



Apttus TurboEngines Summer 2020

Administrator Guide



Table of Contents

- About This Guide..... 3
- What's New 5
- About TurboEngines7
 - About TurboConfig.....8
 - Supported Features and Capabilities in Apttus TurboConfig.....8
 - About TurboPricing..... 11
 - Supported Features in Apttus TurboPricing.....12
- Configuring Apttus TurboEngines..... 20
 - Enabling TurboEngines in an Org.....20
 - Prerequisites.....20
 - Enabling TurboEngines 21
 - Creating a Connected App24
 - To create a Connected App24
 - To capture Consumer Key and Consumer Secret.....25
 - Preparing Tenant Information25
 - Post-Provisioning Tasks (TurboConfig)..... 27
 - Configuring Remote Site Settings for TurboConfig.....27
 - Configuring TurboConfig Settings28
 - Configuring Custom Flows for TurboConfig.....29
 - Syncing TurboConfig Data.....29
 - Post-Provisioning Tasks (TurboPricing).....30
 - Setting Up the TurboPricing Endpoint URL.....30
 - Setting Up the Pricing Execution Mode.....30
 - Configuring Data Sync for TurboPricing 31
 - Configuring TurboPricing Callbacks.....34
- Frequently Asked Questions (TurboConfig)..... 61
- Apttus Copyright Disclaimer 64

About This Guide

Apttus TurboEngines Administrator Guide provides information to configure TurboEngines: TurboConfig and TurboPricing. Application administrators and Apttus customer administrators can also use content in this guide to perform updates to configurations, configure settings, and other microservice functionalities.

Topic	Description
What's Covered	This guide provides information for authorized administrators to deploy and configure Apttus TurboEngines for integrated systems.
Primary Audience	<ul style="list-style-type: none"> • TurboEngines Implementation Teams • Customer Administrators
IT Environment	Refer to the latest <i>Apttus TurboEngines Summer 2020 Release Notes</i> for information on System Requirements and Supported Platforms.
Other Resources	<ul style="list-style-type: none"> • <i>Apttus TurboEngines Data Sync Administrator Guide</i> • <i>Apttus TurboEngines Release Notes</i> • <i>Apttus TurboEngines REST API Guide</i> • <i>Apttus CPQ on Salesforce Administrator Guide</i>

This guide describes the following tasks:

- Reviewing the list of supported features for TurboConfig and TurboPricing
- Configuring Apttus TurboEngines
- Enabling TurboEngines
- Completing Pre-Provisioning Tasks
 - Creating a Connected App
 - Preparing Tenant Information
- Completing Post-Provisioning Tasks for TurboConfig
 - Configuring Remote Site Settings
 - Configuring Custom Settings
 - Configuring Custom Flows
 - Syncing TurboConfig Product Data
- Completing Post-Provisioning Tasks for TurboPricing
 - Configuring TurboPricing Settings
 - Customizing TurboPricing Callbacks
 - Configuring data sync settings

- Syncing TurboPricing Pricing Data

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

- Basic Salesforce administration
- Salesforce Lightning experience
- Salesforce and Apttus terms and definitions

What's New

The following table lists changes in documentation to support each release.

Document	Topic	Description
Summer 2020 Rev C	About this Guide	Updated Topic.
Summer 2020 Rev B	Validation Callback Class	New Topic.
	Supported features in Apttus TurboPricing	Updated Topic. Updated the supported features table.
	DBHelper	Updated Topic. Updated the topic with complex query example.
Summer 2020 Rev A	Configuring Data Sync for TurboPricing	Updated "To configure the service URL".
	Navigating the TurboPricing Callback Administrator User Interface	Replaced screenshots.
	Managing TurboPricing Callbacks	Replaced screenshots.
	Helper Functions	Renamed the topic to "Helper Functions for TurboPricing Callbacks" and moved the new topic under "Configuring TurboPricing Callbacks".
	Helper Functions for TurboPricing Callbacks	Renamed from "Helper Functions" and moved out of "Pricing Callback Class for TurboPricing".
	CacheHelper	New topic, moved out of "Helper Functions for TurboPricing Callbacks".
	DBHelper	New topic, moved out of "Helper Functions for TurboPricing Callbacks".
	HttpHelper	New topic, moved out of "Helper Functions for TurboPricing Callbacks".
	LogHelper	New topic, moved out of "Helper Functions for TurboPricing Callbacks".

Document	Topic	Description
	MetadataHelper	New topic, moved out of "Helper Functions for TurboPricing Callbacks".
	PricingHelper	New topic, moved out of "Helper Functions for TurboPricing Callbacks".
	VaultHelper	New topic, moved out of "Helper Functions for TurboPricing Callbacks".
Summer 2020	All topics	First release

