

You can create additional approval rules as needed.

Using Date Criteria in Approval Rule Entry

If you use date based criteria in an approval rule entry, you must YYYY-MM-DD 00:00:00 format for the date. For example, you want to trigger an approval rule on Proposal Line Item with Create Date greater or equal to April 5, 2018.
Use the following and do not delete anything:2018-04-05 00:00:00
This is useful when you want to set an approval rule or entry to take effect on or after a certain date. If you do not use the right format, the system displays an error when you try to submit for approval OR the system will simply ignore the criteria and trigger the rule regardless of the date.

Setting up Auto-escalation

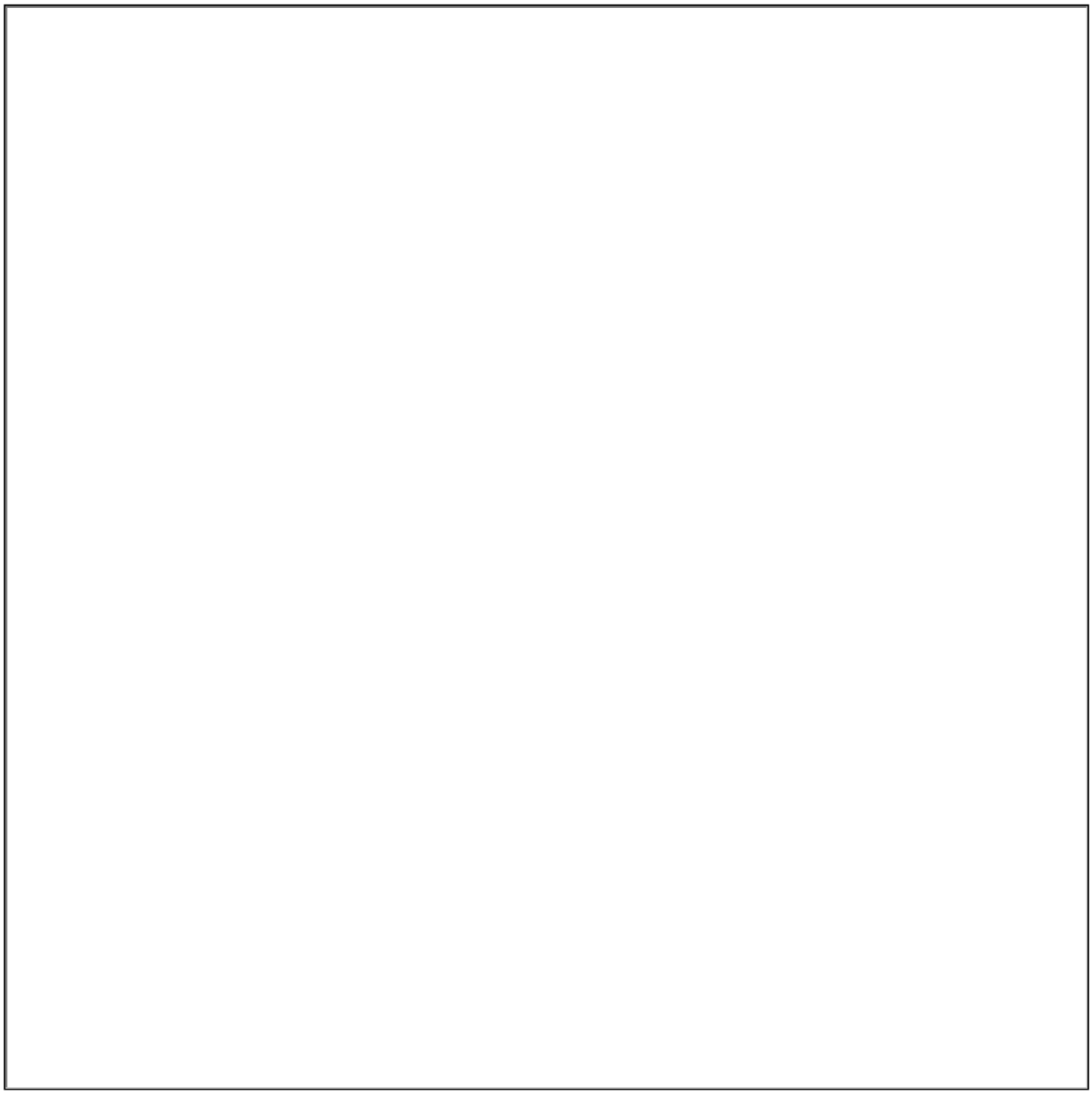
Enabling auto-escalation in an approval process occurs at the approval step level. While the email templates apply to the entire approval process, you can have different auto-escalation settings for each approval step or you can use the global escalation options.

Once an approval process has started, you can choose to manually escalate an approval request from the *Approval Request* related list for the object record currently under approval.

To configure auto-escalation for an approval process

You must have completed the configuration tasks to enable auto-escalation functionality.

1. Create a new approval process or edit an existing one.
2. When you come to the Notification Templates page, you can leave the **Escalation Email Template** field empty and it will use the default template or you can override it with a custom template.



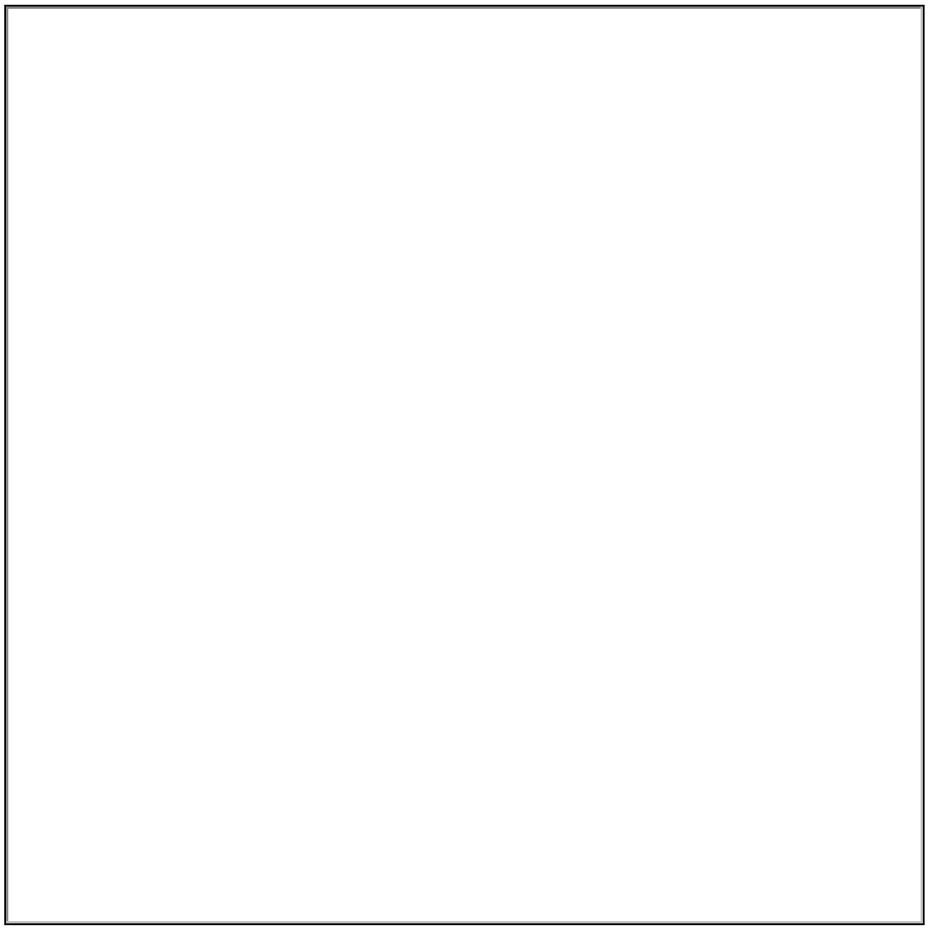
3. Finish creating the approval process and then create or edit an approval step for it.
4. When you come to the Select Escalation Actions page you must select **Step Auto Escalate**. Configure the settings as follows:

Option	Description
Expected *nn* To Complete	These values are used to trigger the auto-escalation. Once an approval request has been created, the system uses the value of expected time to complete to determine when the request should be auto-escalated.

Option	Description
Assignee Type	While you could include any assignee type, the following behavior should be noted. If you select *Related User*, the assignee name can only be a single word otherwise it cannot be reassigned correctly. Also be aware of the differences between selecting a single-step and multi-step assignee type.
Assignee Name	This value is dependent on the assignee type.

Note

This page is displayed regardless of whether the auto-escalation feature is enabled.



5. Save the approval process step to complete the procedure.

When approval requests for that specific approval step exceed the **To Complete** values, they will be auto-escalated per the escalation settings.

To manually escalate an approval request

The configuration tasks must have been completed.

1. Go to an object record that you have submitted for approval. Only the user who initially submitted the approval, or system administrators, can manually escalate a request.
2. From the **Approvals** related list, select **Escalate**.

The request is immediately escalated, according to the auto-escalation settings.

Creating Expressions

Approval process expressions provide a method for qualifying whether an object record should be routed through specific approval steps.

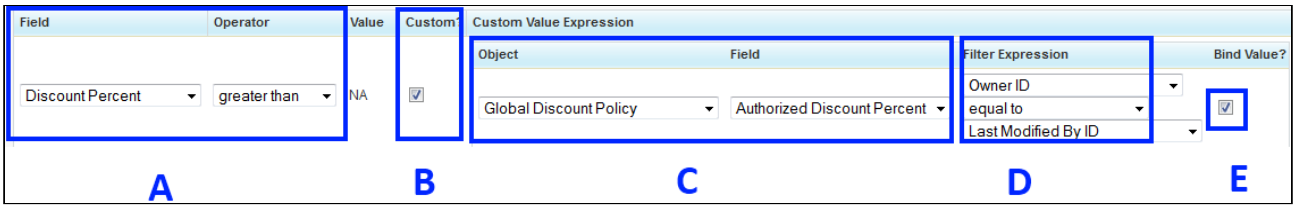
To connect individual expressions together using *OR*, as well as *AND*, you must use Advanced Options. For details, see Creating Advanced Options.

There are three types of expressions: standard, advanced and bound.

Expression	Description
Standard	This uses a field from the object record in the approval process, along with an operator, to compare to a single value to see if the process should enter a certain approval step. The value can be a constant value, whether entered manually or selected from a picklist, depending on the field type.
Advanced	This builds upon the simple expression, by enabling the value to come from a custom value expression. The custom value is derived from building another equation using a custom field in a custom table.
Bound	The same as advanced, except that the filter expression can use a field from the business object record in the approval process.

Your criteria can contain a simple field, operator, and value, or the value can be composed of a "custom" expression evaluated at runtime which includes resolving a value from a custom table, a custom field in that table, and an optional filter expression to narrow the rows down in that table. Custom filter expressions can also use bind variables that substitute the value of a field from the context object at runtime.

Bound Expression



A	The field comes from the business object associated with the approval process. Along with the operator, this forms the typical first two thirds of an equation.
B	Selecting **Custom** enables the **Custom Value Expression** section for creating advanced and bound expressions. It also sets **Value** to a read-only *N/A*.
C	The **Object** is a custom Salesforce object containing the available fields. The value selected for **Field** is what is compared to the field in part *A*. At its most basic form it is those two fields and the operator that form the expression. The expression will be evaluated for each row in the custom object.
D	The **Filter Expression** can be used to limit the number of rows in the custom object which are used as part of the overall expression. Any of the fields in the custom object can be used to create a filter expression, independent of the overall expression being evaluated.
E	Selecting **Bind Value?** creates a bound expression and makes the second field used in the Filter Expression come from the same business object as the field in part *A*. The **Bind Value** check box identifies the following: • A specific field on the custom object and compares it to a field on the context object. • A unique record on the custom object that needs to be used in the step entry criteria.

Attention

When building your expressions – both the Custom Value and Filter Expression – you must ensure the data types for the values in the fields being compared match. If they do not match, an error will occur when you preview the process or submit a request.

To create standard expressions

Must be in *Step 2: Specify Step Entry Criteria* when creating or editing an Approval Process Step.

1. From the **Field** list, select the required object record field you want to compare.
2. Select the **Operator** and enter a valid **Value**.
3. Click ✚ or ▬ to add and remove additional field expressions. By default, each line is linked with *AND*, click **Advanced Options** to use *AND*, *OR*, and parenthesis.
   See Creating Advanced Options for details.
4. Click **Save** once all the expressions have been added.

To create advanced and bound expressions

Must be in *Step 2: Specify Step Entry Criteria* when creating or editing an Approval Process Step.

1. From the **Field** list, select the required object record field you want to compare and select the **Operator**.
2. Select **Custom** to display the **Custom Value Expression** options.
3. Select the required options:

Option	Description
Object	This is the custom Salesforce object that is a table containing fields that are used for the expression. **Note** These must have been added via Approvals System Properties. See Setting up Custom Assignee Value Expression Objects for details.

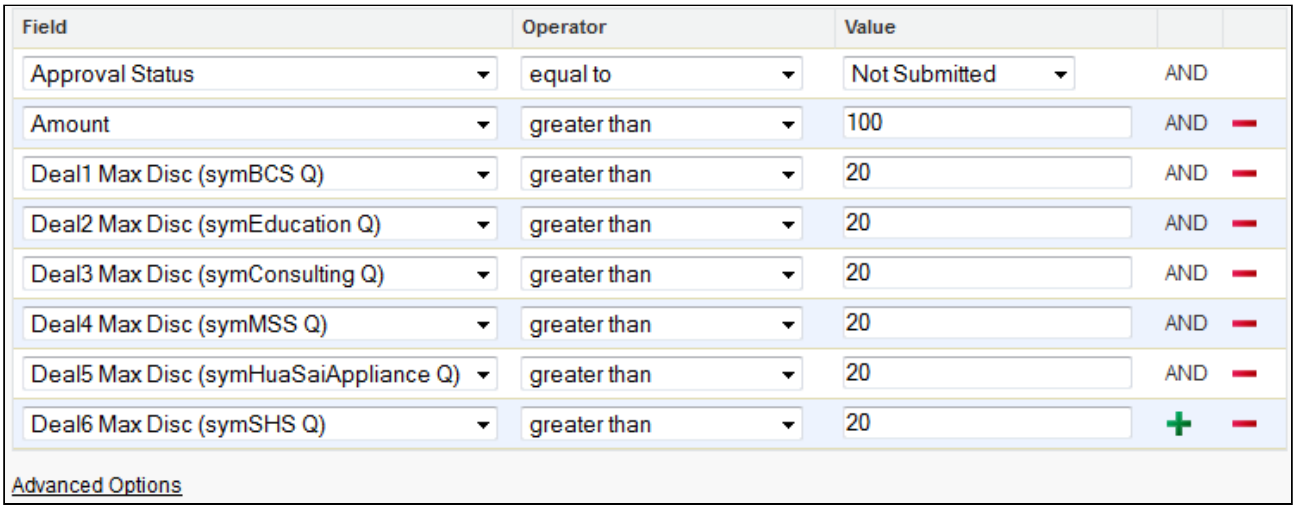
Option	Description
Field	This is the field in the custom object which will be compared against the field and operator selected in step 1.
Filter Expression	These two fields and operator act as an independent expression within the larger overall expression.
Bind Value	This enables the third **Filter Expression** picklist to be populated with values from the same object record that is being used in the main Field picklist.

4. Click ✚ to add another line.
   - Or -
   Click **Next** or **Save** to leave the page.

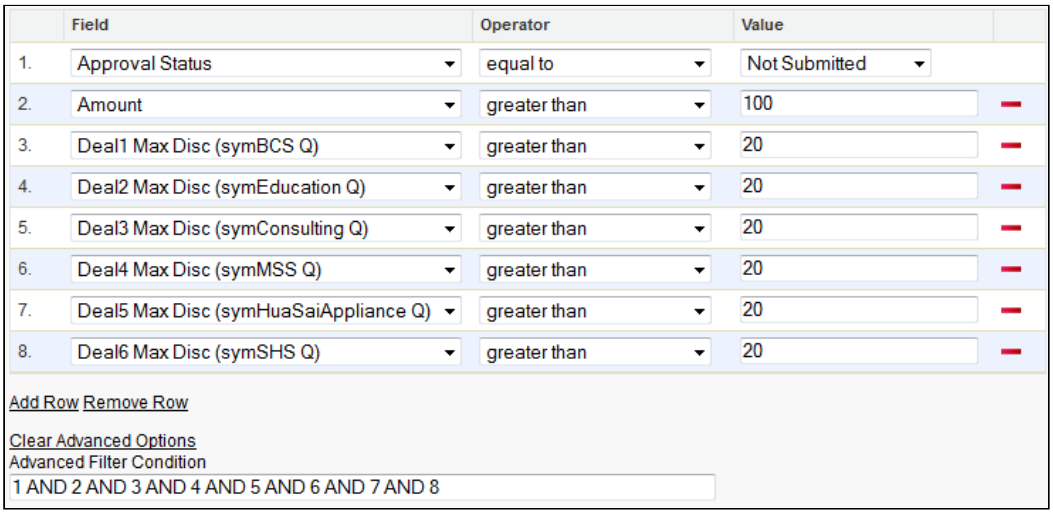
Creating Advanced Options

By default, when you create expressions each of them is associated by *AND*. This means the criteria for each expression must be met before the approval process will be routed according to the entry criteria, approval step criteria, or term exception approval conditions.

The **AND** column is read-only and cannot be changed. You must select **Advanced Options**.

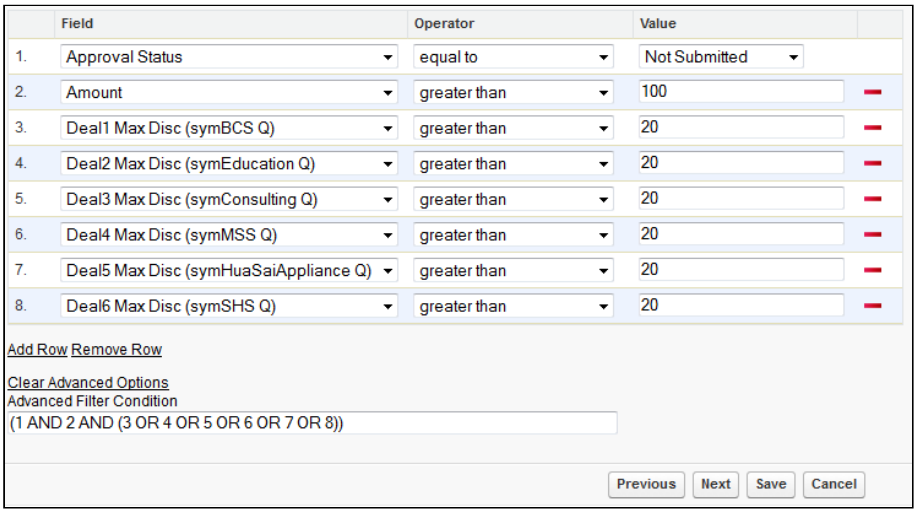


When you select **Advanced Options** the **AND** column is removed, a number is prepended to each expression, and the **Advanced Filter Condition** field is displayed. It displays each of the expressions – identified by their number – joined by the default *AND*.



In the **Advanced Filter Condition** field you can use any combination of *AND*, *OR*, parenthesis, and numbers representing the expressions. These are the only values that can be used. Using other values may cause unexpected behavior.

As it is a free text field, there is no validation on the characters you can enter in the field. You can use up to 255 characters in the field.



To set advanced options

You must be in one of the following custom object sections:

- **Approval Process > Entry Criteria**
- **Approval Process > Approval Steps > Step Entry Criteria**

1. After you have created the required expressions, select **Advanced Options**. The **Advanced Filter Condition** field is displayed with the expressions listed in sequence, linked by *AND*.

2. In the **Advanced Filter Condition** field, create the equation using only the numbers representing the expressions, *AND*, *OR*, and parenthesis.

Option	Description
Valid example	((1 OR 2) AND 4) OR 3
Invalid example	((1 > 2) & (three or 4)) AND 5

3. Click **Next** or **Save** depending on which custom object you are in.

The expressions are linked and evaluated according to the formula in the Advanced Filter Condition field. There is no auto-validation on the formula. You should test the resulting criteria before using it in a live environment.
To remove the advanced options and return to expressions that are all linked by *AND*, click Clear Advanced Options.

Initiating an Approval Request

If the object for which you have defined the approval process fulfils the search entry criteria and the step entry defined in the approval process, an approval request is initiated. All line items that require an approval, show up with an approval required label. To view the object and line items that require an approval you can use the Approval button on the object, to view a list of approvals required along with the approvers.

Setting up Approval Rules

Approval rules and dimensions enable you to package approval routing criteria together and use them for handling the *Assigned Approvers* step in the Approval Steps of your Approval Process.

Typically if you choose to assign approval responsibilities to a specific user or role, there is no further logic or criteria that can be put into that assignment. The only real criteria available for the approval step is the *Step Entry Criteria*. By associating rules at the assignee step, via approval rules, you can dynamically assign approvers based on the same object records information used for the entry criteria.

- Dimensions
  Dimensions enable you to map fields from a variety of business objects including, but not limited to, Opportunity, Quote/Proposal, and Agreement Term Exception.
- Approval Rules
  Approval rules are used to associate Salesforce business objects, approval rule

dimensions, and entry criteria for deciding which assignees are required during the approval workflow.

Dimensions

Dimensions enable you to map fields from a variety of business objects including, but not limited to, Opportunity, Quote/Proposal, and Agreement Term Exception.

Once a dimension has been created, you can create an approval rule and associate it with the dimension.

To create approval rule dimensions

1. Click the **Approval Rule Dimensions** tab or click ☐ and then select **Approval Rule Dimensions**.
2. Click **New** and then enter the dimension information:

Option	Description
Dimension Name	Enter a meaningful name, which will make it easy to identify and select in the approval process.
Business Object	This should match the object in the approval process, that you plan to use the approval rule with. The objects you can select from are dependent on the packages you have installed.
Dimension Type	Field is a field associated with the business object. Formula is a type which can be extended through the associated business object.
Field	The fields associated with the business object are available if the Dimension Type is Field. If the Dimension Type is Formula, then enter a formula with a reference field syntax (nnn__r.area__c), such as *Approval_Submitter__r.Area__c*.
Description	Description for the approval rule dimension.

3. Click **Save** to go to the Approval Rule Dimensions details page - or - Click **Save & New** to add another dimension.

The dimension is added to the list on the Approval Rule Dimensions page.

You can now create approval rules and associate them with the dimension. Either from the dimension details page or from the Approval Rules tab.

Approval Rules

Approval rules are used to associate Salesforce business objects, approval rule dimensions, and entry criteria for deciding which assignees are required during the approval workflow.

To create approval rules

Dimensions must exist, so that the approval rule can be associated with one.

1. From the Approval Rule Dimension page, click **New Approval Rule** in one of the **Approval Rules (Dimension)** sections. - or - Click and then select **Approval Rules** and click **New**.
2. Select the **Business Object** that is the same as the one associated with the approval process this rule will be used with and click **Next**.
3. In the **Configure Rule** section, enter the minimum required settings for the rule:

Option	Description
Business Object	This is the value selected in Step 2 and cannot be edited here.
Rule Name	Mandatory name for the rule.
Sequence	Mandatory number indicating in which sequence the rule will be evaluated.
Rule Type	Select **Dimension** or **Condition** for this mandatory setting.
Description	Description of the rule.

Option	Description
Dimensions	Select the dimensions you want the rule to be used with. The available dimensions are based on the business object selected in Step 2. There can be a maximum of six dimensions.
Dimension Types	Select **Discrete** or **Range**. Discrete must be a specific value, such as a number or a value for a category. The value *Rejected* in the category *Approval Status* would be discrete. Range is continuous data, where a value can be measured to multiple degrees. For example the number of units for a product could fall into a range of 1-99, 100-199, 200-499 and 500 or more.

> **Note**
>
> You should not use the Rule Criteria section, as this kind of logic is typically handled by Approval Step logic.

4. Select **Active** so the rule can be used with approval processes.
5. Click **Save** to go to the Approval Rule details page - or - Click **Save & New** to add another rule.

The rule is added to the list on the Approval Rules page.

From the Approval Rules detail page, you should add assignees, using the **Manage Entries** option.

Approval Rule Assignees

The Manage Entries section of an Approval Rule designates assignees who will receive approval requests, based on the associated dimension and where they are called within an approval step.

You can have a user, role, queue, custom version of those, related user, or auto approval as an assignee. Like configuring assignees in the approval step, you can indicate whether they should only be notified or sent an email informing them they must decide on the approval.

If you want to automatically progress an item, you can select Auto Approval as an assignee and associate the rule with a *Subprocess* in an Approval Step. In instances where approval routing is based on thresholds, an Auto Approval assignee can ensure the process is handled correctly.

In instances where approval routing is based on thresholds, an Auto Approval assignee can ensure the process is handled correctly. For instance, if approval routing is based on discount percentages and no approval is required for the 0-20% range, the approval step for that range could go to an Auto Approval assignee.

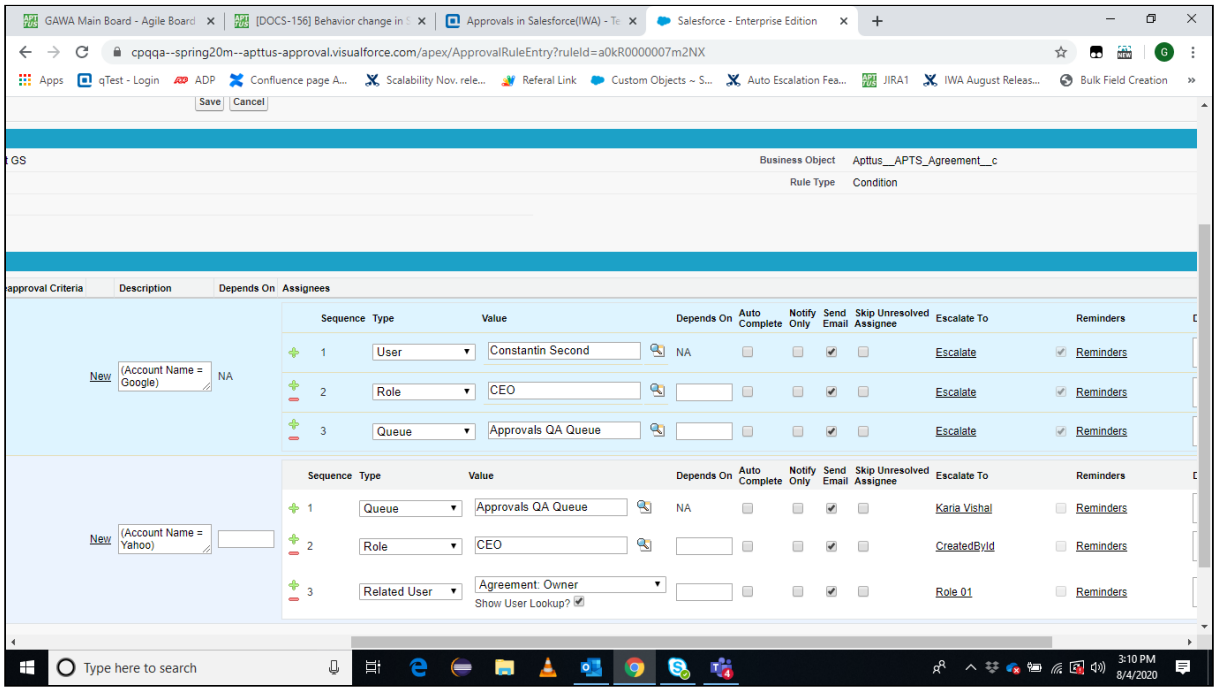
Note

You must ensure valid values are entered or the expected assignees may not get notified as expected. While you can enter someone's name, username, or email address, it is recommended that you either use the username (that they login to the system with) or the email address associated with their account. As these should be unique, you can best ensure you have selected the correct user.

To set the assignees

There must be an Approval Rule.

1. From the Approval Rule Detail page, click **Manage Entries**.
2. Select the criteria, based on the previously selected dimensions.



3. Enter the assignee details:

Option	Description
Sequence	If you are going to use this rule with a *Subprocess* in an Approval Step, you can select to add another assignee, otherwise do not add more.
Type	Choose from **User, Role, Queue, Related User, Custom User, Custom Role, or Custom Queue.** or **Auto Approval**. ⚠ Type Auto Approval is deprecated. Conga recommends that you use the Auto-Complete checkbox instead.
Value	You need to enter a valid Salesforce value for the assignee, based on the Type selection.
Label	Enter a label to associate with the Value.

Option	Description
Depends On	If you have multiple assignees for the rule entry, enter the number corresponding to the assignee who you want to make the current assignee dependent on.
Notify Only	An email is sent to the assignee, without any action items.
Send Email	An email is sent to the assignee, informing them they must make an approval decision.
Auto Complete	The approval request is auto-approved on submit action.
Skip Unresolved Assignee	Select to optionally skip an unresolved approver from the list of sequential approvers.
Escalate To	If the assignee does not approve or reject a request within a stipulated frame of time, the request is escalated to the specific assignee.
Description	Description for the assignee.

4. Click ☐ to add another rule entry - or - Click **Save** to go to the Approval Rule Detail page.

The rule entries are now in use by the approval rule.

Now that the approval rule has assignees, you can create an approval process and reference the approval rule and it will use values entered here to route the approval.

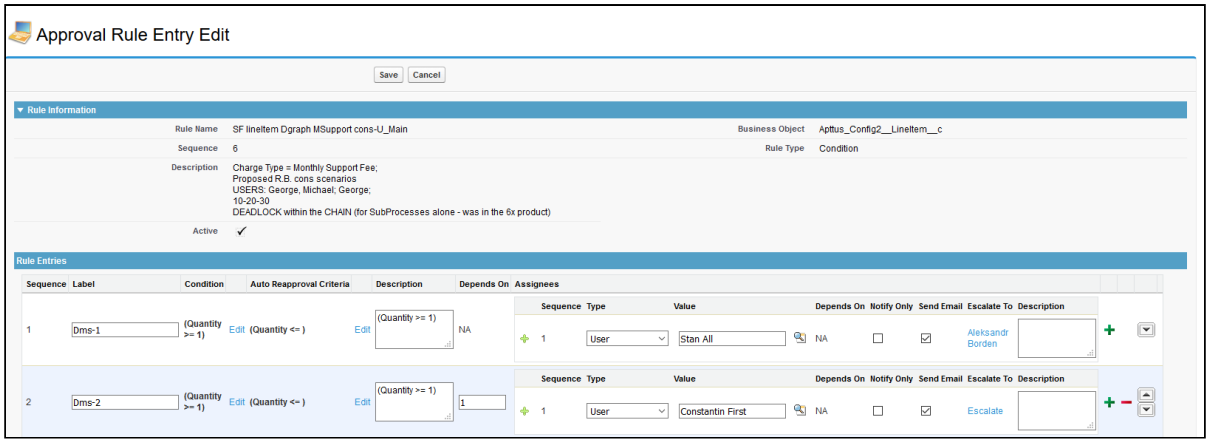
Setting up Auto-escalation for Rule Assignees

Important

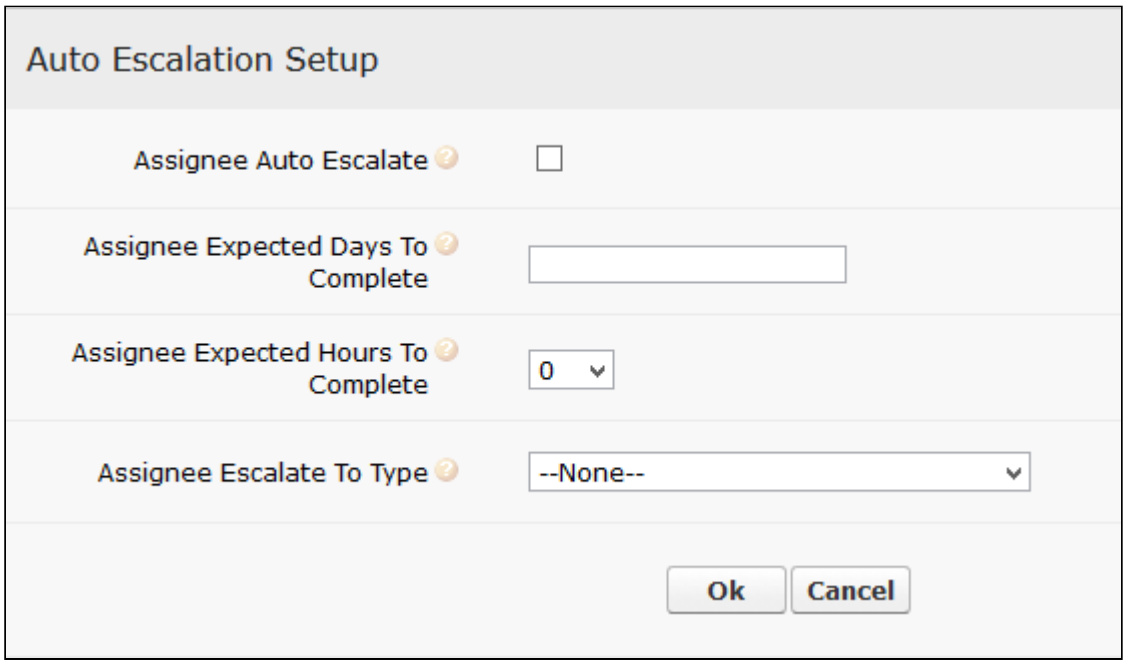
> Auto Escalation should be set-up on approval rules that will be used as sub-process (context header object) ONLY. Auto escalation should not be setup on child process (child records or line items).

To setup auto-escalation for assignees, there must be an Approval Rule.

1. From the Approval Rule Detail page, click **Manage Entries**.
2. Select the criteria, based on the previously selected dimensions.



3. Enter the assignee details.
4. Click **Escalate To**. The Auto Escalation Setup dialog appears.



5. Enter the escalation details:

Option	Description
Assignee Auto Escalate	Select this check box to enable auto escalation for the specific assignee. If the Assignee does not approve or reject the request within the specified days or hours, the request is escalated.
Assignee Expected Days/Hours To Complete	Specify the days or hours within which the assignee should approve or reject the request. If no action is taken on the request, it is escalated to the assignee specified in the Assignee Escalate To Type field.
Assignee Escalate To Type	Specify the assignee you want to escalate the request to. The Assignee type can be user, role, queue, backup admin, user manager, delegated user, related user, User Approval Matrix Next Approver

Approval Policies

Approval policies enable you to set the number of approved requests required for a process to be considered approved, based on unanimity, majority, or percentage. This is done by selecting a policy type for an approval rule and associating it with an approval process through a subprocess assignee.

When there are multiple approvers for a sub-process, as the approvers complete their requests the system uses the policy to determine when the approval process is finished. When an approval process reaches the approval or rejection threshold, any remaining approval requests do not need to be completed.

For example, if you have a majority policy with five approvers and the first three approvers approve the approval request, there is no need for the remaining two approvers to answer the request. 3 out of 5 have approved it to reach a majority and the Approval Status for the object is set to Approved. Even if a single approver rejects the request, the approval status for the object is set as Rejected.

There are four policy types:

Unanimous	All approvers must approve their requests for it to be approved. This is the same as the default Approvals behavior.
Majority	Majority of approvers must approve the request for it to be approved. To ensure this policy evaluates as expected, you should ensure you have an odd number of approvers. If you have an even number of approvers and you end up with the same number of approvals and rejections, the request is rejected.
Percent	The designated percentage must be met or exceeded for the request to be approved.
Quorum	The designated number must be met or exceeded for the request to be approved in the **Approval Count** field. For example, if you set up **Approval Count** as 2, and you have 4 approval requests, out of which 2 of your approval requests are approved, then the Approval Status is set to *Approved*. If any 2 of your approval requests are rejected, the system waits for the next 2 approvers to take the decision. If the remaining 2 approvers approve the request, the Approval Status is set to *Approved*. **Note** The above system behaviour is functional when you have selected the **Continue Policy Approval on a Reject** check box.

Note

You cannot use approval policies and consolidated approvals together. You can only use one or the other in an approval process.

From the May Patch 1 release onwards, once an assignee rejects an approval request, you can choose to proceed to send approval requests to the rest of the assignees in the policy despite a rejection. In the earlier releases, the policy status would be set to **Rejected** even if a single assignee in the rule rejected a request.

The Approval policy (supported for Sub Process or Child Process steps) supports a new feature to **Continue Policy Approval on a Reject** when a policy is enabled.

- When checked, this feature allows the user to approve their assigned approval requests, as long as the policy criteria can still be met, even though one or more approval requests within the Sub process or Child process are already rejected.
- When unchecked, all the approval requests with status other than Approved, within the Sub Process or Child Process will be cancelled, whenever any request is rejected. Policy approval for that approval Sub Process or Child Process will stop immediately.

The **Continue Approval Policy After Rejection** check box takes precedence over the **Continue Pending Approvals on a Reject** check box, which is set at the process level.

The table highlights the behavior for the **Continue Approval Policy After Rejection** check box.

Policy Type	Behavior
Unanimous	All the assignees must approve the request for the status to be approved. Even if a single approver rejects the request, the Approval Status is Rejected. Continue Approval Policy after Reject (selected): All the assignees are routed the approval request, the approval request cycle continues even after a rejection. Continue Approval Policy after Reject (de-selected): The approval request cycle stops after a rejection. The Policy status is set to Rejected.
Majority	Continue Approval Policy after Reject (selected): All the assignees are routed the approval request, the approval request cycle continues even after a rejection. If the majority of the approvers have approved the request, the status is Approved. Continue Approval Policy after Reject (de-selected): The approval request cycle stops after a rejection. The Policy status is set to Rejected.
Quorum	Continue Approval Policy after Reject (selected): All the assignees are routed the approval request, the approval request cycle continues even after a rejection. If the specified quorum of the approvers have approved the request, the status is Approved. Continue Approval Policy after Reject (de-selected): The approval request cycle stops after a rejection. The Policy status is set to Rejected.

Policy Type	Behavior
Percent	Continue Approval Policy after Reject (selected): All the assignees are routed the approval request, the approval request cycle continues even after a rejection. If the specified percentage of the approvers have approved the request, the status is Approved. The approval request cycle stops after a rejection. The Policy status is set to Rejected.

To create an approval policy

1. Go to the **Approval Rule Dimensions** tab and create a dimension.
2. Go to the **Approval Rules** tab and create a rule. Create a rule per usual, but from the **Approval Policy** list, select from **Unanimous**, **Majority**, **Quorum** or **Percent**.
   If you select *Percent*, an additional field is displayed for entering the percentage threshold for requests, to have the Approval Status reach Approved.
   If you select *Quorum*, 2 additional fields are displayed.
   Approval Count for entering the designated number of approval requests to set the Approval Status to Approved. **Continue Policy Approval on a Reject** check box enables you to let the approvers decide even when the designated number of approvers have approved or rejected an approval request.

   
3. Go to the **Approval Process** tab and create a process. When you create an Approval Step, in *Step 4: Select Assigned Approvers*, from **Assignee Type** select **Subprocess** and from **Assignee Value**, select the approval rule created earlier.