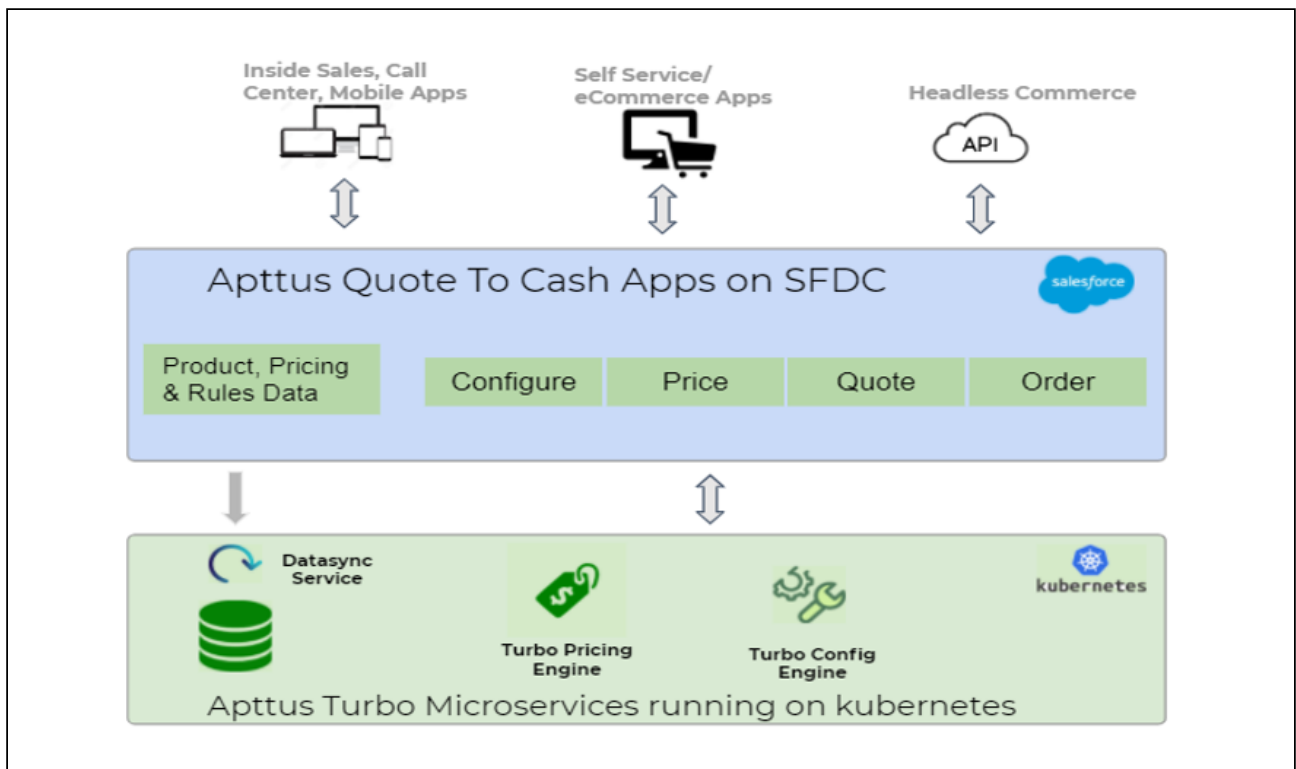
About TurboEngines

Apttus TurboEngines is a concurrent processing engine provided by Apttus that comprises various microservices that process product configurations (**TurboConfig**), pricing calculations (**TurboPricing**), and other product-related business data, such as promotions. Apttus TurboEngines offload the computation workload from the Salesforce platform to the **Apttus Flexible Compute Platform** to reduce the processing time on the cart. Processing the computation workload in the Apttus Flexible Compute Platform reduces the interaction costs and the quote turnaround time specifically during peak load or large transactions.

TurboEngines scale on the following dimensions:

- Number of users
- Size of transaction
- The complexity of the product and rules

TurboEngines also provide a critical component called **TurboEngines Data Sync** services that provide a high-performance mechanism to sync pricing master data at regular, scheduled intervals (or on-demand) between Salesforce and the Apttus Flexible Compute Platform. Data is pushed to TurboPricing consumer endpoints and made available for processing to take advantage of the performance improvements offered by the TurboEngines platform.



About TurboConfig


TurboConfig is a configuration engine created to process product configuration rules when products and bundles are configured on a cart and when finalizing the quote. TurboConfig offloads the computation workload from the Salesforce platform to the Apttus Flexible Compute Platform built using microservices to reduce the processing time of the configuration rules. Computation workload includes the processing of rules defined on the products. For example, in a TurboConfig enabled flow, when the Sales rep adds the product or the favorite configuration to cart, the constraint rules associated with them are offloaded to the Apttus Flexible Compute Platform to process. TurboConfig engine executes the rules, maintains rule states, and avoid unnecessary line item processing.

TurboConfig is recommended when you have a large number of rules or highly complex configuration rules to be applied while selecting a product or configuring a bundle.

To get started enabling TurboConfig for your org, refer to [Enabling TurboEngines in an Org](#). To learn more about the TurboConfig service, refer to [Frequently Asked Questions \(TurboConfig\)](#).

Supported Features and Capabilities in Apttus TurboConfig

The following features and its capabilities are supported when TurboConfig mode is enabled. For information on the listed features, refer to *CPQ on Salesforce Administrator Guide*.

 You cannot use TurboConfig and TurboPricing simultaneously.

Feature	Capability	Supported
Constraint Rules	Inclusion rules	Yes
	Exclusion rules	Yes
	Validation rules	Yes
	Recommendation rules	Yes
	Replacement rules	Yes

Feature	Capability	Supported
	Product Scope: Product, Product Group, Product Family, Product Field Set	Yes
	Product Option Group scope	No
	Match in Primary Lines or Options	Yes
	Match in Location	No
	Match in Asset	No
	Match in Cart Options	Yes
	Repeat Inclusion	No
	Condition Association	Yes
	Condition Criteria	Yes
	Action Criteria	Yes
	Match in Related Lines	No
	Is Bundle Context	No
Option Configuration	Min/Max Options	Yes
	Min/Max Total Quantity	Yes
	Is Hidden	Yes
	Is Picklist	Yes
	Modifiable Type	Yes
	Option Sequencing	No
	Default / Required Option	Yes
	Inclusion criteria	No
	Min/Max Quantity	Yes
	Quantity: Default, Modifiable	Yes
	Quantity: Auto Update	No

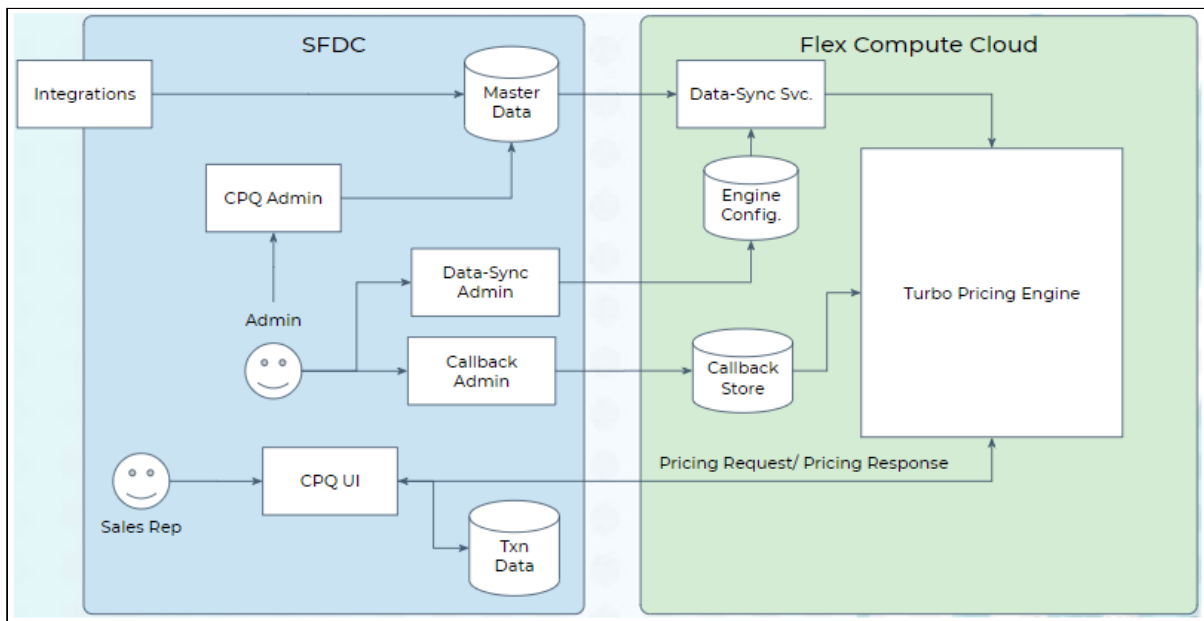
Feature	Capability	Supported
	Allow Cloning	No
	Config Type	No
Product Attributes Display	Attribute: Read-Only, Hidden, Primary	Yes
	Two column Attribute Display	Yes
	Three column Attribute Display	Yes
Product Attribute Rules	Product Scope: Product, Product Family, Product Group	Yes
	Filter Criteria	Yes
	Action Types: Allow, Default, Hidden, Disabled, Required, Reset	Yes
	Target field: Product Attribute Value, Line Item, Product, Pricelist, Product Configuration	Yes
Attribute Value Matrices	Product Scope: Product, Product Family, Product Group, Location	Yes
	Application Type: Default, Constraint, Force Set	Yes
Field Expressions / Rollup	Evaluation Context: Constraint Rule action, Record Update, Default Quantity, Rollup	Yes
	Update Product Attribute and Line Item object fields	Yes
	Rollup Group By Field: Line Item, Product Attribute	Yes