The approval process is in place to make use of the policy the next time an approval request for an object meets the entry criteria is initiated.

If you select Percent, an additional field is displayed for entering the percentage threshold for requests, to have the Approval Status reach Approved.If you select Quorum, 2 additional fields are displayed. Approval Count for entering the designated number of approval requests to set the Approval Status to Approved. Continue Policy Approval on a Reject enables you to let the approvers decide even when the designated number of approvers have approved or rejected an approval request.

Approvals for SOAP API Developers

This section describes how to use Application Programming Interfaces (APIs) to extend the features offered by Conga. These extensions add more functionality to the features available through configuration on Salesforce.

Topic	Description
What's Covered	This section provides information on how you can use Approval APIs to extend and customize the features offered by Conga Approvals.
Primary Audience	Administrators responsible for setting up approvals and users for Approvals.
IT Environment	For information pertaining to the requirements and recommendations, see System Requirements and Supported Platforms Matrix.
Updates	For a comprehensive list of updates to this guide for each release, see the What's New in Approvals Documentation topic.
Other Resources	• See Approvals for Administrators for basic admin tasks and end-user experience. • See Approvals for Users for information on Approvals workflow and use cases.

This section describes the APIs provided to work with Approvals objects.

Before using Approvals, 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:

- SOAP API Guide Structure
- Integrating Conga with External Systems
- IWA Web Service

SOAP API Guide Structure

Document Setup

The CPQ and CM 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

Conga 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 (WDSL) 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

Conga SOAP API works with standard SOAP development environments. For a list of compatible development platforms, see Salesforce Developer Force API details.

⚠ A later version of this content will contain sample code.

Field Types

Conga 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.
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.

Type	Description
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.

Integrating Conga with External Systems

Additional steps are required when you choose to integrate Approvals on Salesforce with external applications, customer portals, or other critical business systems. Because Approvals 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**
>
> Conga recommends that you work with Conga Professional Services to design and implement your integration. Use this documentation for basic integration steps and to reference Intelligent Workflow Approvals Web Service calls.

The following basic steps are required to get started using the Conga Intelligent Workflow Approvals 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 Conga Professional Services. Refer to Salesforce Documentation for complete instructions on generating the Web Service WSDL.
2. **Generate the Conga Web Services WSDL** – After you have connected to Conga Web Service, go to your organization and download the WSDL for the appropriate Web Service.
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 Intelligent Workflow Approvals 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:

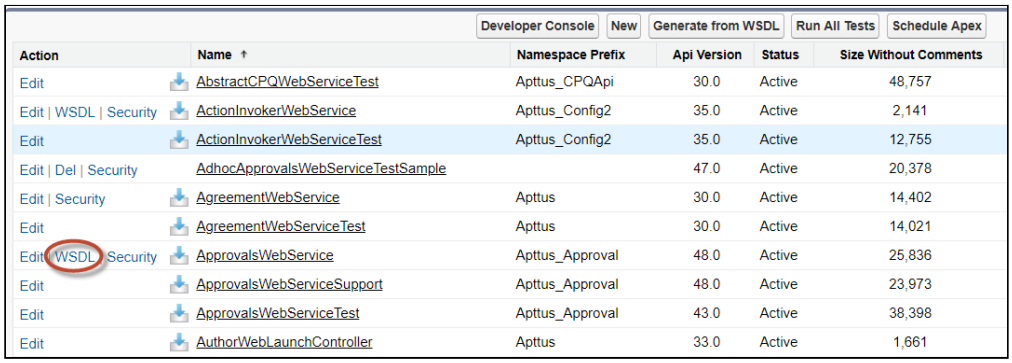
- https://success.salesforce.com/issues_view?id=a1p3A000000eatxQAA&title=generated-wsdl-for-apex-webservices-is-malformed
- https://success.salesforce.com/issues_view?id=a1p300000008XKUAA2

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 you 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, **ApprovalsWebService**).

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 Intelligent Workflow Approvals 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.

IWA Web Service

Conga IWA APIs are based on Salesforce APIs and use the same standards and platforms.

You can invoke Conga IWA APIs belonging to the ApprovalsWebService Class from the following command:

```
Apttus_Approval.ApprovalWebService.<Name of the Function>
where the name of the function is API Name and it parameters.
```

Approvals Web Serice class contains the following APIs:

- Adding Comments to Approval Requests
- Approving Approval Requests
- Cancelling Approvals
- Checking If an Approval is Required
- Checking if User is Authorised to Approve or Reject
- Creating Ad-hoc Approvals
- Enabling Auto Re-approvals on Proposal and Proposal Line Items
- Escalating Approval Requests
- Previewing Approvals
- Reassigning Approval Requests
- Rejecting Approval Requests
- Retrieving Add Comment Page URL
- Retrieving Approval History
- Retrieving Approval Page URL
- Retrieving Approve or Reject Page URL
- Retrieving Reassign Page URL
- Submitting for Approvals
- Taking Ownership of an Approval Request
- Previewing Adhoc Approvals
- Submit For Approvals Using An Adhoc Approval Specification
- Submit For Approvals With Comments Using An Adhoc Approval Specification
- Delete An Adhoc Approval Step
- Adhoc Approval Process Runtime APIs
- Submitting Bulk Approval Requests
- Recalling Bulk Approval Requests
- Submitting Approvals Asynchronously

Adding Comments to Approval Requests

This API enables you to add comments to approval requests. This API accepts the ID of the approval request and comments as input parameters.

API	Signature
addCommentsToRequest	*webService static Boolean addCommentsToRequest(Id requestId, String comments)*

Request Parameters

Name	Type	Required?	Description
requestId	ID	Yes	ID of the approval request.
comments	String	Yes	Comments to add to the approval request.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the comments are added to the approval request.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Header>
       < app :SessionHeader>
          < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
       </ app :SessionHeader>
    </ soapenv :Header>
    < soapenv :Body>
       < app :addCommentsToRequest>
          < app :requestId>a0oR0000006cqc6</ app :requestId>
          < app :comments>Approve</ app :comments>
       </ app :addCommentsToRequest>
    </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Body>
       < addCommentsToRequestResponse >
          < result >true</ result >
       </ addCommentsToRequestResponse >
    </ soapenv :Body>
</ soapenv :Envelope>
```

Approving Approval Requests

This API approves the approval request. This API accepts the ID of the approval request and comments as input parameters.

API	Signature
approveRequest	*webService static Boolean approveRequest(Id requestId, String comments)*

Request Parameters

Name	Type	Required?	Description
requestId	ID	Yes	ID of the approval request.
comments	String	Yes	Comments to add to the approval request.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the approval request is approved successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>

   < soapenv :Body>
      < app :approveRequest>
         < app :requestId>a0oR0000006cqc6</ app :requestId>
         < app :comments>Approve</ app :comments>
      </ app :approveRequest>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
  xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < approveRequestResponse >
         < result >true</ result >
      </ approveRequestResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Approving Bulk Approval Requests

With this API, you can approve bulk approval requests. This API accepts the object type and the object ID of the context object as input parameters. You can also add comments to your approval requests. You cannot approve approval requests with attachments.

If you submit previously submitted or approved approvals as part of bulk approval request, the approval request is submitted again.

This API is useful when you need to approve a large number of approval requests for various context objects. For example, you are the approvals admin for a data service organization and you need to approve 5 agreements. Using submiForApprovalsBulk API, you can submit all the approval records in one go by calling this API and providing the type and IDs of the five agreements as request parameters.

API	Signature
approveRequestsBulk	*webService static Id approveRequestsBulk(List actionRequests)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
actionRequests	List	Yes	List of requests to bulk approve

Response Parameter		
Name	**Type**	**Description**
Id	ID	

```
// multiple context object type and ids
String agmtType =  'Apttus__APTS_Agreement__c' ;
ID agmtId1 =  'a07R000000APxGG' ;
ID agmtId2 =  'a07R000000AS9nJ' ;
ID agmtId3 =  'a07R000000AS9nO' ;
```

```
// submission comments
Apttus_Approval.SubmissionComments submitComments1 = new
 Apttus_Approval.SubmissionComments();
submitComments1.commentsLevel =
Apttus_Approval.SubmissionComments.PROCESS_LEVEL_COMMENTS;
submitComments1.commentsCount = 1 ;
submitComments1.processName = 'ProcessName' ;
submitComments1.processCommentLabel = 'ProcessLabel' ;
submitComments1.processCommentMandatory = false ;
submitComments1.processComment = 'Test Comment 1' ;

Apttus_Approval.SubmissionComments submitComments2 = new
 Apttus_Approval.SubmissionComments();
submitComments2.commentsLevel =
Apttus_Approval.SubmissionComments.PROCESS_LEVEL_COMMENTS;
submitComments2.commentsCount = 1 ;
submitComments2.processName = 'ProcessName' ;
submitComments2.processCommentLabel = 'ProcessLabel' ;
submitComments2.processCommentMandatory = false ;
submitComments2.processComment = 'Test Comment 2' ;

Apttus_Approval.SubmissionComments submitComments3 = null ;


// create list of context objects
List<Apttus_Approval.BulkCtxObjectParam> ctxParams = new
 List<Apttus_Approval.BulkCtxObjectParam>();

Apttus_Approval.BulkCtxObjectParam ctxParam1 = new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam1.ctxSObjectType = agmtType;
ctxParam1.ctxSObjectId = agmtId1;
ctxParam1.comments = submitComments1;
ctxParams.add(ctxParam1);

Apttus_Approval.BulkCtxObjectParam ctxParam2 = new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam2.ctxSObjectType = agmtType;
ctxParam2.ctxSObjectId = agmtId2;
ctxParam2.comments = submitComments2;
ctxParams.add(ctxParam2);
```

```
Apttus_Approval.BulkCtxObjectParam ctxParam3 =  new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam3.ctxSObjectType = agmtType;
ctxParam3.ctxSObjectId = agmtId3;
ctxParam3.comments = submitComments3;
ctxParams.add(ctxParam3);

// bulk submit
Id approvalrequestID =
Apttus_Approval.ApprovalsWebService.approveRequestsBulk(ctxParams);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService"
xmlns:bul = "http://soap.sforce.com/schemas/class/Apttus_Approval/
BulkCtxObjectParam"
xmlns:sub = "http://soap.sforce.com/schemas/class/Apttus_Approval/
SubmissionComments" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQOruA8X3nIS2hgZgZxEZkUDHDMlzIfBNGDey8s_.AcTbbEghNGUnMEh5oGcG5mkmrVuHp1F9
gHzfIfYAvzuRDU6zg7kO</ app :sessionId>
      </ app :SessionHeader>
   </ soapenv :Header>
   < soapenv :Body>
      < app :submitForApprovalsBulk>
```

```
         <!--Zero or more repetitions:-->
         < app :ctxObjParams>
            < bul :comments>
               < sub :commentsCount>1</ sub :commentsCount>
               < sub :commentsLevel>Process</ sub :commentsLevel>
               < sub :processComment>?</ sub :processComment>
               < sub :processCommentLabel>ProcessLabel</ sub :processComment
Label>
               < sub :processCommentMandatory>false</ sub :processCommentMan
datory>
               < sub :processName>ProcessName</ sub :processName>
               <!--Zero or more repetitions:-->
               < sub :stepCommentList>
               </ sub :stepCommentList>
            </ bul :comments>
            < bul :ctxSObjectId>a07R000000AiYQX</ bul :ctxSObjectId>
            < bul :ctxSObjectType>Apttus__APTS_Agreement__c</ bul :ctxSObjec
tType>
         </ app :ctxObjParams>
         < app :ctxObjParams>
            < bul :comments />
            < bul :ctxSObjectId>a07R000000Aj68Y</ bul :ctxSObjectId>
            < bul :ctxSObjectType>Apttus__APTS_Agreement__c</ bul :ctxSObjec
tType>
         </ app :ctxObjParams>
      </ app :submitForApprovalsBulk>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < submitForApprovalsBulkResponse >
         < result >true</ result >
      </ submitForApprovalsBulkResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Cancelling Approvals

This API cancels the entire approval process associated with the context object. Currently, assigned approvers will be notified via email. This API accepts the object type and the object ID of the approval request as input parameters.

API	Signature
cancelApprovals	*webService static Boolean cancelApprovals(String sObjectType, Id sObjectId)*

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	ID of the approval context object.
sObjectType	String	Yes	Type of the approval context object.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the approval is cancelled successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Header>
       < app :SessionHeader>
          < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
       </ app :SessionHeader>
       </ soapenv :Header>
    < soapenv :Body>
       < app :cancelApprovals>
          < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
          < app :sObjectId>a07R000000AiYlw</ app :sObjectId>
       </ app :cancelApprovals>
    </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Body>
       < cancelApprovalsResponse >
          < result >true</ result >
       </ cancelApprovalsResponse >
    </ soapenv :Body>
</ soapenv :Envelope>
```

Checking If an Approval is Required

This API determines if a given context object, in its current state, requires approval. This API accepts the ID of the context object and the type of the context object as request parameters.

isApprovalRequired

API	Signature
isApprovalRequired	*webService static Boolean isApprovalRequired(String sObjectType, Id sObjectId)*

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	ID of the business context object.
sObjectType	String	Yes	Type of the context object.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if approval is required.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/
Apttus_Approval/ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :isApprovalRequired>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R000000AiYlw</ app :sObjectId>
      </ app :isApprovalRequired>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < isApprovalRequiredResponse >
         < result >true</ result >
      </ isApprovalRequiredResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

checkIfApprovalRequired

API	Signature
checkIfApprovalRequired	*webService static* *List checkIfApprovalRequired(List sObjectIds)*

This API determines if a given list of context objects, in its current state, requires approval. It returns a list of true/false values corresponding to the order in which ids were input.

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectIds	List<ID>	Yes	List of IDs of business context objects.

Response Parameter		
Name	**Type**	**Description**
result	List<Boolean>	Returns a list of boolean indicators. The indicator is true if approval is required.

Integration Details

Use the following information in your integrations with Conga Intelligent Workflow Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
```

```
   < soapenv :Body>
      < app :checkIfApprovalRequired>
         <!--Zero or more repetitions:-->
         < app :sObjectIds>a07R000000AiYlw</ app :sObjectIds>
      </ app :checkIfApprovalRequired>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
  xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < checkIfApprovalRequiredResponse >
         < result >false</ result >
      </ checkIfApprovalRequiredResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

checkIfApprovalRequired2

API	Signature
checkIfApprovalRequired2	*webService static* *List* checkIfApprovalRequired2(String headerIdStatus, List childIdStatusList, List modifiedChildObjectIds)

This API determines if a context header object (For example: Agreement) or context child object (For example: AgreementLineItems), in its current state, requires approval. It returns a single consolidated list with updated approval status (a01U0000017FRf9IAG__Approval Required) against the corresponding Id.

> headerIdStatus & childIdStatus are constructed as follows:
> Example: (a01U0000017FRf9IAG__None) - ID and Approval_Status__c value of the object, separated by '__' (double-underscore)

Request Parameters			
Name	**Type**	**Required?**	**Description**
headerIdStatus	String	Yes	Header ID and approval status.
childIdStatusList	List<String>	Yes	List of approval status ID of child objects.
modifiedChildObjectIds	Lsit<ID>	Yes	List of IDs of modified child objects.

Response Parameter		
Name	**Type**	**Description**
result	List<String>	Returns a list of the header ID and child object id approval statuses separated by '__'.

Integration Details

Use the following information in your integrations with Apttus Intelligent Workflow Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
 < soapenv :Header>
 < app :SessionHeader>
```

```
< sessionId >00DR0000001nyVR!
ARYAQLGLUK2b6FoONhmSG.PKSeCYBRUhIi4LqHYHUF8J.W1m8zLATtD0rjJECgiJhcfEwcgRAUC6
uaHqgF9jAmkP1evILkhb</ sessionId >
</ app :SessionHeader>
</ soapenv :Header>
   < soapenv :Body>
      < app :checkIfApprovalRequired2>
         < app :headerIdStatus>a07R000000AijfS__None</ app :headerIdStatus>
      </ app :checkIfApprovalRequired2>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < checkIfApprovalRequired2Response >
         < result >a07R000000AijfSIAR__null</ result >
      </ checkIfApprovalRequired2Response >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Checking if User is Authorised to Approve or Reject

This API checks if the context user is authorized to approve or reject an approval request. This API accepts the request ID of the approval as an input parameter.

API	Signature
canApproveRejectRequest	*webService static Boolean canApproveRejectRequest(Id requestId)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
requestId	ID	Yes	ID of the approval request.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the user is authorized to approve or reject a request.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
```

```
      < app :canApproveRejectRequest>
         < app :requestId>a0oR0000006cgWz</ app :requestId>
      </ app :canApproveRejectRequest>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < canApproveRejectRequestResponse >
         < result >true</ result >
      </ canApproveRejectRequestResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Creating Ad-hoc Approvals

This API creates an ad-hoc approval step. This API accepts Context Request ID and creates an approval step for it.

API	Signature
createAdhocApproval	*webService static Boolean createAdhocApproval(Id contextRequestId, String stepLabel, String assigneeType, Id assigneeId, String submissionComment, String relativeLocationToContext, List dependsOnContextRequestIdList, List dependsOnMeContextRequestIdList, Boolean sendEmail, Boolean notifyOnly)*

createAdhocApproval

Request Parameters			
Name	**Type**	**Required?**	**Description**
contextRequestId	ID	Yes	ID of the context request in relation to which you want to create an approval step.
stepLable	String	Yes	Text label of the step.
assigneeType	String	Yes	Type of the assignee. Assignee can be: • User • Role • Queue • Related User
assigneeID	ID	Yes	ID of the assignee.
submissionComment	String	Yes	Comments for approval submission.
relativeLocationToContext	String	Yes	Relative Location can be: • Before • After • In-parallel
dependsOnContextRequestIdList	List<ID>	Yes	List of request identifiers that the ad-hoc approval depends on.
dependsOnMeContextRequestIdList	List<ID>	Yes	List of request identifiers that depend on this ad-hoc approval.
sendEmail	Boolean	No	Set it as true to send an email request. The default value is true.
notifyOnly	Boolean	No	The default value is false

Response Parameter		
Name	**Type**	**Description**
result	Boolean	The API returns a true value if the ad-hoc approval is created successfully

Enabling Auto Re-approvals on Proposal and Proposal Line Items

This API takes several parameters and updates the reapproval data in the process instance so that proposal line items can be automatically reapproved when resubmitted after finalizing new items in the cart. Note that this API is available in the base Approvals package and can be used with the optional unmanaged custom package "Apttus Proposal Auto-Reapprovals" available on request with Conga Release Management. This package contains the complete sample code that can used along with the API in the base package to make auto-reapprovals work with Proposals and Proposal Line Items.

API	Signature
updateApprovalData	*webService static Boolean updateApprovalData(Id instanceId, String sObjectType, Id contextObjId, List oldContextObjIds, List newContextObjIds)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
instanceId	ID	Yes	The process instance id that is being currently executed in the system.
sObjectType	String	Yes	The sObject type that is used to identify the object type.
contextObjId	ID	Yes	The ID of the context object.

Request Parameters			
Name	**Type**	**Required?**	**Description**
oldContextObjIds	List<ID>oldContextObjIds	Yes	The list of old approval data context object ids that are to be replaced with the new ones.
newContextObjIds	List<ID> newContextObjIds	Yes	The list of new approval data context object id store place with.

Response Parameter			
Name	**Type**	**Required**	**Description**
Ok	Boolean	Yes	Defines if the operation was successful. Returns *true* if yes, otherwise *false*.

Code Sample

The sample below enables you to execute auto re-approvals over Proposal Line Items by considering the input parameters, such as process instance, sObject type, context object, and lists of old and new context object ids. These IDs represent the Proposal Line Item IDs before and after the cart is finalized. You can use this API in a scenario where a Quote/Proposal is approved but the Approvals Manager decides to change the discount from 10 to 15% on a Proposal Line Item, you can resubmit the approval request for this for re-approval. The system honors the auto-reapproval criteria, using which a request once approved by certain assignees are auto-approved.

See the complete code included in the package to understand how this works, especially CustomProposalLineItemSupport.cls. Note that the code in this package is based on a trigger on the Proposal Line Item object which saves the old ids of the Proposal Line Items before they are deleted as well as the new ids of the Proposal Line Items that get inserted after the cart is finalized. You can extend this concept for other custom child objects to the proposal object by creating a trigger on them and using the same pattern in the sample code to save ids before and after new items are created.

```
/**
*Apttus Approvals Management
```

```
*CustomProposalLineItemTrigger
*@2017 Apttus Inc. All rights reserved.
*/


trigger CustomProposalLineItemTrigger on
Apttus_Proposal__Proposal_Line_Item__c (before delete, after insert)
{
     if  (Trigger.isBefore && Trigger.isDelete)
    {
        // save map of old line items to attributes before they are deleted
        CustomProposalLineItemSupport.doBeforeDelete(Trigger.old,
Trigger.oldMap);
    }
     if  (Trigger.isAfter && Trigger.isInsert)
    {
         // update reapproval data with new line items after they are
inserted
        CustomProposalLineItemSupport.doAfterInsert(Trigger. new ,
Trigger.newMap);
    }
}
/**
*Apttus Approvals Management
*CustomProposalLineItemSupport
* @2017 Apttus Inc. All rights reserved.
*/

public  with sharing  class  CustomProposalLineItemSupport  extends
 CustomApprovalsConstants
{
     // line item types
     private   static   final  String LINETYPE_PRODUCT =  'Product/Service' ;
     private   static   final  String LINETYPE_OPTION =  'Option' ;

     // proposal
     private   static   final  String PROPOSAL_SOBJECT_TYPE =
'Apttus_Proposal__Proposal__c' ;
     private   static  ID quoteId =  null ;

     // associated process instance
     private   static  Apttus_Approval__ApprovalProcessInstance__c
quoteProcessInstance =  null ;
```

```
    // map of old line item ids to attribute key
    private   static  Map<ID,String> oldLineId2KeyMap =  new
 Map<ID,String>();

    // map of new line item attribute key to id
    private   static  Map<String,ID> newLineKey2IdMap =  new
 Map<String,ID>();

    // map of old line item ids to new line item ids
    private   static  Map<ID,ID> lineOldId2NewIdMap =  new  Map<ID,ID>();

    /**
    * Process old ProposalLineItems before they are deleted when a cart is
finalized
    *  @param oldLineItems - a list of the old versions of the sObject
records
    *  @param oldLineItemsMap - a map of IDs to the old versions of the
sObject records
    */

    public   static   void
 doBeforeDelete( List<Apttus_Proposal__Proposal_Line_Item__c> oldLineItems,
Map<ID, Apttus_Proposal__Proposal_Line_Item__c> oldLineItemsMap)
 {
    // iterate over line items about to deleted
    for  (Integer i= 0 ; i<oldLineItems.size(); i++)
    {
        Apttus_Proposal__Proposal_Line_Item__c oldLineItem =
oldLineItems[i];
    // get quote from line item
        quoteId = oldLineItem.Apttus_Proposal__Proposal__c;
    // get line item attributes
        String lineNumber =
String.valueOf(oldLineItem.Apttus_QPConfig__PrimaryLineNumber__c);
        String lineType = oldLineItem.Apttus_QPConfig__LineType__c;
        String productId =  null ;

         if  (lineType == LINETYPE_PRODUCT)
        {
            productId = oldLineItem.Apttus_Proposal__Product__c;
        }
         else   if  (lineType == LINETYPE_OPTION)
        {
            productId = oldLineItem.Apttus_QPConfig__OptionId__c;
```

```
            //productId = oldLineItem.Apttus_QPConfig__ProductOptionId__c;
        }

        String chargeType = oldLineItem.Apttus_QPConfig__ChargeType__c;
         // create attribute key
            String oldLineKey = lineNumber +  ':'  + productId +  ':'  +
chargeType;
         // save in map
            oldLineId2KeyMap.put(oldLineItem.Id, oldLineKey);
    }
}
/**
* Process new ProposalLineItems after they are inserted when a cart is
finalized
* @param newLineItems - a list of the new versions of the sObject records
* @param newLineItemsMap - a map of IDs to the new versions of the sObject
records
*/
     public  static  void
 doAfterInsert(List<Apttus_Proposal__Proposal_Line_Item__c> newLineItems,
Map<ID, Apttus_Proposal__Proposal_Line_Item__c> newLineItemsMap)
    {
         // iterate over line items about to deleted
         for  (Integer i= 0 ; i<newLineItems.size(); i++)
        {
            Apttus_Proposal__Proposal_Line_Item__c newLineItem =
newLineItems[i];

         // get quote from line item
            quoteId = newLineItem.Apttus_Proposal__Proposal__c;

         // get line item attributes
            String lineNumber =
String.valueOf(newLineItem.Apttus_QPConfig__PrimaryLineNumber__c);
            String lineType = newLineItem.Apttus_QPConfig__LineType__c;
            String productId =  null ;

             if  (lineType == LINETYPE_PRODUCT)
            {
                productId = newLineItem.Apttus_Proposal__Product__c;
            }
             else  if (lineType == LINETYPE_OPTION)
```

```
            {
                productId = newLineItem.Apttus_QPConfig__OptionId__c;
                 //productId =
newLineItem.Apttus_QPConfig__ProductOptionId__c;
            }
            String chargeType = newLineItem.Apttus_QPConfig__ChargeType__c;

             // create attribute key
                String newLineKey = lineNumber +  ':'  + productId +  ':'  +
chargeType;
             // save in map
                newLineKey2IdMap.put(newLineKey, newLineItem.Id);

        }
         // create map of old line item ids to new line item ids
            Set<ID> oldLineItemIds = oldLineId2KeyMap.keySet();
            List<ID> newLineItemIds =  new  List<ID>();

             for  (String oldLineItemId : oldLineItemIds)
            {
                 // get attribute key
                String attrKey = oldLineId2KeyMap.get(oldLineItemId);
                 // lookup key in new line items map
                String newLineItemId =  null ;
                 if (newLineKey2IdMap.containsKey(attrKey))
                {
                    newLineItemId = newLineKey2IdMap.get(attrKey);
                    newLineItemIds.add(newLineItemId);
                     // associate old key with new one
                    lineOldId2NewIdMap.put(oldLineItemId, newLineItemId);
                }

            }

        // update reapprovals data by calling API in approvals package

         List<ID> oldContextObjIds =  new  List<ID>(oldLineItemIds);
         List<ID> newContextObjIds = newLineItemIds;

         system.debug(LoggingLevel.INFO, 'sObjectType=' +PROPOSAL_SOBJECT_TYP
E);

         system.debug(LoggingLevel.INFO, 'contextObjId=' +quoteId);
```

```
        system.debug(LoggingLevel.INFO, 'oldContextObjIds=' +oldContextObjIds);

        system.debug(LoggingLevel.INFO, 'newContextObjIds=' +newContextObjIds);

         if  (!oldContextObjIds.isEmpty() && !newContextObjIds.isEmpty() && oldContextObjIds.size() == newContextObjIds.size())
        {
             // get process instance associated with the old quote
            Apttus_Approval__ApprovalProcessInstance__c instanceSO = getProcessInstance(quoteId);
            system.debug(LoggingLevel.INFO, 'instanceSO=' +instanceSO);

             // call API to update reapproval data
            Boolean ok = Apttus_Approval.ApprovalsWebService.updateApprovalData(instanceSO.Id, PROPOSAL_SOBJECT_TYPE, quoteId, oldContextObjIds, newContextObjIds);


system.debug(LoggingLevel.INFO, 'Apttus_Approval.ApprovalsWebService.updateApprovalData=' +ok);

        }
    }

/**
* Get process instance for the given proposal id
* @param proposalId
* @return process instance object
*/

     private   static  Apttus_Approval__ApprovalProcessInstance__c getProcessInstance(ID proposalId)
    {
        List<Apttus_Approval__ApprovalProcessInstance__c> instanceList = [select Id, Name, LastModifiedDate, LastModifiedById, LastActivityDate, CreatedDate, CreatedById, Apttus_Approval__Status__c, Apttus_Approval__StartTime__c, Apttus_Approval__ReassignmentEmailTemplate__c, Apttus_Approval__PrevProcessInstanceId__c, Apttus_Approval__NotifyOnlyEmailTemplate__c, Apttus_Approval__InstanceNumber__c,
```

```
Apttus_Approval__EscalationEmailTemplate__c, Apttus_Approval__EndTime__c,
Apttus_Approval__Data__c, Apttus_Approval__ConsolidationVersionNumber__c,
Apttus_Approval__CancellationEmailTemplate__c,
Apttus_Approval__BusinessObjectType__c,
Apttus_Approval__BusinessObjectLink__c,
Apttus_Approval__BusinessObjectId__c,
Apttus_Approval__AssignmentEmailTemplate__c,
Apttus_Approval__ApprovalProcessId__c From
Apttus_Approval__ApprovalProcessInstance__c where
Apttus_Approval__BusinessObjectId__c = :proposalId order by CreatedDate DESC
limit  1 ];
         if  ( ! nullOrEmpty(instanceList))
        {
            return  instanceList[ 0 ];
        }

         return   null ;
    }
/**
* 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());
    }
}
```

Escalating Approval Requests

System follows the escalation scheme specified in the current approval process. Approval Request is reassigned accordingly. This API accepts the ID of the approval request as an input parameter.

API	Signature
escalateRequest	*webService static Boolean escalateRequest(Id requestId)*

Request Parameters

Name	Type	Required?	Description
requestId	ID	Yes	ID of the approval request.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the approval request is escalated.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :escalateRequest>
         < app :requestId>a0oR0000006cqc6</ app :requestId>
      </ app :escalateRequest>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < escalateRequestResponse >
         < result >true</ result >
      </ escalateRequestResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Previewing Approvals

This API generates required approval requests for the given context object. You must query the approval request table to fetch them. This API accepts the object type and the object ID of the approval request as input parameters.

API	Signature
previewApprovals	*webService static Boolean previewApprovals(String sObjectType, Id sObjectId)*

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	ID of the approval context object.
sObjectType	String	Yes	Type of the approval context object.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the API is executed successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Header>
       < app :SessionHeader>
          < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
       </ app :SessionHeader>
       </ soapenv :Header>
    < soapenv :Body>
       < app :previewApprovals>
          < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
          < app :sObjectId>a07R000000AiYlw</ app :sObjectId>
       </ app :previewApprovals>
    </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Body>
       < previewApprovalsResponse >
          < result >true</ result >
       </ previewApprovalsResponse >
    </ soapenv :Body>
</ soapenv :Envelope>
```

Reassigning Approval Requests

This API reassigns the approval request. This API accepts the ID of the approval request, Id of the assignee and comments as input parameters.

API	Signature
reassignRequest	*webService static Boolean reassignRequest(Id requestId, Id toAssigneeId, String comments)*

Request Parameters

Name	Type	Required?	Description
requestId	ID	Yes	ID of the approval request.
toAssigneeId	ID	Yes	ID of the new assignee.
comments	String	Yes	Comments

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the approval request is reassigned.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :reassignRequest>
         < app :requestId>a0oR0000006cr6t</ app :requestId>
         < app :toAssigneeId>005q00000028qNb</ app :toAssigneeId>
         < app :comments>ReAssigned</ app :comments>
      </ app :reassignRequest>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
  xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < reassignRequestResponse >
         < result >true</ result >
      </ reassignRequestResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Rejecting Approval Requests

This API rejects the approval request. This API accepts the ID of the approval request and comments as input parameters.

API	Signature
rejectRequest	*webService static Boolean rejectRequest(Id requestId, String comments)*

Request Parameters

Name	Type	Required?	Description
requestId	ID	Yes	ID of the approval request.
comments	String	Yes	Comments to add to the approval request.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the approval request is rejected successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/
Apttus_Approval/ApprovalsWebService"   xmlns:bul = "http://soap.sforce.com/
schemas/class/Apttus_Approval/BulkActionRequest" >
   < soapenv :Header>
      < app :SessionHeader>
         < sessionId >00DR0000001nyVR!
ARYAQOruA8X3nIS2hgZgZxEZkUDHDMlzIfBNGDey8s_.AcTbbEghNGUnMEh5oGcG5mkmrVuHp1F9
gHzfIfYAvzuRDU6zg7kO</ sessionId >
      </ app :SessionHeader>
   </ soapenv :Header>
   < soapenv :Body>
      < app :rejectRequestsBulk>
         <!--Zero or more repetitions:-->
         < app :actionRequests>
            < bul :comments>i dont approve one</ bul :comments>
            < bul :requestId>a0oR0000006d4Gf</ bul :requestId>
         </ app :actionRequests>
      </ app :rejectRequestsBulk>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService"   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-
instance" >
   < soapenv :Body>
      < rejectRequestsBulkResponse >
         < result   xsi:nil = "true" />
      </ rejectRequestsBulkResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Retrieving Add Comment Page URL

This API gets the URL of the add comments page for the given approval request. This API accepts the approval request object as input parameter.

API	Signature
getAddCommentPageUrl	*webService static String getAddCommentPageUrl(Apttus_Approval__Approval_Request__c requestSO)*

Request Parameters

Name	Type	Required?	Description
Approval_Request__c requestSO	Object	Yes	The approval request object.

Response Parameter

Name	Type	Description
addCommentPageURL	String	The URL of the add comment page.

Retrieving Approval History

This API gets the approval history associated with the approval request. This API accepts the approval request object as an input parameter and returns a list of the approval history objects.

API	Signature
getApprovalHistory	*webService static List* getApprovalHistory(Apttus_Approval__Approval_Request__c requestSO)

Request Parameters			
Name	**Type**	**Required?**	**Description**
Approval_Request__c requestSO	Object	Yes	The approval request object.

Response Parameter		
Name	**Type**	**Description**
approvalHistorySO	List<Approval_Request__c>	List of approval history objects.

Retrieving Approval Page URL

This API gets the approvals page URL for the given approval request. This API accepts the approval request object as input parameter.

API	Signature
getMyApprovalsPageUrl	*webService static String getMyApprovalsPageUrl(Apttus_Approval__Approval_Request__c requestSO)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
Approval_Request__c requestSO	Object	Yes	The approval request object.

Response Parameter		
Name	**Type**	**Description**
myApprovalPageURL	String	The URL of the approval page.

Retrieving Approve or Reject Page URL

This API gets the approve/reject page URL for the given approval request. This API accepts the approval request object as input parameter.

API	Signature
getApproveRejectPageUrl	*webService static String getApproveRejectPageUrl(Apttus_Approval__Approval_Request__c requestSO)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
Approval_Request__c requestSO	Object	Yes	The approval request object.

Response Parameter		
Name	**Type**	**Description**
approveRejectPageURL	String	The URL of the approve/reject page.

Retrieving Reassign Page URL

This API gets the reassign page URL for the given approval request. This API accepts the approval request object as input parameter.

API	Signature
getReassignPageUrl	*webService static String getReassignPageUrl(Apttus_Approval__Approval_Request__c requestSO)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
Approval_Request__c requestSO	Object	Yes	The approval request object.

Response Parameter		
Name	**Type**	**Description**
reassignPageURL	String	The URL of the reassign page.

Submitting for Approvals

With this API, you can submit for approvals. This API accepts the object type and the object ID of the context object as input parameters.

API	Signature
submitForApprovals	*webService static Boolean submitForApprovals(String sObjectType, Id sObjectId)*

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	ID of the approval context object.
sObjectType	String	Yes	Type of the approval context object.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the API is executed successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Header>
       < app :SessionHeader>
          < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
       </ app :SessionHeader>
       </ soapenv :Header>
    < soapenv :Body>
       < app :submitForApprovals>
          < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
          < app :sObjectId>a07R000000AiYlwIAF</ app :sObjectId>
       </ app :submitForApprovals>
    </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Body>
       < submitForApprovalsResponse >
          < result >true</ result >
       </ submitForApprovalsResponse >
    </ soapenv :Body>
</ soapenv :Envelope>
```

Submitting for Approvals with Attachments

With this API, you can submit an approval request with attachments. This API accepts the object type and the object ID of the context object as input parameters.

API	Signature
submitForApprovalsWithAttachments	*static Boolean submitForApprovalsWithAttachments(String sObjectType, Id sObjectId, Id processId, Map templateFld2NameIdMap)*

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	ID of the approval context object.
sObjectType	String	Yes	Type of the approval context object.
processId	ID	Yes	ID of the approval process
templateFld2NameIdMap	Map	Yes	Map of process instance template field names and template names

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the API is executed successfully.

Using submitForApprovalsWithAttachments API to send Files

You can also use this API to send Files instead of Attachments. Sending Files is a four-step process as follows:

1. Select the process Id and file Id (or attachment Id) to submit.

2. Clone the email templates using the CloneEmailTemplate API. You need to clone the email templates to create temporary templates before you can copy attachments to them.
3. Copy the files or attachments using the CopyAttachment API.
4. Get the developer names of the email templates you want to use. Create a map with the template field names and their respective template developer names and pass this to the submitForApprovalsWithAttachments API in the processProperties parameter.

Code Sample

```
//
================================================================================
=======
//  Step 1 - select process
//
================================================================================
=======

// agreement id
ID agmtId =  'a013l00000vbQaPAAU' ;
// content document id
ID contentDocId =  '0693l00000HookC' ;
// get content version
List<ContentVersion> contentVersions = [SELECT Id, VersionNumber, Title,
PathOnClient, ContentDocumentId, ContentDocument.Title
FROM ContentVersion
                    WHERE ContentDocumentId = :contentDocId];
system.debug( 'contentVersions[0]=' +contentVersions[ 0 ]);
ID contentVersionId = contentVersions[ 0 ].Id;
system.debug( 'contentVersionId=' +contentVersionId);

// select process
ID processId =
Apttus_Approval.ApprovalsWebService.selectApprovalProcess(
'Apttus__APTS_Agreement__c' , agmtId);
system.debug( 'processId=' +processId);

//
================================================================================
=======
//  Step 2 - clone email templates
```

```
//
================================================================================
=======

// get email templates for process
// Apttus My Approvals Notification (Assignment)
ID asTemplateId =  '00X1N000000jwAr' ;
// Apttus My Approvals Notification (Reassignment)
ID reTemplateId =  '00X1N000000jwAv' ;
// Apttus My Approvals Notification (Escalation)
ID esTemplateId =  '00X1N000000jwAt' ;
// Apttus My Approvals Notification (NotifyOnly)
ID noTemplateId =  '00X1N000000jwAu' ;

// clone templates
ID asClonedTemplateId =
Apttus_Approval.ApprovalsWebService.cloneEmailTemplate(asTemplateId);
ID reClonedTemplateId =
Apttus_Approval.ApprovalsWebService.cloneEmailTemplate(reTemplateId);
ID esClonedTemplateId =
Apttus_Approval.ApprovalsWebService.cloneEmailTemplate(esTemplateId);
ID noClonedTemplateId =
Apttus_Approval.ApprovalsWebService.cloneEmailTemplate(noTemplateId);
system.debug( 'asClonedTemplateId=' +asClonedTemplateId);
system.debug( 'reClonedTemplateId=' +reClonedTemplateId);
system.debug( 'esClonedTemplateId=' +esClonedTemplateId);
system.debug( 'noClonedTemplateId=' +noClonedTemplateId);

//
================================================================================
=======
//  Step 3 - copy attachments to cloned email templates
//
================================================================================
=======