Feature	Capability	Supported
Callbacks	Option Filter Callback	No
Others	Formula field support (Condition/action)	No
	Lookups (For example, Attributes)	No
	TurboConfig Data Sync	No
	TurboPricing Integration	No
	Publisher API	Yes
	ABO Flow support	No
	Service CPQ	No
	Multi-language support	No

About TurboPricing

TurboPricing is a pricing engine built using microservices to reduce the processing time on the cart. You can enable TurboPricing to offload complex pricing computation workload from the Salesforce platform to the Apttus Flexible Compute Platform. It reduces time to submit prices to customers, improves user experience, and improves user adoption with a more responsive user interface.

The following diagram shows how data flows between a Salesforce org and Apttus Flexible Compute Platform:



Supported Features in Apttus TurboPricing

The following table lists the features supported or not supported in TurboPricing. For information on the listed features, refer to *CPQ on Salesforce Administrator Guide*.

i You cannot use TurboConfig and TurboPricing simultaneously.

Feature	Sub-Feature	Available in TurboPricing
Price Calculation		
Bundle Pricing	Price included in bundle set at PLI	Yes
	Price included in bundle set at Bundle	Yes
	Price Adjustments at Bundle	Yes
	Rollup method Flat at Bundle level	Yes
	Rollup method Per Unit at Bundle level	Yes
	Auto sequencing of options	No

	Contract pricing when same option exists in multiple bundles	No
Defaulting Quantity	Defaulting quantity for Bundles/Mutiple Charges	Yes
	Defaulting quantity FROM Product Attribute	Yes
	Default quantity derived FROM Advanced Formula	Yes
Defaulting Term	Defaulting term for Bundles/ Mutiple Charges	Yes
Price List Item		
Price Method	Use of Tiered Rates*	No
	Use of Per Unit Price method	Yes
	Use of Flat price Price method	Yes
Frequency	Use of Daily Frequency	No
	Use of Weekly Frequency	No
	Use of Frequencies Monthly, Quarterly, Half Yearly, Yearly	Yes
	Use of any Custom Frequency	No
Price Type	Use of Price Type - Included Usage	No
	Use of Price Type - One Time	Yes
	Use of Price Type - Per unit	Yes
	Use of Price Type - Usage	Yes
	Use of Price Type - Recurring	Yes
LineItem Update	Read-only quantity	Yes
	Read-only Selling term	Yes
Price Methods		
Proration	Allow Proration set on PLI	Yes

Price Method Per Unit	Use of Price Method Per Unit	Yes
Price Method Flat Price	Use of Price Method Flat price	Yes
Pricing Methods		
Min/Max Price	Min/Max Price applies to BasePrice	Yes
	Min/Max Price applies to BaseExtendedPrice	No
	Min/Max Price applies to ExtendedPrice	Yes
Price Ramps	Use of Price Ramps	No
	Use of Auto Ramp creation	No
	Use of Price ramp overlap	No
	Use of Price Escalators	No
Defer Pricing	Defer Pricing	No
Cost Models		
Cost Models	Cost Models	No
Conversions		
Currency Conversion	Use of Multi-Currencies	Yes
	Use of Dated Exchange Rates	Yes
	Disable Currency Conversion Rate for a Price List	Yes
	Currency conversion enabled at a rule see level	Yes
UOM Conversion	Use of UOM Conversions	Yes
	Use of Product specific conversion rates	Yes
	Use of Product Family-specific conversion rates	No

Price Method		
Price Method	Use of Price Method Percentage	No, Customer will have to switch to Related Price List Items
Related Pricing		
Related Pricing	Use of Related Price Lists in Pricing	No, customer will have to switch to Related Price List Items
	Source PLI set on Related Price List Item	Yes
	Source product and charge type set on Related Price List Item	Yes
	Source product group and charge type set on Related Price List Item	Yes
	Source product family and charge type set on Related Price List Item	Yes
	Source Custom Group and charge type set on Related Price List Item	Yes
Adjustments in Related Pricing	Adjustment defined on PLI	Yes
	Adjustment defined on Related Price List Item	Yes
Price Rule Set		
Header Scope and Criteria	Use of Effectivity period and Active flag	Yes
	Use of Scope Fields - price list, charge type, product family, product category, product group	Yes
	Use of Advanced Criteria	Yes

	Use of Advanced Criteria with Line Item Reference Fields	Yes
	Use of Wildcards in Advanced Criteria	No
	Application Level Bundle or Line Item	Yes
	Application Level Aggregate	No
	Use of StopProcessingMoreRules flag	Yes
Dimension based Price Rules	Use of StopProcessingMoreRules flag	Yes
	Use of Adjustment applies to - Base Price	Yes
	Use of Adjustment applies to - Base Extended Price	Yes
	Use of Adjustment applies to - Extended Price	No
Criteria based Price Rules	Use of StopProcessingMoreRules flag	Yes
	Use of Adjustment applies to - Base Price	Yes
	Use of Adjustment applies to - Base Extended Price	Yes
	Use of Adjustment applies to - Extended Price	No
	Match in Product Group	Yes
	Match in Asset	No
Price Dimension		


Use of Un-supported Price Dimension Types	Use of Un-supported Price Dimension Types - any type Except Line Item, Product Attribute and Formula Field	No
Use of Custom Price Dimension Types	Use of Custom Price Dimension Types	No, should be converted to Formula Field
Service CPQ	Service CPQ	No
Adjustments		
Manual Adjustments	Line level adjustments	Yes
	Group adjustments	Yes
	Group adjustment spread	No
	Line-level adjustment of usage price tiers	Yes
	Usage Tier Modifiable	Yes
	Misc Charge Types	Yes
Adjustments	Adjustment Bucketing	No
	Ability to create multiple Adjustments	No
	Auto refresh Usage price tiers	No
Bundle/Option level manual Adjustments	Line-level adjustments	Yes
Bundle/Option level Adjustments	Group adjustments	Yes
Asset-Based Pricing		
Assets	Assets	No
Incentives		
Other application types except Promotion	Use of Price Program, Loyalty, Rebate, Milestone Incentive or Custom	No
Promotions applied on Line Item and Summary Group	Promotions applied on Line Item and Summary Group	Yes