// have to do this in separate execution thread otherwise get error:

ID processId =  'a0v3l00000Lp9S1AAJ' ;
ID contentVersionId =  '0683l00000IlB4xAAF' ;
ID asClonedTemplateId =  '00X3l000001qRAHEA2' ;
ID reClonedTemplateId =  '00X3l000001qRAIEA2' ;
ID esClonedTemplateId =  '00X3l000001qRAJEA2' ;
```

```
ID noClonedTemplateId =  '00X3l000001qRAKEA2' ;

// copy attachments to cloned templates
 Boolean copyAs =
Apttus_Approval.ApprovalsWebService.copyAttachment(asClonedTemplateId,
contentVersionId);
 Boolean copyRe =
Apttus_Approval.ApprovalsWebService.copyAttachment(reClonedTemplateId,
contentVersionId);
 Boolean copyEs =
Apttus_Approval.ApprovalsWebService.copyAttachment(esClonedTemplateId,
contentVersionId);
 Boolean copyNo =
Apttus_Approval.ApprovalsWebService.copyAttachment(noClonedTemplateId,
contentVersionId);

//
================================================================================
=======
//  Step 4 - submit with attachments
//
================================================================================
=======

ID agmtId =  'a013l00000vbQaPAAU' ;
ID processId =  'a0v3l00000Lp9S1AAJ' ;
ID asClonedTemplateId =  '00X3l000001qRAHEA2' ;
ID reClonedTemplateId =  '00X3l000001qRAIEA2' ;
ID esClonedTemplateId =  '00X3l000001qRAJEA2' ;
ID noClonedTemplateId =  '00X3l000001qRAKEA2' ;

// get cloned template names
List<ID> clonedTemplateIds =  new  List<ID>();
clonedTemplateIds.add(asClonedTemplateId);
clonedTemplateIds.add(reClonedTemplateId);
clonedTemplateIds.add(esClonedTemplateId);
clonedTemplateIds.add(noClonedTemplateId);

List<EmailTemplate> templates = [SELECT Id, DeveloperName
                                 FROM EmailTemplate
                                 WHERE Id IN :clonedTemplateIds];
system.debug( 'templates=' +templates);

String asClonedTemplateName =  'TEMPpEUFEwzmOwGmG7gMTwMn4MQPd' ;
```

```
String reClonedTemplateName =  'TEMPkBT6h3HbJSvkNn2jN2aYGDv7A' ;
String esClonedTemplateName =  'TEMPlDvMQ2epmJpk33IhrJUxDkCpG' ;
String noClonedTemplateName =  'TEMPlMgrRErwNI7ZnJrajTZrdP4Du' ;

// create process properties map containing cloned template names
Map<String,String> templateFldName2NameMap =  new  Map<String,String>();
templateFldName2NameMap.put( 'Apttus_Approval__AssignmentEmailTemplate__c' ,
asClonedTemplateName);
templateFldName2NameMap.put( 'Apttus_Approval__ReassignmentEmailTemplate__c'
, reClonedTemplateName);
templateFldName2NameMap.put( 'Apttus_Approval__EscalationEmailTemplate__c' ,
esClonedTemplateName);
templateFldName2NameMap.put( 'Apttus_Approval__NotifyOnlyEmailTemplate__c' ,
noClonedTemplateName);
system.debug( 'templateFldName2NameMap=' +templateFldName2NameMap);

// submit with attachments
Apttus_Approval.ApprovalsWebService.submitForApprovalsWithAttachments(
'Apttus__APTS_Agreement__c' ,
                                                        agmtId,
                                                        processId,

templateFldName2NameMap);
```

Submitting For Approvals With Comments

With this API, you can submit an approval context with comments.

API	Signature
submitForApprovalsWithComments	*webService static Boolean submitForApprovalsWithComments(String sObjectType, Id sObjectId, Apttus_Approval.SubmissionComments comments)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectId	ID	Yes	ID of the approval context object.

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectType	String	Yes	Type of the approval context object.
comments	Object	Yes	The SubmissionComments object.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the API is executed successfully.

Code Sample

```
// create process level submission comments
SubmissionComments comments = new SubmissionComments();
comments.commentsLevel = SubmissionComments.PROCESS_LEVEL_COMMENTS;
comments.commentsCount = 1 ;
comments.processName = 'MyProcessName' ;
comments.processCommentLabel = 'MyProcessLabel' ;
comments.processCommentMandatory = false ;
comments.processComment = 'Here is my submission comment' ;

// call submit with comments API
Apttus_Approval.ApprovalsWebService.submitForApprovalsWithAttachmentsAndComm
ents( 'Apttus_APTS_Agreement_c' , agmtId, comments);
```

SubmissionComments Class

```
/**
 *  Apttus Approvals Management
 *  SubmissionComments
 *
 *  @2010-2019 Apttus Inc. All rights reserved.
 */
```

```
global with sharing class SubmissionComments {

    public static final String PROCESS_LEVEL_COMMENTS = 'Process';
    public static final String STEP_LEVEL_COMMENTS = 'Step';

    // comments level
    public String commentsLevel = null;
    // comments count
    public Integer commentsCount = 1;

    // process name
    public String processName = null;
    // process comments label
    public String processCommentLabel = null;
    // process comment mandatory
    public Boolean processCommentMandatory = false;
    // process comments
    public String processComment = null;

    // step comments list
    public List<StepComment> stepCommentList = new List<StepComment>();

    /**
    * Public constructor
    */
    public SubmissionComments() {

    }

    /**
    * Gets process name
    */
    global String getProcessName() {
        return processName;
    }

    /**
    * Gets process comment label
    */
    global String getProcessCommentLabel() {
        return processCommentLabel;
    }
```

```
    /**
     * Gets process comment
     */
    global String getProcessComment() {
        return  processComment;
    }

    /**
     * Sets process comment
     * @param comment
     */
    global  void  setProcessComment(String comment) {
        this .processComment = comment;
    }

    /**
     * Is comment at process level?
     */
    global Boolean isProcessLevelComment() {
        return  (PROCESS_LEVEL_COMMENTS == commentsLevel);
    }

    /**
     * Sets process comment to manadatory
     */
    global  void  setProcessCommentMandatory() {
        this .processCommentMandatory =  true ;
    }

    /**
     * Is comment at process level mandatory?
     */
    global Boolean isProcessLevelCommentMandatory() {
        return  processCommentMandatory;
    }

    /**
     * Is comment at step level?
     */
    global Boolean isStepLevelComment() {
        return  (STEP_LEVEL_COMMENTS == commentsLevel);
    }

    /**
```

```
     * Get comments count - max of 3 at step-level and 1 at process-level
     */
    global Integer getCommentsCount() {
         return  commentsCount;
    }

     /**
     * Gets step comment list
     */
    global List<StepComment> getStepCommentList() {
         return  stepCommentList;
    }

     /**
     * Add stepComment to list
     * @param comment
     */
    global  void  addStepComment(StepComment comment) {
        stepCommentList.add(comment);
    }

     /**
     * Creates the json representation of the submission comments
     * @return the json string
     */
    global String toJSON() {
         // get the json representation
         return  System.JSON.serializePretty( this );

    }

     /**
     * Parses the submission comments object from the given JSON string
     * @param jsonString the json string to parse
     * @return the submission comments data object
     */
    global  static  SubmissionComments parse(String jsonString) {
         // load the class (salesforce class loading bug)
        System.Type wrapperType =
System.Type.forName(SystemUtil.getFQClassName( 'SubmissionComments' ));
         // deserialize the object
         return  (SubmissionComments) System.Json.deserialize(jsonString,
SubmissionComments. class );
```

```
    }

    /**
     * Inner class to hold step level comments
     */
    global class StepComment {

        // step name
        public String stepName = null;
        // step comment label
        public String stepCommentLabel = null;
        // step comments
        public String stepComment = null;

        /**
        * Public constructor
        */
        public StepComment() {

        }

        /**
        * Gets step name
        */
        global String getStepName() {
            return stepName;
        }

        /**
        * Gets step comment label
        */
        global String getStepCommentLabel() {
            return stepCommentLabel;
        }

        /**
        * Gets step comment
        */
        global String getStepComment() {
            return stepComment;
        }

        /**
```

```
         * Set step comment
         * @param comment
         */
        global void setStepComment(String comment) {
            this.stepComment = comment;
        }
    }

}
```

Integration Details

Use the following information in your integrations with Conga Approvals API. Refer to Integrating Apttus 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:app="http://soap.sforce.com/schemas/class/Apttus_Approval/ApprovalsWebService" xmlns:sub="http://soap.sforce.com/schemas/class/Apttus_Approval/SubmissionComments">
   <soapenv:Header>
      <app:SessionHeader>
         <app:sessionId>00DR0000001nyVR!ARYAQOruA8X3nIS2hgZgZxEZkUDHDMlzIfBNGDey8s_.AcTbbEghNGUnMEh5oGcG5mkmrVuHp1F9gHzfIfYAvzuRDU6zg7kO</app:sessionId>
      </app:SessionHeader>
   </soapenv:Header>
   <soapenv:Body>
      <app:submitForApprovalsWithComments>
         <app:sObjectType>Apttus__APTS_Agreement__c</app:sObjectType>
         <app:sObjectId>a07R000000AiYQXIA3</app:sObjectId>
         <app:comments>
            <sub:commentsCount>1</sub:commentsCount>
```

```
            < sub :commentsLevel>Process</ sub :commentsLevel>
            < sub :processComment>Here is my comment</ sub :processComment>
            < sub :processCommentLabel>ProcessLabel</ sub :processCommentLab
el>
            < sub :processCommentMandatory>false</ sub :processCommentMandat
ory>
            < sub :processName>ProcessName</ sub :processName>
         </ app :comments>
      </ app :submitForApprovalsWithComments>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < submitForApprovalsWithCommentsResponse >
         < result >true</ result >
      </ submitForApprovalsWithCommentsResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Submitting For Approvals With Comments JSON

With this API, you can submit an approval context with comments JSON.

API	Signature
submitForApprovalsWithComments	*webService static Boolean submitForApprovalsWithCommentsJSON(String sObjectType, Id sObjectId, String commentsJSON)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectId	ID	Yes	ID of the approval context object.

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectType	String	Yes	Type of the approval context object.
commentsJSON	String	Yes	A JSON representation of the SubmissionComments object.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the API is executed successfully.

Code Sample

The sample below enables you to set process and step level comments.

```
// create process level submission comments JSON structure
SubmissionComments commentsInfo =  new  SubmissionComments();
commentsInfo.commentsLevel = SubmissionComments.PROCESS_LEVEL_COMMENTS;
commentsInfo.commentsCount =  1 ;
commentsInfo.processName =  'ProcessName' ;
commentsInfo.processCommentLabel =  'ProcessLabel' ;
commentsInfo.processCommentMandatory =  false ;
commentsInfo.processComment =  'Here is my comment' ;

String submitCommentsJSON = commentsInfo.toJSON();
/* submitCommentsJSON =
{
  "stepCommentList" : [],
    "processName" : "ProcessName",
    "processCommentMandatory" : false,
    "processCommentLabel" : "ProcessLabel",
    "processComment" : "Here is my comment",
    "commentsLevel" : "Process",
    "commentsCount" : 1
}
```

```
*/
//  get JSON representation for submission comments
String submitCommentsJSON = commentsInfo.toJSON();
// submit with comments
ApprovalsWebService.submitForApprovalsWithCommentsJSON(SObjectConstants.SOBJ
ECT_TYPE_AGREEMENT,  'a013l00000sR4iPAAS' , submitCommentsJSON);



// create step level submission comments JSON structure
SubmissionComments commentsInfo =  new  SubmissionComments();
commentsInfo.commentsLevel = SubmissionComments.STEP_LEVEL_COMMENTS;
commentsInfo.commentsCount =  3 ;
// create step comment
SubmissionComments.StepComment stepComment1 =  new
 SubmissionComments.StepComment();
stepComment1.stepName =  'User Assignee' ;
stepComment1.stepCommentLabel =  'GS_Business_Justification' ;
stepComment1.stepComment =  'This is comment 1' ;
commentsInfo.addStepComment(stepComment1);
// create step comment
SubmissionComments.StepComment stepComment2 =  new
 SubmissionComments.StepComment();
stepComment2.stepName =  'Queue Assignee' ;
stepComment2.stepCommentLabel =  'GS_Deal_Strategy' ;
stepComment2.stepComment =  'This is comment 2' ;
commentsInfo.addStepComment(stepComment2);
// create step comment
SubmissionComments.StepComment stepComment3 =  new
 SubmissionComments.StepComment();
stepComment3.stepName =  'Role Assignee' ;
stepComment3.stepCommentLabel =  'GS_Supporting_Information' ;
stepComment3.stepComment =  'This is comment 3' ;
commentsInfo.addStepComment(stepComment3);

/* submitCommentsJSON =
{
  "stepCommentList" : [ {
    "stepName" : "User Assignee",
    "stepCommentLabel" : "GS_Business_Justification",
    "stepComment" : "This is comment 1"
  }, {
    "stepName" : "Queue Assignee",
    "stepCommentLabel" : "GS_Deal_Strategy",
```

```
    "stepComment" : "This is comment 2"
  }, {
    "stepName" : "Role Assignee",
    "stepCommentLabel" : "GS_Supporting_Information",
    "stepComment" : "This is comment 3"
  } ],
  "processName" : null,
  "processCommentMandatory" : false,
  "processCommentLabel" : null,
  "processComment" : null,
  "commentsLevel" : "Step",
  "commentsCount" : 3
}
*/
//  get JSON representation for submission comments
String submitCommentsJSON = commentsInfo.toJSON();
// submit with comments
ApprovalsWebService.submitForApprovalsWithCommentsJSON(SObjectConstants.SOBJ
ECT_TYPE_AGREEMENT,  'a013l00000sR4iPAAS' , submitCommentsJSON);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
```

```
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :submitForApprovalsWithCommentsJSON>
         < app :sObjectType>Apttus__APTS_Agreement__C</ app :sObjectType>
         < app :sObjectId>a07R000000AiYlwIAF</ app :sObjectId>
         < app :commentsJSON>{
  "stepCommentList" : [],
    "processName" : "ProcessName",
    "processCommentMandatory" : false,
    "processCommentLabel" : "ProcessLabel",
    "processComment" : "Here is my comment",
    "commentsLevel" : "Process",
    "commentsCount" : 1
}</ app :commentsJSON>
      </ app :submitForApprovalsWithCommentsJSON>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < submitForApprovalsWithCommentsJSONResponse >
         < result >true</ result >
      </ submitForApprovalsWithCommentsJSONResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Taking Ownership of an Approval Request

This API takes ownership of the given approval request. This API accepts process definition ID, list of request IDs, user ID and ID of the context object as request parameters.

Request Parameters			
Name	**Type**	**Required?**	**Description**
processDefnId	ID	Yes	ID of the process definition.
requestIdList	Set<ID>	Yes	Set of request identifiers.
userId	ID	Yes	ID of the requesting owner.
ctxObjectId	ID	Yes	ID of the business context object.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	The API returns a true value upon successful execution of the API.

Previewing Adhoc Approvals

This API enables you to preview approvals for a context object using the adhoc approval specification for that object. This API accepts the context object type and id as input parameters and returns true if the preview is successful.

API	Signature
previewAdhocApprovals	*webService static Boolean previewAdhocApprovals(String sObjectType, Id sObjectId)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectId	ID	Yes	ID of the approval context object.

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectType	String	Yes	Type of the approval context object.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the preview is successful.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :previewAdhocApprovals>
```

```
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a0oq00000059yUI</ app :sObjectId>
      </ app :previewAdhocApprovals>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
  xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < previewAdhocApprovalsResponse >
         < result >true</ result >
      </ previewAdhocApprovalsResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Submit For Approvals Using An Adhoc Approval Specification

This API enables you to submit approvals for a context object using the adhoc approval specification for that object. This API accepts the context object type and id as input parameters and returns true if the submit is successful.

API	Signature
submitForAdhocApprovals	*webService static Boolean submitForAdhocApprovals(String sObjectType, Id sObjectId)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectId	ID	Yes	ID of the business object.

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectType	String	Yes	Type of business object.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the approval request is submitted successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :submitForAdhocApprovals>
```

```
         < app :sObjectType>Apttus__APTS_Agreement__C</ app :sObjectType>
         < app :sObjectId>a07R000000AiYlwIAF</ app :sObjectId>
      </ app :submitForAdhocApprovals>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < submitForAdhocApprovalsResponse >
         < result >true</ result >
      </ submitForAdhocApprovalsResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Submit For Approvals With Comments Using An Adhoc Approval Specification

This API enables you to submit approvals for a context object with comments using the adhoc approval specification for that object. This API accepts the context object type and id, along with the submission comments structure as input parameters and returns true if the submit is successful.

⚠ The SubmissionComments structure is same as from previous API docs.

API	Signature
submitForAdhocApprovalsWithComments	*webService static Boolean submitForAdhocApprovalsWithComments(String sObjectType, Id sObjectId, Apttus_Approval.SubmissionComments comments)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectId	ID	Yes	ID of the approval context object.
sObjectType	String	Yes	Type the approval context object.
comments	SubmissionComments	Yes	The comment to add.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the approval request is submitted successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/ApprovalsWebService"  xmlns:sub = "http://soap.sforce.com/schemas/class/Apttus_Approval/SubmissionComments" >
   < soapenv :Header>
      < app :SessionHeader>
```

```
         < app :sessionId>00DR0000001nyVR!
ARYAQOruA8X3nIS2hgZgZxEZkUDHDMlzIfBNGDey8s_.AcTbbEghNGUnMEh5oGcG5mkmrVuHp1F9
gHzfIfYAvzuRDU6zg7kO</ app :sessionId>
      </ app :SessionHeader>
   </ soapenv :Header>
   < soapenv :Body>
      < app :submitForApprovalsWithComments>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R000000AiYQXIA3</ app :sObjectId>
         < app :comments>
            < sub :commentsCount>1</ sub :commentsCount>
            < sub :commentsLevel>Process</ sub :commentsLevel>
            < sub :processComment>Here is my comment</ sub :processComment>
            < sub :processCommentLabel>ProcessLabel</ sub :processCommentLab
el>
            < sub :processCommentMandatory>false</ sub :processCommentMandat
ory>
            < sub :processName>ProcessName</ sub :processName>
         </ app :comments>
      </ app :submitForApprovalsWithComments>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < submitForApprovalsWithCommentsResponse >
         < result >true</ result >
      </ submitForApprovalsWithCommentsResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Delete An Adhoc Approval Step

This API enables you to delete an approval step which was added through the Add Adhoc Step process from the MyApprovals tab. This API accepts the id of the request to be removed as input parameter and returns true if the step is successfully deleted.

API	Signature
deleteAdhocApproval	*webService static Boolean deleteAdhocApproval(Id ctxRequestId)*

Request Parameters

Name	Type	Required?	Description
ctxRequestId	ID	Yes	ID of the request to be removed/deleted.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the approval step is deleted successfully.

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
AppsWebService" >
    < soapenv :Header>
       < app :SessionHeader>
```

```
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :deleteAdhocApproval>
         < app :ctxRequestId>a0oq00000059yUa</ app :ctxRequestId>
      </ app :deleteAdhocApproval>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < deleteAdhocApprovalResponse >
         < result >true</ result >
      </ deleteAdhocApprovalResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Adhoc Approval Process Runtime APIs

This topic contains information about the following APIs:

- Create a Runtime Adhoc Approval Process
- Update a Runtime Adhoc Approval Process
- Retrieve a Runtime Adhoc Approval Process
- Delete a Runtime Adhoc Approval Process
- Add Comment to a Runtime Adhoc Approval Process
- Add Attachments to a Runtime Adhoc Approval Process
- Delete Attachments to a Runtime Adhoc Approval Process
- Use Case and Sample Code for Adhoc Approval Runtime Process APIs

Create a Runtime Adhoc Approval Process

This API enables you to create a runtime adhoc approval process that can be used to submit for adhoc approvals. This API accepts an approval process structure as input parameter and returns the same structure with record IDs populated. For details on the input structure, refer to AdhocApprovalProcessDTO.

API	Signature
createRuntimeAdhocApprovalProcess	*webService static Apttus_Approval.AdhocApprovalProcessDTO CreateRuntimeAdhocApprovalProcess(Apttus_Approval.AdhocApprovalProcessDTO adhocApprovalProcessDTO)*

Request Parameters

Name	Type	Required?	Description
adhocApprovalProcessDTO	adhocApprovalProcessDTO	Yes	The adhoc approver process structure to create.

Response Parameter

Name	Type	Description
result	Boolean	The runtime adhoc approval process structure for the created process.

AdhocApprovalProcessDTO

```
1  /**
2   *  Apttus Approvals Management
3   *  AdhocApprovalProcessDTO
4   *
5   *  @2019 Apttus Inc. All rights reserved.
6   */
7  global with sharing class AdhocApprovalProcessDTO {
```

```
8
9       // adhoc approval process
10      private  AdhocApprovalProcess__c adhocApprovalProcessSO =
    null ;
11
12      // adhoc approval groups
13      private  List<AdhocApprovalGroupDTO> adhocApprovalGroups =
    new  List<AdhocApprovalGroupDTO>();
14
15
16      /**
17      * Public constructor
18      */
19      global AdhocApprovalProcessDTO() {
20
21      }
22
23      /**
24      * Class Constructor specifying initial values
25      * @param adhocApprovalGroupSO the adhoc approval group
26      */
27      global AdhocApprovalProcessDTO(AdhocApprovalProcess__c
    adhocApprovalProcessSO) {
28          this ();
29          setAdhocApprovalProcessSO(adhocApprovalProcessSO);
30
31      }
32
33      /**
34      * Gets adhoc approval process
35      * @return adhoc approval process sobject
36      */
37      global AdhocApprovalProcess__c getAdhocApprovalProcessSO() {
38          return  adhocApprovalProcessSO;
39
40      }
41
42      /**
43      * Sets adhoc approval process
44      * @param adhocApprovalProcessSO the adhoc approval process
45      */
```

```
46      global void setAdhocApprovalProcessSO(AdhocApprovalProcess__c adhocApprovalProcessSO) {
47          this.adhocApprovalProcessSO = adhocApprovalProcessSO;
48
49      }
50
51      /**
52       * Determines if a process has one or more approval groups
53       * @return <code>true</code> approval groups exist, <code>false</code> otherwise
54       */
55      global Boolean hasAdhocApprovalGroups() {
56          return adhocApprovalGroups != null && adhocApprovalGroups.size() > 0;
57
58      }
59
60      /**
61       * Gets list of adhoc approval groups
62       * @return list of adhoc approval groups
63       */
64      global List<AdhocApprovalGroupDTO> getAdhocApprovalGroups() {
65          return adhocApprovalGroups;
66
67      }
68
69      /**
70       * Add an adhoc approval group
71       * @param adhocApprovalGroup the adhoc approval group
72       */
73      global void addAdhocApprovalGroup(AdhocApprovalGroupDTO adhocApprovalGroup) {
74          this.adhocApprovalGroups.add(adhocApprovalGroup);
75
76      }
77
78      /**
79       * Add a list of adhoc approval group
80       * @param adhocApprovalGroups list of adhoc approval group
81       */
82      global void addAdhocApprovalGroups(List<AdhocApprovalGroupDTO> adhocApprovalGroups) {
83          this.adhocApprovalGroups.addAll(adhocApprovalGroups);
```

```
84
85      }
86
87      /**
88      * Creates the json representation of this object
89      * @return the json string
90      */
91      public  String toJSON() {
92          // get the json representation
93          return  System.JSON.serializePretty( this );
94
95      }
96
97
98      /**
99      * Inner class to hold adhoc approval groups
100     */
101     global  class  AdhocApprovalGroupDTO {
102
103         // adhoc approval group
104         public  AdhocApprovalGroup__c adhocApprovalGroupSO =
    null ;
105
106         // adhoc approvers
107         private  List<AdhocApproverDTO> adhocApprovers =  new
     List<AdhocApproverDTO>();
108
109         /**
110         * Class constructor
111         */
112         private  AdhocApprovalGroupDTO() {
113
114        }
115
116         /**
117         * Class Constructor specifying initial values
118         * @param adhocApprovalGroupSO the adhoc approval group
119         */
120        global AdhocApprovalGroupDTO(AdhocApprovalGroup__c
    adhocApprovalGroupSO) {
121             this ();
122            setAdhocApprovalGroupSO(adhocApprovalGroupSO);
123
```

```
124         }
125 
126         /**
127          * Gets adhoc approval group
128          * @return adhoc approval group sobject
129          */
130         global AdhocApprovalGroup__c getAdhocApprovalGroupSO() {
131             return  adhocApprovalGroupSO;
132 
133         }
134 
135         /**
136          * Set adhoc approval group
137          * @param adhocApprovalGroup the adhoc approval group
138          */
139         global  void
 setAdhocApprovalGroupSO(AdhocApprovalGroup__c
adhocApprovalGroupSO) {
140             this .adhocApprovalGroupSO = adhocApprovalGroupSO;
141 
142         }
143 
144         /**
145          * Determines if an approval group has one or more
approvers
146          * @return <code>true</code> approval groups exist,
<code>false</code> otherwise
147          */
148         global Boolean hasAdhocApprovers() {
149             return  adhocApprovers !=  null  &&
adhocApprovers.size() >  0 ;
150 
151         }
152 
153         /**
154          * Gets list of adhoc approvers
155          * @return list of adhoc approvers
156          */
157         global List<AdhocApproverDTO> getAdhocApprovers() {
158             return  adhocApprovers;
159 
160         }
161 
162         /**
163          * Add an adhoc approver
```

```
164             * @param adhocApprover the adhoc approver
165             */
166         global void addAdhocApprover(AdhocApproverDTO
adhocApprover) {
167             this.adhocApprovers.add(adhocApprover);
168 
169         }
170 
171         /**
172         * Add a list of adhoc approvers
173         * @param adhocApprovers list of adhoc approvers
174         */
175         global void addAdhocApprovers(List<AdhocApproverDTO>
adhocApprovers) {
176             this.adhocApprovers.addAll(adhocApprovers);
177 
178         }
179 
180     }
181 
182 
183     /**
184     * Inner class to hold adhoc approver
185     */
186     global class AdhocApproverDTO {
187 
188         // adhoc approver
189         public AdhocApprover__c adhocApproverSO = null;
190 
191         /**
192         * Class constructor
193         */
194         private AdhocApproverDTO() {
195 
196         }
197 
198         /**
199         * Class Constructor specifying initial values
200         * @param adhocApproverSO the adhoc approver
201         */
202         global AdhocApproverDTO(AdhocApprover__c adhocApproverSO)
{
203             this();
```

```
204             setAdhocApproverSO(adhocApproverSO);
205
206         }
207
208         /**
209          * Get adhoc approver
210          * @return the adhoc approver sobject
211          */
212         global AdhocApprover__c getAdhocApproverSO() {
213             return  adhocApproverSO;
214
215         }
216
217         /**
218          * Set adhoc approver
219          * @param adhocApprover the adhoc approver
220          */
221         global  void  setAdhocApproverSO(AdhocApprover__c
adhocApprover) {
222             this .adhocApproverSO = adhocApprover;
223
224         }
225
226     }
227
228 }
```

Update a Runtime Adhoc Approval Process

This API enables you to update a runtime adhoc approval process that can be used to submit for adhoc approvals. This API accepts an approval process structure as input parameter and returns the same structure with record IDs populated. For details on the input structure, refer to AdhocApprovalProcessDTO.

API	Signature
updateRuntimeAdhocApprovalProcess	*webService static Boolean updateRuntimeAdhocApprovalProcess(String sObjectType, Id sObjectId)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
adhocApproverProcessDTO	adhocApproverProcessDTO	Yes	The adhoc approver process structure to update.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	The runtime adhoc approval process structure for the updated process.

Retrieve a Runtime Adhoc Approval Process

This API enables you to retrieve a runtime adhoc approval process. This API accepts the context object type and id as input parameters and returns the adhoc process definition.

API	Signature
GetRuntimeAdhocApprovalProcess	*webService static Apttus_Approval.AdhocApprovalProcessDTO GetRuntimeAdhocApprovalProcess(String sObjectType, Id sObjectId)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectId	ID	Yes	Id of the approval context object.
sObjectType	Type	Yes	Type of the approval context object.

Response Parameter		
Name	Type	Description
result	Boolean	The runtime adhoc approval process structure.

Code Sample

```
    /**
     * Retrieve a runtime adhoc approval process
     * @param sObjectType the approval context sobject type
     * @param sObjectId the approval context sobject identifier
     * @return the runtime adhoc approval process structure
     */
    WebService static AdhocApprovalProcessDTO
GetRuntimeAdhocApprovalProcess(String sObjectType, ID sObjectId);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
< soapenv :Header>
< app :SessionHeader>
< sessionId >00DR0000001nyVR!
ARYAQLoAlVIxONFhJTAFfySXH7zWCeGqpdBT_w5DVKoZRR0WSbfTd.8KOe0PzjlQVcgkwAh2Xhxl
AkstCjM1D2ujPD56hZFJ</ sessionId >
</ app :SessionHeader>
```

```
</ soapenv :Header>
   < soapenv :Body>
      < app :GetRuntimeAdhocApprovalProcess>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R0000009z33H</ app :sObjectId>
      </ app :GetRuntimeAdhocApprovalProcess>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService"   xmlns:AdhocApprovalProcessDTO = "http://
soap.sforce.com/schemas/class/Apttus_Approval/AdhocApprovalProcessDTO" >
   < soapenv :Body>
      < GetRuntimeAdhocApprovalProcessResponse >
         < result />
      </ GetRuntimeAdhocApprovalProcessResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Delete a Runtime Adhoc Approval Process

This API enables you to delete a runtime adhoc approval process. This API accepts the context object type and id as input parameters and returns true if the adhoc process definition is successfully deleted.

API	Signature
DeleteRuntimeAdhocApprovalProcess	*webService static Boolean DeleteRuntimeAdhocApprovalProcess(String sObjectType, Id sObjectId)*

Request Parameters			
Name	**Type**	**Required?**	**Description**
sObjectId	ID	Yes	Id of the approval context object.
sObjectType	Type	Yes	Type of the approval context object.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the approval request is deleted successfully.

Code Sample

```
1   /**
2       * Delete a runtime adhoc approval process
3       * @param sObjectType the approval context sobject type
4       * @param sObjectId the approval context sobject identifier
5       * @return <code>true</code> if the process is deleted,
    <code>false</code> otherwise
6       */
7      WebService static Boolean
    DeleteRuntimeAdhocApprovalProcess(String sObjectType, ID
    sObjectId);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
< soapenv :Header>
< app :SessionHeader>

< sessionId >00DR0000001nyVR!
ARYAQLoAlVIxONFhJTAFfySXH7zWCeGqpdBT_w5DVKoZRR0WSbfTd.8KOe0PzjlQVcgkwAh2Xhxl
AkstCjM1D2ujPD56hZFJ</ sessionId >

</ app :SessionHeader>
</ soapenv :Header>
   < soapenv :Body>
      < app :DeleteRuntimeAdhocApprovalProcess>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R0000009z33H</ app :sObjectId>
      </ app :DeleteRuntimeAdhocApprovalProcess>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < DeleteRuntimeAdhocApprovalProcessResponse >
         < result >true</ result >
      </ DeleteRuntimeAdhocApprovalProcessResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Add Comment to a Runtime Adhoc Approval Process

This API enables you to add a comment to a runtime adhoc approval process. This API accepts the context object type, id, and comment as input parameters and returns true if the comments are successfully added.

API	Signature
AddCommentToRuntimeAdhocApprovalProcess	*webService static Boolean AddCommentToRuntimeAdhocApprovalProcess(String sObjectType, Id sObjectId, String sComment)*

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	Id of the approval context object.
sObjectType	Type	Yes	Type of the approval context object.
sComment	String	Yes	The comment to add.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the comment is added successfully.

Code Sample

```
1  /**
2       * Add comment to a runtime adhoc approval process
3       * @param sObjectType the approval context sobject type
4       * @param sObjectId the approval context sobject identifier
5       * @param sComment the comment to add
6       * @return <code>true</code> if the comment was added,
   <code>false</code> otherwise
7       */
```

```
8    WebService static Boolean
AddCommentToRuntimeAdhocApprovalProcess(String sObjectType, ID
sObjectId, String sComment);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
< soapenv :Header>
< app :SessionHeader>
< sessionId >00DR0000001nyVR!
ARYAQLoAlVIxONFhJTAFfySXH7zWCeGqpdBT_w5DVKoZRR0WSbfTd.8KOe0PzjlQVcgkwAh2Xhxl
AkstCjM1D2ujPD56hZFJ</ sessionId >
</ app :SessionHeader>
</ soapenv :Header>
   < soapenv :Body>
      < app :AddCommentToRuntimeAdhocApprovalProcess>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R0000009z33H</ app :sObjectId>
         < app :sComment>Approval Comment</ app :sComment>
      </ app :AddCommentToRuntimeAdhocApprovalProcess>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
    < soapenv :Body>
       < AddCommentToRuntimeAdhocApprovalProcessResponse >
          < result >true</ result >
       </ AddCommentToRuntimeAdhocApprovalProcessResponse >
    </ soapenv :Body>
</ soapenv :Envelope>
```

Add Attachments to a Runtime Adhoc Approval Process

This API enables you to add one or more attachments to a runtime adhoc approval process. This API accepts the context object type, id, and list of attachment IDs to add as input parameters and returns true if the attachments are successfully added.

API	Signature
AddAttachmentsToRuntimeAdhocApprovalProcess	*webService static Boolean AddAttachmentsToRuntimeAdhocApprovalProcess(String sObjectType, Id sObjectId, List* attachmentIds)

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	Id of the approval context object.
sObjectType	Type	Yes	Type of the approval context object.
attachmentIds	List	Yes	List of attachment Id's to add.

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the attachments are added successfully.

Code Sample

```
/**
     * Add attachments to a runtime adhoc approval process
     * @param sObjectType the approval context sobject type
     * @param sObjectId the approval context sobject identifier
     * @param attachmentIds the list of attachment ids to add
     * @return <code>true</code> if the attachments were added,
<code>false</code> otherwise
     */
    WebService static Boolean
AddAttachmentsToRuntimeAdhocApprovalProcess(String sObjectType, ID
sObjectId, List<ID> attachmentIds);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
```

```
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :AddAttachmentsToRuntimeAdhocApprovalProcess>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R0000009z33H</ app :sObjectId>
         <!--Zero or more repetitions:-->
         < app :attachmentIds>069R0000000ZIdt</ app :attachmentIds>
      </ app :AddAttachmentsToRuntimeAdhocApprovalProcess>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"
 xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < AddAttachmentsToRuntimeAdhocApprovalProcessResponse >
         < result >true</ result >
      </ AddAttachmentsToRuntimeAdhocApprovalProcessResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Delete Attachments to a Runtime Adhoc Approval Process

This API enables you to delete one or more attachments from a runtime adhoc approval process. This API accepts the context object type, id, and list of attachment IDs to delete as input parameters and returns true if the attachments were successfully deleted.

API	Signature
DeleteAttachmentsFromRuntimeAdhocApprovalProcess	*webService static Boolean DeleteAttachmentsFromRuntimeAdhocApprovalProcess(String sObjectType, Id sObjectId, List* attachmentIds)

Request Parameters

Name	Type	Required?	Description
sObjectId	ID	Yes	Id of the approval context object.
sObjectType	Type	Yes	Type of the approval context object.
attachmentIds	List	Yes	List of attachment Id's to remove.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the attachments are removed successfully.

Code Sample

```
/**
     * Delete attachments from a runtime adhoc approval process
     * @param sObjectType the approval context sobject type
     * @param sObjectId the approval context sobject identifier
     * @param attachmentIds the list of attachment ids to remove
     * @return <code>true</code> if the attachments were removed, <code>false</code> otherwise
     */
    WebService static Boolean DeleteAttachmentsFromRuntimeAdhocApprovalProcess(String sObjectType, ID sObjectId, List<ID> attachmentIds);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/
Apttus_Approval/ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQNCIk5WB9S9PlmZzS2uecBjxnhB20ndYmyxseH1LEfStQi1cXAtXiDxEyd3kAbYrAXbpJmDZ
zVXGxrxxleE8Fmm6kqSm</ app :sessionId>
      </ app :SessionHeader>
      </ soapenv :Header>
   < soapenv :Body>
      < app :DeleteAttachmentsFromRuntimeAdhocApprovalProcess>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R0000009z33H</ app :sObjectId>
      </ app :DeleteAttachmentsFromRuntimeAdhocApprovalProcess>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < DeleteAttachmentsFromRuntimeAdhocApprovalProcessResponse >
         < result >true</ result >
      </ DeleteAttachmentsFromRuntimeAdhocApprovalProcessResponse >
```

```
    </ soapenv :Body>
</ soapenv :Envelope>
```

Use Case and Sample Code for Adhoc Approval Runtime Process APIs

Adhoc Approval feature allows an approver to add a new approval step in an active approval process. For example, if, while approving a request, the Deal Desk concludes that a quote will require additional approvals from Legal and the CFO, creates those approval steps for a quote, decides the sequence for approval, and creates a new approval step. You can create and modify an adhoc approval process in runtime.

API flow for Adhoc Aprpoval Runtime Process:

- Create a Run-Time Adhoc Approval Process
- Add Comments
- Add Attachments
- Update the process
- Retrieve the adhoc process
- Delete attachments from an already existing adhoc process

This sample code shows how you can use the following 7 APIs to create, read, update, and retrieve Adhoc approval runtime process info and be able to add comments and attachments to it.