Promotions applied on other items	Promotions applied on other items except Line Item and Summary Group	No
Support for Promotion type - Own every X Get Y,Support for Promotion type - Own every X Get Y	Support for Promotion type - Own every X Get Y	No
Support for other Promotion types - except Own every X Get Y,Support for Promotion types - except Own every X Get Y	Support for other Promotion types - except Own every X Get Y	Yes
Incentive Limits	Support for Promotion Limits	Yes
Incentive Coupons	Support for Coupon Limits	No
Sales Promotions	Sales Promotions	No
Advanced Criteria set in Price Ruleset	Support for Incentive Criteria on Price Ruleset	Yes
Advanced Criteria with Reference Fields on Price Ruleset	Advanced Criteria with Reference Fields on Price Ruleset	No
Quotes		
Quote collaboration	Quote collaboration	No
Carts		
Favorite Configurations	Use of Favorite Configurations	No
Smart Carts	Use of Smart Carts	No
Submit for Approval	Submit for Approval	Yes
Copy	Copy products	No
Cart Line Item		
Revalidate	Revalidate	No
Totaling and Summary Groups		
Adhoc Totaling	Adhoc Totaling	No

Deal Guidance		
Deal Guidance	Deal Guidance	No
Callbacks		
Callbacks	Pricing Callback	Yes
	Validation Callback	No
	Cart Approval Callback	No
	Advanced Approval Callback	No
	Loyalty Cycle Callback	No
	Bulk Loyalty Point Callback Class	No
	Adjustment Spread Callback	No
	Loyalty Point Callback	No
	Related Pricing Callback	No
	Pricing Extension Callback	No
User Experience		
Bundle-specific option line item update	Read-only quantity	Yes
	Read-only selling term	Yes
LineItem Update	Line level adjustments	Yes
	Group adjustments	Yes

Configuring Apttus TurboEngines

The topics in this section provide information and step-by-step tasks for enabling TurboEngines for your organization.


 Please start with the [Enabling TurboEngines in an Org](#) topic and refer to the necessary steps you must take before and after the TurboEngines provisioning process.

- [Enabling TurboEngines in an Org](#)
- [Creating a Connected App](#)
- [Preparing Tenant Information](#)
- [Post-Provisioning Tasks \(TurboConfig\)](#)
- [Post-Provisioning Tasks \(TurboPricing\)](#)

Enabling TurboEngines in an Org

This topic provides a summary of the necessary steps for enabling TurboEngines (TurboConfig and TurboPricing) for your org.

An administrator can be any of the following persona: Customer Administrator, Partner Administrator, any other administrators assigned the responsibility of enabling TurboEngines for their org. In the table in this topic, this persona is referred to as the *Tenant Admin*.


 You cannot use TurboConfig and TurboPricing simultaneously.

Prerequisites

- Check the "Supported Features" topics (under [About TurboEngines](#)) for the service you want to enable. Make sure all of the features you want are included before making a provisioning request.
- You must have the appropriate TurboEngines license before turning on your org. If you do not have a license, please reach out to your Apttus Account Executive.
- You must have the Summer 2020 build of Apttus Configuration & Pricing (Apttus CPQ) in the Salesforce org to enable TurboConfig and TurboPricing. Refer to "Packages" in *CPQ on Salesforce Summer 2020 Release Notes*.

Enabling TurboEngines

To enable TurboEngines, perform the following steps for each org:

Step	Task	Owner	Description
Pre-Provisioning Tasks			
1	Set up Connected App in your org	Tenant Admin	Create a connected app to provide authentication and authorization to TurboConfig and TurboPricing Data Sync Service.
2	Prepare pre-provisioning tenant information	Tenant Admin	<p>Gather all required information for provisioning your TurboConfig or TurboPricing org.</p> <p>Provide this information to Apttus Technical Support to begin the provisioning process.</p>
Post-Provisioning Tasks			
<div>  Perform the following steps only after receiving a notice from Apttus Technical Support that the requested orgs are provisioned. You must have the new service URLs to proceed. </div>			


Step	Task	Owner	Description
3	Set up Remote Site Settings (TurboConfig)	Tenant Admin	Use the service URL you received from Apttus Technical Support to set up the remote site settings for TurboConfig.
4	Configure Services	Tenant Admin	<p>For TurboConfig, update the following settings:</p> <ol style="list-style-type: none"> 1. Set up the Config Execution Mode 2. Set up the custom flow and Configure Products button <p>For TurboPricing, update the following settings:</p> <ol style="list-style-type: none"> 1. Set up the Pricing Execution Mode 2. Set up the TurboPricing endpoint URL
6	Configure data sync settings for TurboConfig	Tenant Admin	Complete data sync post-provisioning.

Step	Task	Owner	Description
7	Configure data sync settings for TurboPricing	Tenant Admin	Complete data sync post-provisioning.
8	Sync data to TurboEngines	TurboEngines Administrator (can be Tenant Admin)	<p>TurboPricing: Set up and schedule or activate data sync to sync pricing master data.</p> <p>TurboConfig: Publish Products using REST API: TurboConfig Publisher.</p>

Creating a Connected App

As part of the pre-provisioning process you must configure a Connected App in your org to provide authentication and authorization for the following TurboEngines services:

- TurboConfig
- TurboPricing Data Sync Service

 The example in the following tasks is provided for TurboConfig but is the same process for any service configuration.

To create a Connected App

1. Navigate to **Setup > App Setup > Create > Apps**.
2. Scroll down and search for the **Connected Apps** related list and click **New** to create a new app.
3. Fill in the following details in the **Basic Information** section.

Field	Description
Connected App Name	Enter the name of the Connect App.
API Name	The API name is generated automatically based on the name of the Connected App.
Contact Email	Enter the email address of the administrator managing the Connected App.

4. Fill in the following details in the **API (Enable OAuth Settings)** section.

Fields	Description
Enable OAuth Settings	Select this to define the OAuth settings. For example, <i>TurboConfig</i> . When you enable this field, additional settings are displayed under API (Enable OAuth Settings) section.
Enable for Device Flow	Select this to enable the connected app for an external application.

Fields	Description
Callback URL	Callback URL is generated automatically when you select the field Enable for Device Flow . For example, <i>https://test.salesforce.com/services/oauth2/success</i> is generated based on the instance URL. You can also add other URLs in separate lines.
Selected OAuth Scope	Select all the entries under Available OAuth Scopes and move them to Selected OAuth Scopes by clicking the Add arrow.
Require Secret for Web Server Flow	Select this to require the connected app to provide a consumer secret for authorization.

5. You must leave all other fields blank. Click **Save**.


To capture Consumer Key and Consumer Secret

After you create a Connected App, CPQ generates **Consumer Key** and **Consumer Secret**. You must provide the values of **Consumer Key** and **Consumer Secret** to Apttus Technical Support.


1. Navigate to **Setup > App Setup > Create > Apps**.
2. Scroll down and search for the **Connected Apps** related list.
3. Click the name of the Connected App you created in the previous topic.
4. Click **Copy** next to **Consumer Key**.
5. Click **Click to reveal** next to **Consumer Secret**. After the value of the field is displayed, click **Copy**.
6. Store the information for the next part of the process.

Preparing Tenant Information

Your provisioning request for TurboConfig or TurboPricing must include specific information related to your tenant. Before your org can be provisioned, you must gather the required information and provide it to Apttus Technical Support. What information must be collected will differ depending on the service you are provisioning.