```
1  /**
2   *  Apttus Approvals Management
3   *  ApprovalsWebServiceTestSample
4   *
5   *  @2019 Apttus Inc. All rights reserved.
6   */
7  public  with sharing  class  AdhocApprovalsWebServiceTestSample {
8
9       /**
10      * Class Constructor
11      */
12      public  AdhocApprovalsWebServiceTestSample() {
13
14      }
15
```

```
16      // test webservice API calls by running as anonymous apex
from developer console
17      //
AdhocApprovalsWebServiceTestSample.TestCreateRuntimeAdhocApprovalP
rocess('Apttus__APTS_Agreement__c','a07q000000EZfki');
18      //
AdhocApprovalsWebServiceTestSample.TestUpdateRuntimeAdhocApprovalP
rocess('Apttus__APTS_Agreement__c','a07q000000EZfki');
19      //
AdhocApprovalsWebServiceTestSample.TestGetRuntimeAdhocApprovalProc
ess('Apttus__APTS_Agreement__c','a07q000000EZfki');
20      //
AdhocApprovalsWebServiceTestSample.TestAddCommentToRuntimeAdhocApp
rovalProcess('Apttus__APTS_Agreement__c','a07q000000EZfki');
21      //
AdhocApprovalsWebServiceTestSample.TestAddAttachmentsToRuntimeAdho
cApprovalProcess('Apttus__APTS_Agreement__c','a07q000000EZfki');
22      //
AdhocApprovalsWebServiceTestSample.TestDeleteAttachmentsFromRuntim
eAdhocApprovalProcess('Apttus__APTS_Agreement__c','a07q000000EZfki
');
23      //
AdhocApprovalsWebServiceTestSample.TestDeleteRuntimeAdhocApprovalP
rocess('Apttus__APTS_Agreement__c','a07q000000EZfki');
24
25      /**
26      * Create a runtime adhoc approval process
27      * @param sObjectType the approval context sobject type
28      * @param sObjectId the approval context sobject identifier
29      * @return the runtime adhoc approval process structure for
the created process
30      */
31      public   static   void
 TestCreateRuntimeAdhocApprovalProcess(String sObjectType, ID
sObjectId) {
32
33          // create insert structure
34         Apttus_Approval.AdhocApprovalProcessDTO adhocProcessDTO =
new  Apttus_Approval.AdhocApprovalProcessDTO();
35
36          // get users to add as approvers
37         List<User> users = [SELECT Id,Name
38                             FROM User
39                             WHERE UserType !=  'AutomatedProcess'
```

```
                                LIMIT  2 ];
        User userSO1 = users[ 0 ];
        User userSO2 = users[ 1 ];

         // add process
        Apttus_Approval__AdhocApprovalProcess__c adhocProcessSO =
new  Apttus_Approval__AdhocApprovalProcess__c(
            Id =  null ,
            Apttus_Approval__AttachmentIds__c =  null ,
            Apttus_Approval__BusinessObjectId__c = sObjectId,
            Apttus_Approval__BusinessObjectType__c = sObjectType,
            Apttus_Approval__DisplayFields__c =  null ,
            Apttus_Approval__DisplayHeaderFields__c =  null ,
            Apttus_Approval__ProcessComments__c =  null
        );
        adhocProcessDTO.setAdhocApprovalProcessSO(adhocProcessSO);
         // create group 1
        Apttus_Approval__AdhocApprovalGroup__c adhocGroupSO1 =
new  Apttus_Approval__AdhocApprovalGroup__c(
            Apttus_Approval__AdhocApprovalProcessId__c =  null ,
            Apttus_Approval__DependsOn__c =  null ,
            Apttus_Approval__GroupName__c =  'Group 1' ,
            Apttus_Approval__GroupSequence__c =  1
        );
         // add group 1 to process

Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO
adhocGroupDTO1 =  new

Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO(adho
cGroupSO1);
        adhocProcessDTO.addAdhocApprovalGroup(adhocGroupDTO1);
         // create group 1 approver 1
        Apttus_Approval__AdhocApprover__c adhocApproverSO1 =  new
 Apttus_Approval__AdhocApprover__c(
            Apttus_Approval__AdhocApprovalGroupId__c =  null ,
            Apttus_Approval__ApproverSequence__c =  1 ,
            Apttus_Approval__DependsOn__c =  null ,
            Apttus_Approval__AssigneeId__c = userSO1.Id,
            Apttus_Approval__AssigneeType__c =  'User' ,
            Apttus_Approval__AssigneeValue__c = userSO1.Name,
            Apttus_Approval__AutoReapprovalEnabled__c =  true ,
            Apttus_Approval__IsReviewer__c =  false
```

```
75          );
76           // add group 1 approver 1
77          Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO
adhocApproverDTO1 =  new
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO(adhocAppr
overSO1);
78          adhocGroupDTO1.addAdhocApprover(adhocApproverDTO1);
79           // create group 1 approver 2
80          Apttus_Approval__AdhocApprover__c adhocApproverSO2 =  new
 Apttus_Approval__AdhocApprover__c(
81              Apttus_Approval__AdhocApprovalGroupId__c =  null ,
82              Apttus_Approval__ApproverSequence__c =  2 ,
83              Apttus_Approval__DependsOn__c =  null ,
84              Apttus_Approval__AssigneeId__c = userSO2.Id,
85              Apttus_Approval__AssigneeType__c =  'User' ,
86              Apttus_Approval__AssigneeValue__c = userSO2.Name,
87              Apttus_Approval__AutoReapprovalEnabled__c =  true ,
88              Apttus_Approval__IsReviewer__c =  false
89          );
90           // add group 1 approver 2
91          Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO
adhocApproverDTO2 =  new
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO(adhocAppr
overSO2);
92          adhocGroupDTO1.addAdhocApprover(adhocApproverDTO2);
93           // create group 2
94          Apttus_Approval__AdhocApprovalGroup__c adhocGroupSO2 =
new  Apttus_Approval__AdhocApprovalGroup__c(
95              Apttus_Approval__AdhocApprovalProcessId__c =  null ,
96              Apttus_Approval__DependsOn__c =  null ,
97              Apttus_Approval__GroupName__c =  'Group 2' ,
98              Apttus_Approval__GroupSequence__c =  2
99          );
100          // add group 2 to process
101
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO
adhocGroupDTO2 =  new
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO(adho
cGroupSO2);
102          adhocProcessDTO.addAdhocApprovalGroup(adhocGroupDTO2);
103          // create group 2 approver 1
```

```
104         adhocApproverSO1 =  new
 Apttus_Approval__AdhocApprover__c(
105             Apttus_Approval__AdhocApprovalGroupId__c =  null ,
106             Apttus_Approval__ApproverSequence__c =  1 ,
107             Apttus_Approval__DependsOn__c =  null ,
108             Apttus_Approval__AssigneeId__c = userSO1.Id,
109             Apttus_Approval__AssigneeType__c =  'User' ,
110             Apttus_Approval__AssigneeValue__c = userSO1.Name,
111             Apttus_Approval__AutoReapprovalEnabled__c =  true ,
112             Apttus_Approval__IsReviewer__c =  false
113         );
114          // add group 2 approver 1
115         adhocApproverDTO1 =  new
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO(adhocAppr
overSO1);
116         adhocGroupDTO2.addAdhocApprover(adhocApproverDTO1);
117          // create group 2 approver 2
118         adhocApproverSO2 =  new
 Apttus_Approval__AdhocApprover__c(
119             Apttus_Approval__AdhocApprovalGroupId__c =  null ,
120             Apttus_Approval__ApproverSequence__c =  2 ,
121             Apttus_Approval__DependsOn__c =  null ,
122             Apttus_Approval__AssigneeId__c = userSO2.Id,
123             Apttus_Approval__AssigneeType__c =  'User' ,
124             Apttus_Approval__AssigneeValue__c = userSO2.Name,
125             Apttus_Approval__AutoReapprovalEnabled__c =  true ,
126             Apttus_Approval__IsReviewer__c =  false
127         );
128          // add group 2 approver 2
129          adhocApproverDTO2 =  new
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO(adhocAppr
overSO2);
130         adhocGroupDTO2.addAdhocApprover(adhocApproverDTO2);
131
132          // call API to create process
133         Apttus_Approval.AdhocApprovalProcessDTO
adhocProcessDTOResult =
Apttus_Approval.ApprovalsWebService.CreateRuntimeAdhocApprovalProc
ess(adhocProcessDTO);
134         system.debug( 'adhocProcessDTOResult=' +adhocProcessDTORes
ult);
```

```
135
136     }
137
138     /**
139     * Update a runtime adhoc approval process
140     * @param sObjectType the approval context sobject type
141     * @param sObjectId the approval context sobject identifier
142     * @return the runtime adhoc approval process structure for
    the edited process
143     */
144     public static void
     TestUpdateRuntimeAdhocApprovalProcess(String sObjectType, ID
    sObjectId) {
145
146         // create update structures
147         Apttus_Approval.AdhocApprovalProcessDTO adhocProcessDTO =
    new Apttus_Approval.AdhocApprovalProcessDTO();
148
    Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO
    adhocGroupDTO = null ;
149         Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO
    adhocApproverDTO = null ;
150
151         // get users to add as approvers
152         List<User> users = [SELECT Id,Name
153                             FROM User
154                             WHERE UserType != 'AutomatedProcess'
155                             LIMIT 5 ];
156         User userSO1 = users[ 0 ];
157         User userSO2 = users[ 1 ];
158         User userSO3 = users[ 2 ];
159         User userSO4 = users[ 3 ];
160         User userSO5 = users[ 4 ];
161
162         // get process
163         Apttus_Approval__AdhocApprovalProcess__c processSO =
    [SELECT Id,Name,
164
    Apttus_Approval__BusinessObjectType__c,
165
    Apttus_Approval__BusinessObjectId__c
166                                             FROM
    Apttus_Approval__AdhocApprovalProcess__c
```

```
167                                                 WHERE
    Apttus_Approval__BusinessObjectType__c = :sObjectType AND
168
    Apttus_Approval__BusinessObjectId__c = :sObjectId
169                                                 LIMIT  1 ];
170         system. assert (processSO !=  null );
171
172          // update process
173         Apttus_Approval__AdhocApprovalProcess__c adhocProcessSO =
    new  Apttus_Approval__AdhocApprovalProcess__c(
174             Id = processSO.Id,
175             Apttus_Approval__AttachmentIds__c =  null ,
176             Apttus_Approval__BusinessObjectId__c =
    processSO.Apttus_Approval__BusinessObjectId__c,
177             Apttus_Approval__BusinessObjectType__c =
    processSO.Apttus_Approval__BusinessObjectType__c,
178             Apttus_Approval__DisplayFields__c =  null ,
179             Apttus_Approval__DisplayHeaderFields__c =
    'Apttus__Total_Contract_Value__c' ,
180             Apttus_Approval__ProcessComments__c =  'new comment
    added'
181         );
182         adhocProcessDTO.setAdhocApprovalProcessSO(adhocProcessSO);
183
184          // get groups and approvers
185         List<Apttus_Approval__AdhocApprovalGroup__c> groups =
    [SELECT Id,
186                                                     Name,
187
    Apttus_Approval__AdhocApprovalProcessId__c,
188
    Apttus_Approval__AdhocApprovalProcessId__r.Apttus_Approval__Busine
    ssObjectId__c,
189
    Apttus_Approval__GroupName__c,
190
    Apttus_Approval__GroupSequence__c,
191
    Apttus_Approval__DependsOn__c,
192                                                     (SELECT Id,
193                                                             Name,
194
    Apttus_Approval__AdhocApprovalGroupId__c,
```

```
195 
    Apttus_Approval__ApproverSequence__c,
196 
    Apttus_Approval__DependsOn__c,
197 
    Apttus_Approval__AssigneeId__c,
198 
    Apttus_Approval__AssigneeType__c,
199 
    Apttus_Approval__AssigneeValue__c,
200 
    Apttus_Approval__AutoReapprovalEnabled__c,
201 
    Apttus_Approval__IsReviewer__c,
202 
    Apttus_Approval__SendEmail__c
203                                                 FROM
    Apttus_Approval__AdhocApprovers__r
204                                                 ORDER BY
    Apttus_Approval__ApproverSequence__c)
205                                             FROM
    Apttus_Approval__AdhocApprovalGroup__c
206                                             WHERE
    Apttus_Approval__AdhocApprovalProcessId__c = :processSO.Id
207                                             ORDER BY
    Apttus_Approval__GroupSequence__c
208                                             LIMIT  100 ];
209 
210         // iterate over groups to update
211         for  (Apttus_Approval__AdhocApprovalGroup__c groupSO :
    groups) {
212             Apttus_Approval__AdhocApprovalGroup__c adhocGroupSO =
     null ;
213              if  (groupSO.Apttus_Approval__GroupSequence__c ==  1 )
     {
214                 // create group 1
215                adhocGroupSO =  new
     Apttus_Approval__AdhocApprovalGroup__c(
216                     Id = groupSO.Id,
217                     Apttus_Approval__AdhocApprovalProcessId__c =
    processSO.Id,
218                     Apttus_Approval__DependsOn__c =  null ,
219                     Apttus_Approval__GroupName__c =  'Group 1 -
    Edited' ,
220                     Apttus_Approval__GroupSequence__c =  1
```

```
                );
                 // update group 1
                adhocGroupDTO =  new
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO(adho
cGroupSO);
adhocProcessDTO.addAdhocApprovalGroup(adhocGroupDTO);

                 // iterate over approvers to update
                 for  (Apttus_Approval__AdhocApprover__c
approverSO : groupSO.Apttus_Approval__AdhocApprovers__r) {
                    Apttus_Approval__AdhocApprover__c
adhocApproverSO =  null ;
                     if
 (approverSO.Apttus_Approval__ApproverSequence__c ==  1 ) {
                         // create group 1 approver 1
                        adhocApproverSO =  new
 Apttus_Approval__AdhocApprover__c(
                            Id = approverSO.Id,
Apttus_Approval__AdhocApprovalGroupId__c = groupSO.Id,
                            Apttus_Approval__ApproverSequence__c =
 1 ,
                            Apttus_Approval__DependsOn__c =  null ,
                            Apttus_Approval__AssigneeId__c =
userSO1.Id,
                            Apttus_Approval__AssigneeType__c =
'User' ,
                            Apttus_Approval__AssigneeValue__c =
userSO1.Name,
Apttus_Approval__AutoReapprovalEnabled__c =  true ,
                            Apttus_Approval__IsReviewer__c =
false
                        );

                    }  else   if
 (approverSO.Apttus_Approval__ApproverSequence__c ==  2 ) {
                         // create group 1 approver 2
                        adhocApproverSO =  new
 Apttus_Approval__AdhocApprover__c(
                            Id = approverSO.Id,
```

```
247                 Apttus_Approval__AdhocApprovalGroupId__c = groupSO.Id,
248                                 Apttus_Approval__ApproverSequence__c = 2 ,
249                                 Apttus_Approval__DependsOn__c = '1' ,
250                                 Apttus_Approval__AssigneeId__c = userSO2.Id,
251                                 Apttus_Approval__AssigneeType__c = 'User' ,
252                                 Apttus_Approval__AssigneeValue__c = userSO2.Name,
253                 Apttus_Approval__AutoReapprovalEnabled__c = true ,
254                                 Apttus_Approval__IsReviewer__c = false
255                             );
256
257                     }
258
259                     // update approver
260                     adhocApproverDTO = new Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO(adhocApproverSO);
261                 adhocGroupDTO.addAdhocApprover(adhocApproverDTO);
262
263                 }
264
265             } else if (groupSO.Apttus_Approval__GroupSequence__c == 2 ) {
266                 // create group 2
267                 adhocGroupSO = new Apttus_Approval__AdhocApprovalGroup__c(
268                     Id = groupSO.Id,
269                     Apttus_Approval__AdhocApprovalProcessId__c = processSO.Id,
270                     Apttus_Approval__DependsOn__c = null ,
271                     Apttus_Approval__GroupName__c = 'Group 2 - Edited' ,
272                     Apttus_Approval__GroupSequence__c = 2
273                 );
274                 // update group 2
275                 adhocGroupDTO = new
```

```
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO(adhocGroupSO);
276
adhocProcessDTO.addAdhocApprovalGroup(adhocGroupDTO);
277
278                 // iterate over approvers to update
279                 for  (Apttus_Approval__AdhocApprover__c approverSO : groupSO.Apttus_Approval__AdhocApprovers__r) {
280                     Apttus_Approval__AdhocApprover__c adhocApproverSO =  null ;
281                      if (approverSO.Apttus_Approval__ApproverSequence__c ==  1 ) {
282                         // create group 2 approver 1
283                         adhocApproverSO =  new Apttus_Approval__AdhocApprover__c(
284                             Id = approverSO.Id,
285                             Apttus_Approval__AdhocApprovalGroupId__c = groupSO.Id,
286                             Apttus_Approval__ApproverSequence__c = 1 ,
287                             Apttus_Approval__DependsOn__c =  null ,
288                             Apttus_Approval__AssigneeId__c = userSO3.Id,
289                             Apttus_Approval__AssigneeType__c = 'User' ,
290                             Apttus_Approval__AssigneeValue__c = userSO3.Name,
291                             Apttus_Approval__AutoReapprovalEnabled__c =  true ,
292                             Apttus_Approval__IsReviewer__c = false
293                         );
294
295                     }  else   if (approverSO.Apttus_Approval__ApproverSequence__c ==  2 ) {
296                         // create group 2 approver 2
297                         adhocApproverSO =  new Apttus_Approval__AdhocApprover__c(
298                             Id = approverSO.Id,
299                             Apttus_Approval__AdhocApprovalGroupId__c = groupSO.Id,
300                             Apttus_Approval__ApproverSequence__c = 2 ,
```

```
301                             Apttus_Approval__DependsOn__c = '1',
302                             Apttus_Approval__AssigneeId__c =
userSO4.Id,
303                             Apttus_Approval__AssigneeType__c =
'User',
304                             Apttus_Approval__AssigneeValue__c =
userSO4.Name,
305
Apttus_Approval__AutoReapprovalEnabled__c = true,
306                             Apttus_Approval__IsReviewer__c =
false
307                         );
308
309                     }
310
311                     // update approver
312                     adhocApproverDTO = new
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO(adhocAppr
overSO);
313
adhocGroupDTO.addAdhocApprover(adhocApproverDTO);
314
315                 }
316
317                 // create group 2 approver 3
318                 Apttus_Approval__AdhocApprover__c adhocApproverSO
= new Apttus_Approval__AdhocApprover__c(
319                     Id = null,
320                     Apttus_Approval__AdhocApprovalGroupId__c =
groupSO.Id,
321                     Apttus_Approval__ApproverSequence__c = 3,
322                     Apttus_Approval__DependsOn__c = '2',
323                     Apttus_Approval__AssigneeId__c = userSO5.Id,
324                     Apttus_Approval__AssigneeType__c = 'User',
325                     Apttus_Approval__AssigneeValue__c =
userSO5.Name,
326                     Apttus_Approval__AutoReapprovalEnabled__c =
true,
327                     Apttus_Approval__IsReviewer__c = false
328                 );
329                 // update group 2 approver 3
330                 adhocApproverDTO = new
```

```
Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO(adhocApproverSO);
                adhocGroupDTO.addAdhocApprover(adhocApproverDTO);

            }

        }

        // print input structure
        PrintAdhocApprovalProcess(adhocProcessDTO);

        // call API to update process
        Apttus_Approval.AdhocApprovalProcessDTO adhocProcessDTOResult = Apttus_Approval.ApprovalsWebService.UpdateRuntimeAdhocApprovalProcess(adhocProcessDTO);

        // print output structure
        PrintAdhocApprovalProcess(adhocProcessDTOResult);

    }

    /**
    * Retrieve a runtime adhoc approval process
    * @param sObjectType the approval context sobject type
    * @param sObjectId the approval context sobject identifier
    * @return the runtime adhoc approval process structure
    */
    public static void TestGetRuntimeAdhocApprovalProcess(String sObjectType, ID sObjectId) {

        // call API to get process
        Apttus_Approval.AdhocApprovalProcessDTO adhocProcessDTOResult = Apttus_Approval.ApprovalsWebService.GetRuntimeAdhocApprovalProcess(sObjectType, sObjectId);

        // print output structure
        PrintAdhocApprovalProcess(adhocProcessDTOResult);

    }

```

```
364      /**
365      * Delete a runtime adhoc approval process
366      * @param sObjectType the approval context sobject type
367      * @param sObjectId the approval context sobject identifier
368      * @return <code>true</code> if the process is deleted,
<code>false</code> otherwise
369      */
370      public static void
 TestDeleteRuntimeAdhocApprovalProcess(String sObjectType, ID
sObjectId) {
371
372          // call API to delete process
373          Boolean ok =
Apttus_Approval.ApprovalsWebService.DeleteRuntimeAdhocApprovalProc
ess(sObjectType, sObjectId);
374          system.assertEquals(ok, true );
375
376      }
377
378      /**
379      * Add comment to a runtime adhoc approval process
380      * @param sObjectType the approval context sobject type
381      * @param sObjectId the approval context sobject identifier
382      * @return <code>true</code> if the comment was added,
<code>false</code> otherwise
383      */
384      public static void
 TestAddCommentToRuntimeAdhocApprovalProcess(String sObjectType,
ID sObjectId) {
385
386          // call API to add comment
387          Boolean ok =
Apttus_Approval.ApprovalsWebService.AddCommentToRuntimeAdhocApprov
alProcess(sObjectType, sObjectId, 'MY NEW comment' );
388          system.assertEquals(ok, true );
389
390          // call API to get process
391          Apttus_Approval.AdhocApprovalProcessDTO
adhocProcessDTOResult =
Apttus_Approval.ApprovalsWebService.GetRuntimeAdhocApprovalProcess
(sObjectType, sObjectId);
392          // get comment from process
393          Apttus_Approval__AdhocApprovalProcess__c adhocProcessSO =
adhocProcessDTOResult.getAdhocApprovalProcessSO();
```

```
394         String comment = adhocProcessSO.Apttus_Approval__ProcessComments__c;
395         system.assertEquals(comment,  'MY NEW comment' );
396 
397     }
398 
399     /**
400     * Add attachments to a runtime adhoc approval process
401     * @param sObjectType the approval context sobject type
402     * @param sObjectId the approval context sobject identifier
403     * @return <code>true</code> if the attachments were added, <code>false</code> otherwise
404     */
405     public  static  void  TestAddAttachmentsToRuntimeAdhocApprovalProcess(String sObjectType, ID sObjectId) {
406 
407         // create list of attachments to add
408         List<ID> attachmentIds =  new  List<ID>{sObjectId, sObjectId};
409         // call API to add attachments
410         Boolean ok = Apttus_Approval.ApprovalsWebService.AddAttachmentsToRuntimeAdhocApprovalProcess(sObjectType, sObjectId, attachmentIds);
411         system.assertEquals(ok,  true );
412 
413         // call API to get process
414         Apttus_Approval.AdhocApprovalProcessDTO adhocProcessDTOResult = Apttus_Approval.ApprovalsWebService.GetRuntimeAdhocApprovalProcess(sObjectType, sObjectId);
415         // get attachments from process
416         Apttus_Approval__AdhocApprovalProcess__c adhocProcessSO = adhocProcessDTOResult.getAdhocApprovalProcessSO();
417         String attachmentIdsStr = adhocProcessSO.Apttus_Approval__AttachmentIds__c;
418         String attachmentIdsStrExpected = sObjectId +  ','  + sObjectId;
419         system.assertEquals(attachmentIdsStr, attachmentIdsStrExpected);
420 
421     }
422 
```

```
423      /**
424      * Delete attachments from a runtime adhoc approval process
425      * @param sObjectType the approval context sobject type
426      * @param sObjectId the approval context sobject identifier
427      * @return <code>true</code> if the attachments were removed,
<code>false</code> otherwise
428      */
429      public static void
 TestDeleteAttachmentsFromRuntimeAdhocApprovalProcess(String
sObjectType, ID sObjectId) {
430
431          // create list of attachments to delete
432          List<ID> attachmentIds =  new  List<ID>{sObjectId};
433          // call API to delete attachments
434          Boolean ok =
Apttus_Approval.ApprovalsWebService.DeleteAttachmentsFromRuntimeAd
hocApprovalProcess(sObjectType, sObjectId, attachmentIds);
435          system.assertEquals(ok,  true );
436
437          // call API to get process
438          Apttus_Approval.AdhocApprovalProcessDTO
adhocProcessDTOResult =
Apttus_Approval.ApprovalsWebService.GetRuntimeAdhocApprovalProcess
(sObjectType, sObjectId);
439          // get attachments from process
440          Apttus_Approval__AdhocApprovalProcess__c adhocProcessSO =
adhocProcessDTOResult.getAdhocApprovalProcessSO();
441          String attachmentIdsStr =
adhocProcessSO.Apttus_Approval__AttachmentIds__c;
442          system.assertEquals(attachmentIdsStr, sObjectId);
443
444      }
445
446      /**
447      * Prints an adhoc process definition
448      * @param approvalProcessDTO the adhoc approval process
definition
449      */
450      public static void
 PrintAdhocApprovalProcess(Apttus_Approval.AdhocApprovalProcessDTO
approvalProcessDTO) {
451
452          // print process info
453          Apttus_Approval__AdhocApprovalProcess__c processSO =
approvalProcessDTO.getAdhocApprovalProcessSO();
```

```
454         system.debug( 'processSO=' +processSO);
455 
456          // print approval groups and approvers
457          for
  (Apttus_Approval.AdhocApprovalProcessDTO.AdhocApprovalGroupDTO
groupDTO : approvalProcessDTO.getAdhocApprovalGroups()) {
458             Apttus_Approval__AdhocApprovalGroup__c groupSO =
groupDTO.getAdhocApprovalGroupSO();
459             system.debug( 'groupSO=' +groupSO);
460 
461              // get approvers
462              for
  (Apttus_Approval.AdhocApprovalProcessDTO.AdhocApproverDTO
approverDTO : groupDTO.getAdhocApprovers()) {
463                 Apttus_Approval__AdhocApprover__c approverSO =
approverDTO.getAdhocApproverSO();
464                 system.debug( 'approverSO=' +approverSO);
465 
466             }
467 
468         }
469 
470     }
471 
472 }
```

Submitting Bulk Approval Requests

With this API, you can submit bulk approval requests for review. This API accepts the object type and the object ID of the context object as input parameters. You can also add comments to your approval requests. You cannot submit approval requests with attachments.

If you submit previously submitted or approved approvals as part of bulk approval request, the approval request is submitted again.

This API is useful when you need to submit a large number of approval requests for various context objects. For example, you are the approvals admin for a data service organization and you need to submit all the approval records for 5 agreements. Using submiForApprovalsBulk API, you can submit all the approval records in one go by calling this API and providing the type and IDs of the five agreements as request parameters.

API	Signature
submitForApprovalsBulk	*webService static Boolean submitForApprovalsBulk(List* ctxObjParams)

Request Parameters

Name	Type	Required?	Description
ctxObjParams	Object	Yes	Object containing all the approval parameters

approvalParams

Name	Type	Required?	Description
approvalObjectId	ID	Yes	ID of the approval context object.
approvalObjectType	String	Yes	Type of the approval context object.
approvalCommentsObject	Object	No	The approval comment object

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the API is executed successfully.

```
/**
 *  Apttus Approvals Management
 *  BulkCtxObjectParam
 *
 *  @2020 Apttus Inc. All rights reserved.
 */
global with sharing class BulkCtxObjectParam {

    /**
     * Context Object Type
```

```
     */
    webservice String ctxSObjectType { global get; global set; }

     /**
     * Context Object Id
     */
    webservice ID ctxSObjectId { global get; global set; }

     /**
     * Submission Comments
     */
    webservice SubmissionComments comments { global get; global set; }

     /**
     * Global Constructor
     */
    global BulkCtxObjectParam() {
        ctxSObjectType =  null ;
        ctxSObjectId =  null ;
        comments =  null ;

    }

     /**
     * Creates the json representation of the bulk context objects
     * @return the json string
     */
    global String toJSON() {
         // get the json representation
         return  System.JSON.serializePretty( this );

    }

     /**
     * Parses the bulk context object param from the given JSON string
     * @param jsonString the json string to parse
     * @return the bulk context param object
     */
    global  static  BulkCtxObjectParam parse(String jsonString) {
         // load the class (salesforce class loading bug)
        System.Type wrapperType =
System.Type.forName(SystemUtil.getFQClassName( 'BulkCtxObjectParam' ));
         // deserialize the object
```

```
        return  (BulkCtxObjectParam) System.Json.deserialize(jsonString,
BulkCtxObjectParam. class );

    }

}
// multiple context object type and ids
String agmtType =  'Apttus__APTS_Agreement__c' ;
ID agmtId1 =  'a07R000000APxGG' ;
ID agmtId2 =  'a07R000000AS9nJ' ;
ID agmtId3 =  'a07R000000AS9nO' ;

// submission comments
Apttus_Approval.SubmissionComments submitComments1 =  new
 Apttus_Approval.SubmissionComments();
submitComments1.commentsLevel =
Apttus_Approval.SubmissionComments.PROCESS_LEVEL_COMMENTS;
submitComments1.commentsCount =  1 ;
submitComments1.processName =  'ProcessName' ;
submitComments1.processCommentLabel =  'ProcessLabel' ;
submitComments1.processCommentMandatory =  false ;
submitComments1.processComment =  'Test Comment 1' ;

Apttus_Approval.SubmissionComments submitComments2 =  new
 Apttus_Approval.SubmissionComments();
submitComments2.commentsLevel =
Apttus_Approval.SubmissionComments.PROCESS_LEVEL_COMMENTS;
submitComments2.commentsCount =  1 ;
submitComments2.processName =  'ProcessName' ;
submitComments2.processCommentLabel =  'ProcessLabel' ;
submitComments2.processCommentMandatory =  false ;
submitComments2.processComment =  'Test Comment 2' ;

Apttus_Approval.SubmissionComments submitComments3 =  null ;


// create list of context objects
List<Apttus_Approval.BulkCtxObjectParam> ctxParams =  new
 List<Apttus_Approval.BulkCtxObjectParam>();

Apttus_Approval.BulkCtxObjectParam ctxParam1 =  new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam1.ctxSObjectType = agmtType;
ctxParam1.ctxSObjectId = agmtId1;
ctxParam1.comments = submitComments1;
```

```
ctxParams.add(ctxParam1);

Apttus_Approval.BulkCtxObjectParam ctxParam2 = new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam2.ctxSObjectType = agmtType;
ctxParam2.ctxSObjectId = agmtId2;
ctxParam2.comments = submitComments2;
ctxParams.add(ctxParam2);

Apttus_Approval.BulkCtxObjectParam ctxParam3 = new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam3.ctxSObjectType = agmtType;
ctxParam3.ctxSObjectId = agmtId3;
ctxParam3.comments = submitComments3;
ctxParams.add(ctxParam3);

// bulk submit
Boolean ok =
Apttus_Approval.ApprovalsWebService.submitForApprovalsBulk(ctxParams);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService"
 xmlns:bul = "http://soap.sforce.com/schemas/class/Apttus_Approval/
BulkCtxObjectParam"
```

```
xmlns:sub = "http://soap.sforce.com/schemas/class/Apttus_Approval/
SubmissionComments" >
   < soapenv :Header>
      < app :SessionHeader>
         < app :sessionId>00DR0000001nyVR!
ARYAQOruA8X3nIS2hgZgZxEZkUDHDMlzIfBNGDey8s_.AcTbbEghNGUnMEh5oGcG5mkmrVuHp1F9
gHzfIfYAvzuRDU6zg7kO</ app :sessionId>
      </ app :SessionHeader>
   </ soapenv :Header>
   < soapenv :Body>
      < app :submitForApprovalsBulk>
         <!--Zero or more repetitions:-->
         < app :ctxObjParams>
            < bul :comments>
               < sub :commentsCount>1</ sub :commentsCount>
               < sub :commentsLevel>Process</ sub :commentsLevel>
               < sub :processComment>?</ sub :processComment>
               < sub :processCommentLabel>ProcessLabel</ sub :processComment
Label>
               < sub :processCommentMandatory>false</ sub :processCommentMan
datory>
               < sub :processName>ProcessName</ sub :processName>
               <!--Zero or more repetitions:-->
               < sub :stepCommentList>
               </ sub :stepCommentList>
            </ bul :comments>
            < bul :ctxSObjectId>a07R000000AiYQX</ bul :ctxSObjectId>
            < bul :ctxSObjectType>Apttus__APTS_Agreement__c</ bul :ctxSObjec
tType>
         </ app :ctxObjParams>
         < app :ctxObjParams>
            < bul :comments />
            < bul :ctxSObjectId>a07R000000Aj68Y</ bul :ctxSObjectId>
            < bul :ctxSObjectType>Apttus__APTS_Agreement__c</ bul :ctxSObjec
tType>
         </ app :ctxObjParams>
      </ app :submitForApprovalsBulk>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < submitForApprovalsBulkResponse >
         < result >true</ result >
      </ submitForApprovalsBulkResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Recalling Bulk Approval Requests

With this API, you can cancel or recall previously submitted bulk approval requests. This API accepts the object type and the object ID of the context object as input parameters. You can also add comments to your cancel requests.

If you submit previously submitted or approved approvals as part of bulk cancel requests, all the approvals are canceled.

This API is useful when you need to recall a large number of approval requests for various context objects. For example, you are the approvals admin for a data service organization and you need to recall all the approval records for 5 agreements. Using cancelApprovalsBulk API, you can recall all the approval records in one go by calling this API and providing the type and IDs of the five agreements as request parameters.

API	Signature
cancelApprovalsBulk	*webService static Boolean cancelApprovalsBulk(List* ctxObjParams)

Request Parameters

Name	Type	Required?	Description
approvalParams	Object	Yes	Object containing all the approval parameters

approvalParams			
Name	**Type**	**Required?**	**Description**
approvalObjectId	ID	Yes	ID of the approval context object.
approvalObjectType	String	Yes	Type of the approval context object.
approvalCommentsObject	Object	No	The approval comment object

Response Parameter		
Name	**Type**	**Description**
result	Boolean	Returns true if the API is executed successfully.

```
// multiple context object type and ids
String agmtType =  'Apttus__APTS_Agreement__c' ;
ID agmtId1 =  'a07R000000APxGG' ;
ID agmtId2 =  'a07R000000AS9nJ' ;
ID agmtId3 =  'a07R000000AS9nO' ;


// create list of context objects
List<Apttus_Approval.BulkCtxObjectParam> ctxParams =  new
 List<Apttus_Approval.BulkCtxObjectParam>();

Apttus_Approval.BulkCtxObjectParam ctxParam1 =  new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam1.ctxSObjectType = agmtType;
ctxParam1.ctxSObjectId = agmtId1;
ctxParam1.comments = submitComments1;
ctxParams.add(ctxParam1);

Apttus_Approval.BulkCtxObjectParam ctxParam2 =  new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam2.ctxSObjectType = agmtType;
ctxParam2.ctxSObjectId = agmtId2;
ctxParam2.comments = submitComments2;
```

```
ctxParams.add(ctxParam2);

Apttus_Approval.BulkCtxObjectParam ctxParam3 = new
 Apttus_Approval.BulkCtxObjectParam();
ctxParam3.ctxSObjectType = agmtType;
ctxParam3.ctxSObjectId = agmtId3;
ctxParam3.comments = submitComments3;
ctxParams.add(ctxParam3);

// bulk submit
Boolean ok =
Apttus_Approval.ApprovalsWebService.cancelApprovalsBulk(ctxParams);
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService"
 xmlns:bul = "http://soap.sforce.com/schemas/class/Apttus_Approval/
BulkCtxObjectParam"
 xmlns:sub = "http://soap.sforce.com/schemas/class/Apttus_Approval/
SubmissionComments" >
    < soapenv :Header>
       < app :SessionHeader>
          < app :sessionId>00DR0000001nyVR!
ARYAQOruA8X3nIS2hgZgZxEZkUDHDMlzIfBNGDey8s_.AcTbbEghNGUnMEh5oGcG5mkmrVuHp1F9
gHzfIfYAvzuRDU6zg7kO</ app :sessionId>
```

```
      </ app :SessionHeader>
   </ soapenv :Header>
   < soapenv :Body>
      < app :cancelApprovalsBulk>
         <!--Zero or more repetitions:-->
         < app :ctxObjParams>
            < bul :comments>
               < sub :commentsCount>1</ sub :commentsCount>
               < sub :commentsLevel>Process</ sub :commentsLevel>
               < sub :processComment>?</ sub :processComment>
               < sub :processCommentLabel>ProcessLabel</ sub :processComment
Label>
               < sub :processCommentMandatory>false</ sub :processCommentMan
datory>
               < sub :processName>ProcessName</ sub :processName>
               <!--Zero or more repetitions:-->
               < sub :stepCommentList>
               </ sub :stepCommentList>
            </ bul :comments>
            < bul :ctxSObjectId>a07R000000AiYQX</ bul :ctxSObjectId>
            < bul :ctxSObjectType>Apttus__APTS_Agreement__c</ bul :ctxSObjec
tType>
         </ app :ctxObjParams>
         < app :ctxObjParams>
            < bul :comments />
            < bul :ctxSObjectId>a07R000000Aj68Y</ bul :ctxSObjectId>
            < bul :ctxSObjectType>Apttus__APTS_Agreement__c</ bul :ctxSObjec
tType>
         </ app :ctxObjParams>
      </ app :cancelApprovalsBulk>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < cancelApprovalsBulkResponse >
         < result >true</ result >
      </ cancelApprovalsBulkResponse >
   </ soapenv :Body>
```

```
</soapenv:Envelope>
```

Submitting Approvals Asynchronously

With this API, you can submit approval requests asynchronously. This API accepts the object type and the object ID of the context object as input parameters.

API	Signature
submitForApprovalsAsync	*WebService static Boolean submitForApprovalsAsync(String sObjectType, ID sObjectId);*

Request Parameters

Name	Type	Required?	Description
approvalObjectId	ID	Yes	ID of the approval context object.
approvalObjectType	String	Yes	Type of the approval context object.

Response Parameter

Name	Type	Description
result	Boolean	Returns true if the API is executed successfully.

```
/**
 *  Apttus Approvals Management
 *  TestSubmitAsyncAPISample
 *
 *  @2020 Apttus Inc. All rights reserved.
 */
public with sharing class TestSubmitAsyncAPISample {

    /**
    * Class Constructor
    */
```

```
    public TestSubmitAsyncAPISample() {

    }

    // test webservice API calls by running as anonymous apex from
developer console
    //
TestSubmitAsyncAPISample.TestSubmitForApprovalsAsync('Apttus__APTS_Agreement
__c','a07R000000AiYQX');

    /**
    * Submit an approvals process asynchronously
    * @param sObjectType the approval context sobject type
    * @param sObjectId the approval context sobject identifier
    * @return true if successful
    */
    public static void TestSubmitForApprovalsAsync(String sObjectType,
ID sObjectId) {

        // call API to submit async
        Boolean ok =
Apttus_Approval.ApprovalsWebService.SubmitForApprovalsAsync(sObjectType,
sObjectId);
        system.assertEquals(ok, true );

    }

}
```

Integration Details

Use the following information in your integrations with Conga Approvals 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:app = "http://soap.sforce.com/schemas/class/
Apttus_Approval/ApprovalsWebService" >
   < soapenv :Header>
      < app :SessionHeader>
         < sessionId >00DR0000001nyVR!
ARYAQDLDaz7Bw9IfDVuVTeyiZwGL.4Oc2G8tgKIoJo0EEw9vlWQVnlEawyHu5UcIu.AUay3cBSoN
FGd8OtrxX.D7yjqBK9SL</ sessionId >
      </ app :SessionHeader>
   </ soapenv :Header>
   < soapenv :Body>
      < app :submitForApprovalsAsync>
         < app :sObjectType>Apttus__APTS_Agreement__c</ app :sObjectType>
         < app :sObjectId>a07R000000AiYQX</ app :sObjectId>
      </ app :submitForApprovalsAsync>
   </ soapenv :Body>
</ soapenv :Envelope>
```

Example Response

```
< soapenv :Envelope  xmlns:soapenv = "http://schemas.xmlsoap.org/soap/
envelope/"   xmlns = "http://soap.sforce.com/schemas/class/Apttus_Approval/
ApprovalsWebService" >
   < soapenv :Body>
      < submitForApprovalsAsyncResponse >
         < result >true</ result >
      </ submitForApprovalsAsyncResponse >
   </ soapenv :Body>
</ soapenv :Envelope>
```

Approvals Features by Release

Review the latest Conga Approvals Features by Release document.

- Features by Release

Features by Release

This document contains an overview of features introduced in each major release of Conga Approvals. For more information, see Conga Approvals Features by Release.