 Configure a Connected App to use with TurboEngines before collecting the information described in this topic.

Refer to the following table for all required pre-provisioning information:

 While implementing TurboPricing, any Id (15-character Id) that is returned from Salesforce is converted to an 18-character Id for the proper functioning of TurboPricing.

Configuration	Required for Service	Description
OrgId	TurboConfig, TurboPricing	<p>This is the Salesforce Organization ID of the org to be provisioned for the TurboEngine service. To locate your Organization ID:</p> <ol style="list-style-type: none"> 1. Log in into the org to be provisioned. 2. Go to Setup > Company Profile > Company Information > Salesforce.com Organization ID. 3. Copy the 15-character ID (to be converted into 18 characters). You can add any random characters to the Org ID for conversion. <p>For example, TenantId = <i>00d3i000000qn7xAAA</i></p>
Tier	TurboConfig, TurboPricing	The Tier (Gold, Silver, and Bronze) to be provisioned. If no tier is provided, then Bronze is selected by default.
Org Type	TurboConfig, TurboPricing	Org type to be provisioned (sandbox or production)
Tenant Name	TurboPricing	The one-word tenant name used for the tenant endpoint (for example, <i>customername-sandbox</i>)
Consumer Key	TurboConfig, TurboPricing	The consumer key (<i>client-id</i> in OAuth 2.0) generated from your Connected App. Refer to Creating a Connected App .
Consumer Secret	TurboConfig, TurboPricing	The secret key (<i>client-secret</i> in OAuth 2.0) generated from your Connected App. Refer to Creating a Connected App
Salesforce User Name	TurboConfig, TurboPricing	Admin username for the org to be provisioned with read/write access to Apttus CPQ (used by Apttus Technical Support for verifying settings)

Configuration	Required for Service	Description
Salesforce Password	TurboConfig, TurboPricing	Password for the Salesforce admin user
Authority	TurboPricing	The URL used to verify session Id for TurboPricing (<i>login.salesforce.com</i> , <i>test.salesforce.com</i> , or a custom Salesforce domain)
InstanceURL	TurboPricing	The URL given by the UI after logging into the org to be provisioned (for example, <i>customerturbo.my.salesforce.com</i>)
OAuthTokenURL	TurboConfig	Salesforce token endpoint URL (this will be <i>login.salesforce.com</i> or <i>test.salesforce.com</i> , depending on your Org Type)

After collecting all required information, provide it with your tenant provisioning request to Apttus Technical Support.

Post-Provisioning Tasks (TurboConfig)

The post-provisioning process for TurboConfig are divided into two main tasks:

- Configuring the service (custom settings and properties)
- Sync product data to TurboConfig (using Publisher APIs)

Refer to the following topics for step-by-step instructions to complete setup and configuration of TurboConfig:



Configuring Remote Site Settings for TurboConfig

To create Remote Site record

1. Go to **Setup > Administration Setup > Security Controls > Remote Site Settings**.
2. Click **New Remote Site**.
3. Enter the name in **Remote Site Name**. For example, *TurboConfig*.
4. Enter the URL for the remote site in **Remote Site URL**.

 Contact Apttus Technical Support for the Remote Site URL.

5. Enable **Active**, if not selected by default.
6. Click **Save**.

Configuring TurboConfig Settings

You can enable TurboConfig either globally or for select CPQ flows. By default, the global **Constraint Rule Execution Mode** is set to *Client*. To enable the TurboConfig for select flows, refer to [Configuring Custom Flows for TurboConfig](#). Follow the steps below to enable TurboConfig at the global level.

Take note of the following before you enable TurboConfig:


- Refer to [Supported Features and Capabilities in Apttus TurboConfig](#) topic to confirm that the features you use are supported in TurboConfig before enabling it globally.
- If you want to enable TurboConfig for certain types of quotes only, create a new flow with execution mode as *CMS*. Refer to [Configuring Custom Flows for TurboConfig](#)
- Once you create a quote using *CMS* execution mode you cannot switch to *Client* mode for that particular quote.
- The constraint rule execution mode (*Client* or *CMS*) is determined at the beginning of the quote. Once determined you cannot change them.
- The constraint rule execution mode *CMS* works on standalone products but the products must be published individually or in groups.
- All formula fields in constraint rule condition criteria and action criteria referring to line item object or any standard object are not supported in *CMS* execution mode. However, custom fields on line item objects are supported.
- Apttus CPQ rollups are not supported in *CMS* execution mode.
- Salesforce lookup fields are supported in *CMS* execution mode.

To enable TurboConfig on the quote

1. Go to **Setup > App Setup > Develop > Custom Settings**.
2. Click **Config System Properties**. Click **Manage**.
3. Click **Edit** next to **System Properties**.
4. Define the fields as explained in the table below:

Field	Description
Constraint Rule Execution Mode	Enter the value <i>CMS</i> in the field.
CMS End Point URL	Enter the End Point URL for TurboConfig.

Field	Description
CMS Publisher Endpoint URL	Enter the Publisher Endpoint URL for TurboConfig.

 CMS End Point URL and CMS Publisher Endpoint URL are provided by Apttus Technical Support.

5. Click **Save**.

Configuring Custom Flows for TurboConfig

You can enable TurboConfig for selective CPQ flows. You can use this functionality to avoid making TurboConfig as the default configuration engine and use the engine to process large and complex configuration rules. To enable TurboConfig for a specific flow, you must create a dedicated data set of **Config System Properties**.

To enable TurboConfig for specific flows

1. Create a custom flow. Refer to "Configuring Flow Settings" in *CPQ on Salesforce Administrator Guide*.
2. Create a formula action at the **Quote/Proposal** object for the flow you created in Step 1. Refer to "Creating Custom Buttons for Different Flows" in *CPQ on Salesforce Administrator Guide*.
3. Go to **Setup > App Setup > Develop > Custom Settings**.
4. Click **Config System Properties**. Click **Manage**.
5. Click **New** to create a new data set.
6. In the **Name** field, enter the name of the custom flow you created in Step 1.
7. Define **Constraint Rule Execution Mode**, **CMS End Point URL**, and **CMS Publisher Endpoint URL** as described in Step 4 in the topic [Configuring TurboConfig Settings](#).
8. Click **Save**.

The Sales rep must use the custom flow that you created to configure the quote using TurboConfig.

Syncing TurboConfig Data

You must publish the products from Salesforce to TurboConfig to be able to start using products in TurboConfig. You must publish the products again if you make changes to the product structure or configuration rules. If you update the rules, you must publish the

products that are impacted by the rules. The Sales rep cannot use the products that are not published, CPQ displays an error if the Sales rep adds an unpublished product to the cart.

You can use TurboConfig Publisher REST APIs to publish the products to TurboConfig. There are two ways to publish the products using the APIs: as **Standalone Products and Bundles**, or by **Category**.

For detailed instructions for using the API along with endpoints and request and response payloads refer to the *Apttus TurboEngines Summer 2020 REST API Guide*.

Post-Provisioning Tasks (TurboPricing)

The post-provisioning process for TurboPricing are divided into four main tasks:


1. Configuring the service (custom settings and properties)
2. Configuring data sync service for use by administrators
3. Configure custom pricing callbacks
4. Setting up and activating data sync for pricing master data from SFDC to TurboPricing

Refer to the following topics for step-by-step instructions to complete setup and configuration of TurboPricing:



Setting Up the TurboPricing Endpoint URL

This section provides information for setting up the TurboPricing endpoint URL in the org.

1. Click the **All Tabs** icon () and click **Admin**. The Home page is displayed.
2. Click **New**. The New Admin page is displayed.
3. In the **Name** field, enter *APTS_PricingServiceOverrideURL*.
4. In the **Value** field, enter the TurboPricing endpoint URL (without https://).
5. Click **Save**.

Setting Up the Pricing Execution Mode

This section provides information for setting up the Pricing Execution Mode in the org.

1. Go to **Setup > App Setup > Develop > Events > Custom Settings**.
2. Click **Config System Properties**. Click **Manage**.

3. Click **Edit** next to System Properties.
4. In the **Pricing Execution Mode**, enter *Turbo*.
5. Click **Save**.

Configuring Data Sync for TurboPricing

To complete post-provisioning for TurboPricing, the tenant admin must configure settings for data sync services. TurboEngines data sync provides a high-performance mechanism to sync pricing master data at regular, scheduled intervals (or on-demand) between Apttus CPQ on Salesforce and TurboPricing. Before the initial data sync, you must configure settings enable data sync services and give the administrator access to the TurboEngines Data Sync Admin user interface (UI) to set up and schedule or activate the sync.

Perform the following tasks to complete post-provisioning data sync tasks for TurboPricing.

- [Configure Data Sync Specific Settings](#)
 - [To configure the service URL](#)
 - [To configure the CSD Trusted Site](#)
- [Configure Permissions for Data Sync Admin User](#)
 - [To provide access to the data sync app](#)
 - [To make all tabs visible in the data sync app](#)
- [Synchronizing User Time Zone Settings](#)
- [Setting up and Syncing TurboPricing Data](#)

Configure Data Sync Specific Settings

You must configure the data sync service URL and a CSP Trusted Site entry so SFDC can communicate with an external server.

To configure the service URL

1. Go to **Setup > App Setup > Develop > Custom Settings**.
2. Click **Manage** next to **Turbo Engine Admin Properties**.
3. Click **New**.
4. Enter the following required properties:
 - **Name:** LightsaberServiceUrl
 - **Turbo Engine service Endpoint:** Endpoint URL provided by Apttus Tech Support
5. Click **Save**.

To configure the CSD Trusted Site

1. Go to **Setup > Administration Setup > Security Controls > CSP Trusted Sites**.
2. Click **New Trusted Site**.
3. Enter the following required properties:
 - **Trusted Site Name**: Enter a name for the trusted site (for example, "TurboEngineAdminService")
 - **Trusted Site URL**: Enter the full URL. This is the service URL you configured in [Configuring the Service URL](#).
 - **Context**: Select a context.
4. Click **Save**.

Configure Permissions for Data Sync Admin User

Users who need to configure and run Turbo Engine Data Sync must have permission to access and use the Data Sync Admin UI. This can be a user assigned to the System Administrator profile, or you can customize a profile and create one or more users in this role.

To check if the current user has the right permissions:

1. Log in to your organization as the admin user.
2. Open the Salesforce App Launcher (Lightning) and launch the **Turbo Engine Admin** app.
3. If the **Data Integration** and **Callbacks** tabs are visible after launching the app, the user has the correct permissions. Otherwise, log back in as a system administrator and perform the following tasks to provide access to the user profile.

To provide access to the data sync app

1. Go to **Setup > App Manager**.
2. Find the **Turbo Engine Admin** app in the list. Click the drop-down at the end of the row and select **Edit**.
3. Click **User Profiles**.
4. From the list of Available Profiles, search and select the app you want to add.
5. Click the right-facing arrow to move the profile from the list of Available Profiles to the list of Selected Profiles.
6. Click **Save**.

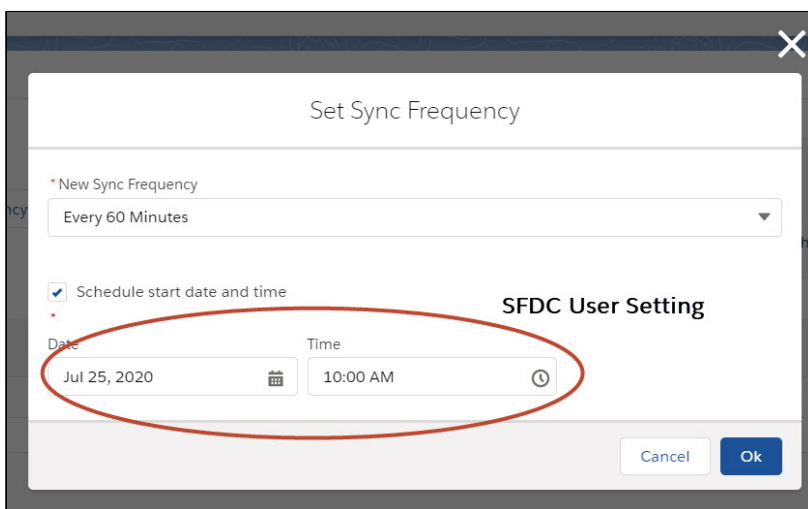
To make all tabs visible in the data sync app

1. Go to **Setup > Profiles**.
2. Search for the profile you want to configure and click **Edit**.
3. Under Custom Settings, make sure the following tabs are set as "Default On":
 - **Data Integration**: This tab serves as the starting point for managing all consumer profiles.
 - **Callbacks**: This tab allows you to manage pricing callbacks for Turbo Pricing.
 - **Consumer Profile**: This tab allows you to set up and configure data sync operations.
 - **Run Details**: This tab allows you to review run history for data sync and take action.

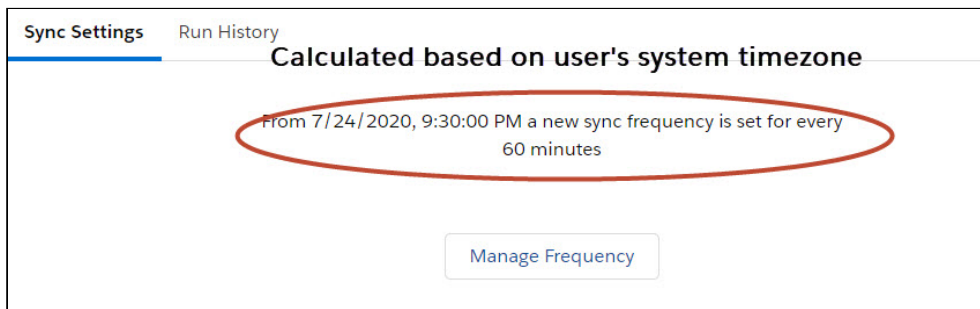
Synchronizing User Time Zone Settings

It is recommended to configure the admin user's time zone settings in Salesforce to match the system settings on the device they are using. This helps avoid confusion when managing frequency from Turbo Engine Admin Consumer Profiles.

Due to Salesforce limitations, the date picker for managing data sync frequency always displays the date and time based on the currently logged-in user's *timezone settings*.



However, the scheduled data and time displayed by the Data Sync Admin UI is calculated based on the users *system timezone*.



To avoid confusion, configure the admin user's Time Zone settings to match the user's system settings. To configure the user's Time Zone settings, go to **User Settings > Language & Time Zone**. When they are synchronized, the next scheduled sync date and time is displayed to match the user's system time.

Setting up and Syncing TurboPricing Data

For complete information and the tasks required to administer TurboEngines data sync for TurboPricing, refer to *TurboEngines Data Sync Summer 2020 Administrator Guide* on the Apttus Documentation portal.

Configuring TurboPricing Callbacks

This topic provides information on configuring TurboPricing callbacks.

Callbacks provide you with a mechanism to apply a custom logic on different components of CPQ such as line items, Cart page, Catalog page at run-time. For example, you can apply custom pricing on the line items in the cart using the Pricing Callback Class. Callbacks are implemented using interfaces that are specific to each callback. These interfaces have various methods that you can use to achieve your task. You must implement the interface in a .NET class and within that class, you must configure your custom logic using the methods of the interface.

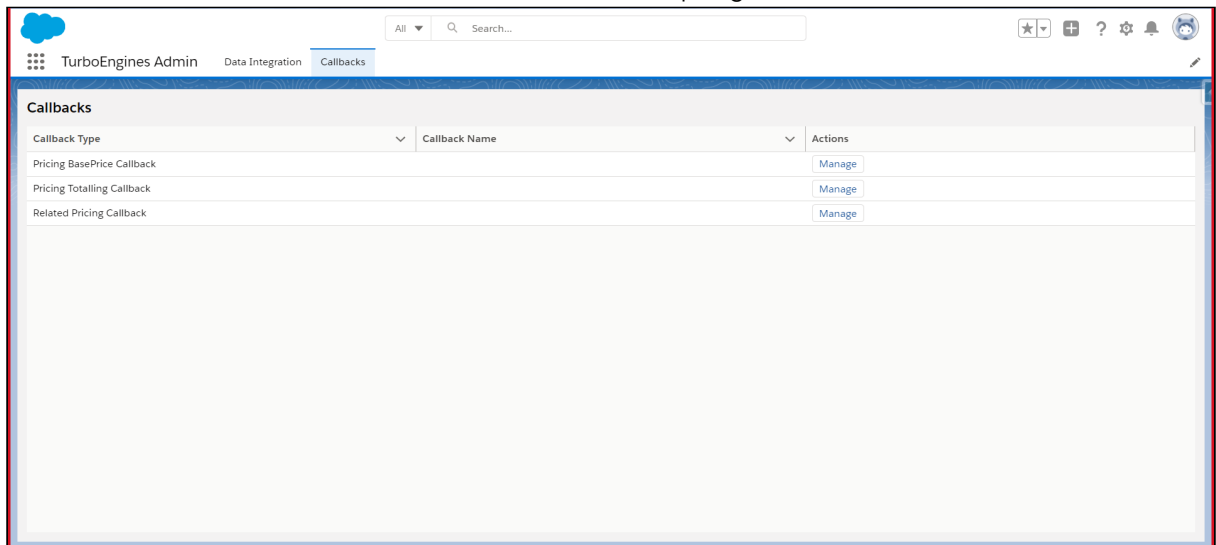
The sections in this topic provide information for:

- [Navigating the TurboPricing Callback Administrator User Interface](#)
- [Managing TurboPricing Callbacks](#)
- [Validation Callback class](#)
- [Helper Functions for TurboPricing Callbacks](#)
- [Pricing Callback Class for TurboPricing](#)

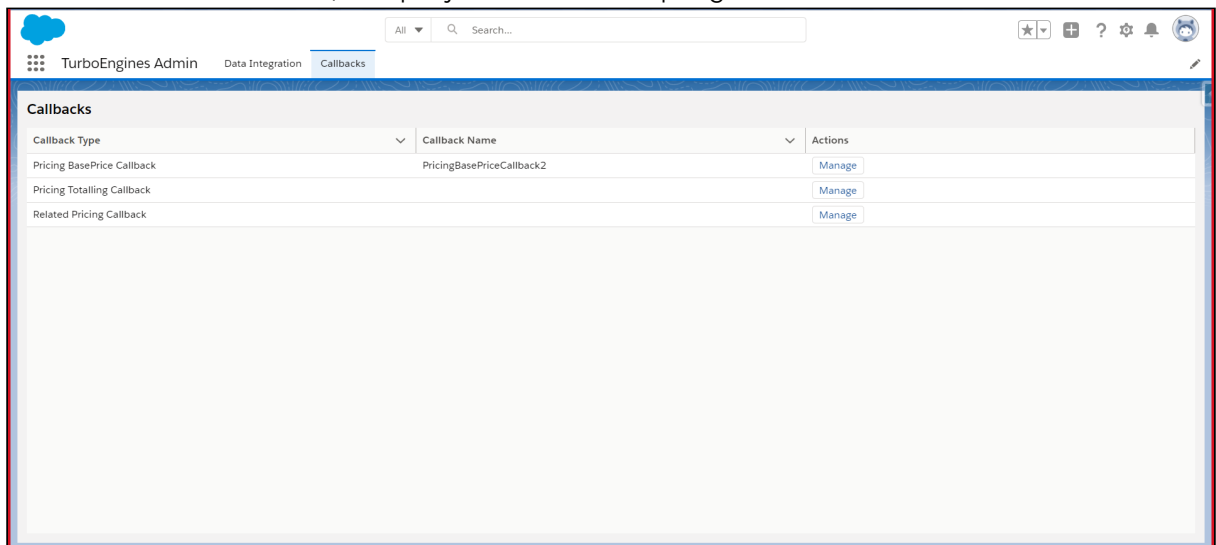
Navigating the TurboPricing Callback Administrator User Interface

This section provides information on navigating the TurboPricing callback administrator user interface.

1. Log in to the Salesforce org.
2. Click **Switch to Lightning Experience**.
3. Click the **Callbacks** tab. A list of callbacks is displayed.

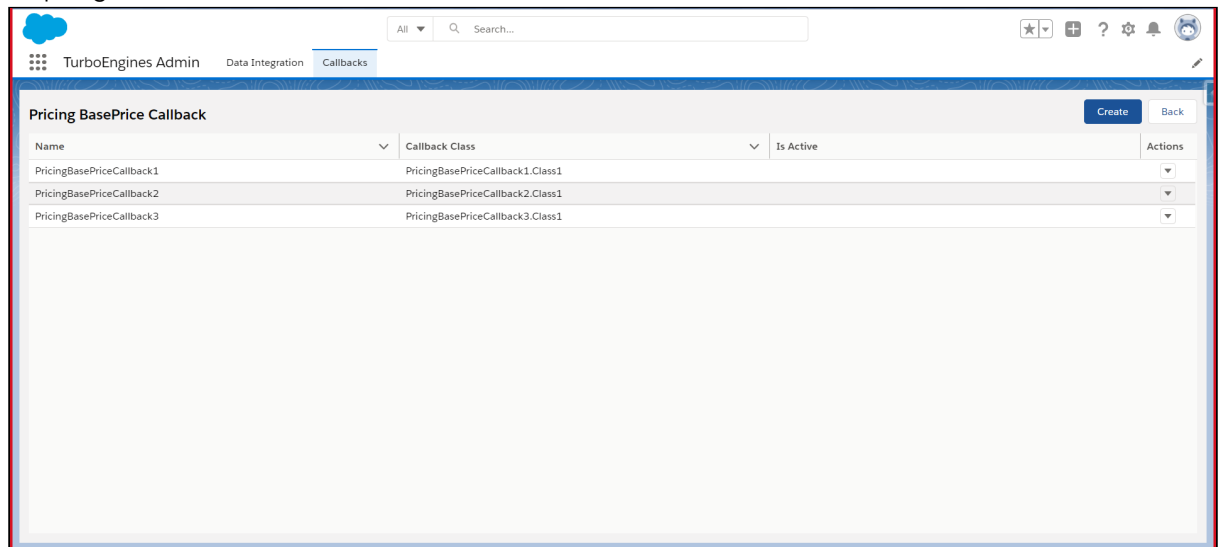


If a callback is enabled, the project name is displayed in the **Callback Name** column.

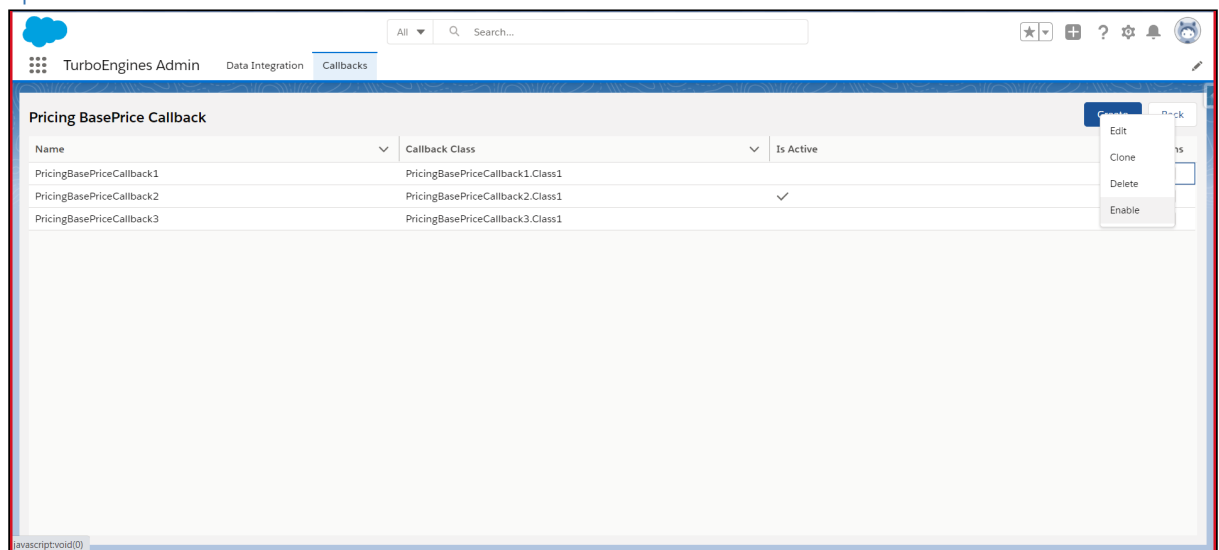


4. Click **Manage** under the **Actions** column for the required callback to configure it. The detail page of the selected callback with a list of projects related to the callback is




displayed.





- Click **Create** to [create a new project](#).
- Click the **Action** icon (▼) for the required callback and [select one of the following options](#):




- Edit
 - Clone
 - Enable
 - Disable
 - Delete
- From the Explorer panel, create a file under the current project if required. See [Managing Files and Folders in the Explorer Panel of the Edit Project Page](#).
 - Test your code before saving it. See [Testing a Callback Method by Executing the Code](#).

9. Click the **Code Difference** icon () to see the difference between the original and modified code. The differences are highlighted for easy identification.
10. At the bottom of the code editor panel:
 - Click the **Collapse** icon () to show the panel and **Expand** icon () to hide the panel.
 - **Output**: This tab displays the output of your code. Whenever you execute a method, the returned result is displayed in this tab.
 - **Input**: This tab displays the input of your code. You can verify what values CPQ has set for parameters or what values CPQ retrieves by reference for a parameter, when you execute some code.
 - **Profiler**: This tab displays the order of execution of methods and performance of methods. You can also check how long CPQ takes to execute each method.
 - **Console**: This tab shows how you can access custom tables, which you need to access in a callback class. It shows the output of an API (external and internal). This tab shows the logs that you write in a method. This tab also helps you debug errors.


Managing Files and Folders in the Explorer Panel of the Edit Project Page

This section describes how you can manage files and folders in the Explorer panel of the edit project screen. You can click the Collapse icon () to show the panel and Expand icon () to hide the panel.


Adding an Item

1. Select a folder if any and click the **Add Item** icon (). The Add Item pop-up is displayed.
2. From the Type drop-down, select what type of item you want to add. The supported values are File and Folder.
3. Enter a name for the item in the **Name** field.
4. Click **Create**.


Renaming an Item

1. Select a file to be renamed.
2. Click **Rename Item** icon (). The Rename Item pop-up is displayed.
3. In the **Name** field, enter a new name and click **Save**.



Deleting an Item



1. Select a file to be removed.
2. Click **Delete Item** icon (). The Delete Item pop-up prompting you to confirm deletion is displayed.
3. Click **Yes** to delete the item.



If there is only one item, the Delete Item icon () is disabled. You must have at least one item in the edit project page.

Testing a Callback Method by Executing the Code

CPQ allows you to test the code you have written for a callback method before saving it. In the **Test Run** panel, click the **Collapse** icon () to show the panel and **Expand** icon () to hide the panel.

1. From the **Class** drop-down, select a class. It displays all classes available in the project callback.
2. From the **Method** drop-down, select a method. It displays all methods currently available in the class. You can execute a method of that particular class. CPQ does not display private methods on the **Method** drop-down.
3. In the **Parameters** field, enter the code.
4. Click the **Refresh project metadata** icon () to refresh project data. For example, if you have an unsaved method and want to execute it for validation, click
5. the **Refresh project metadata** icon. The new method is listed in the **Method** drop-down.
6. Click the **Execute** icon () to execute the method.

Managing TurboPricing Callbacks

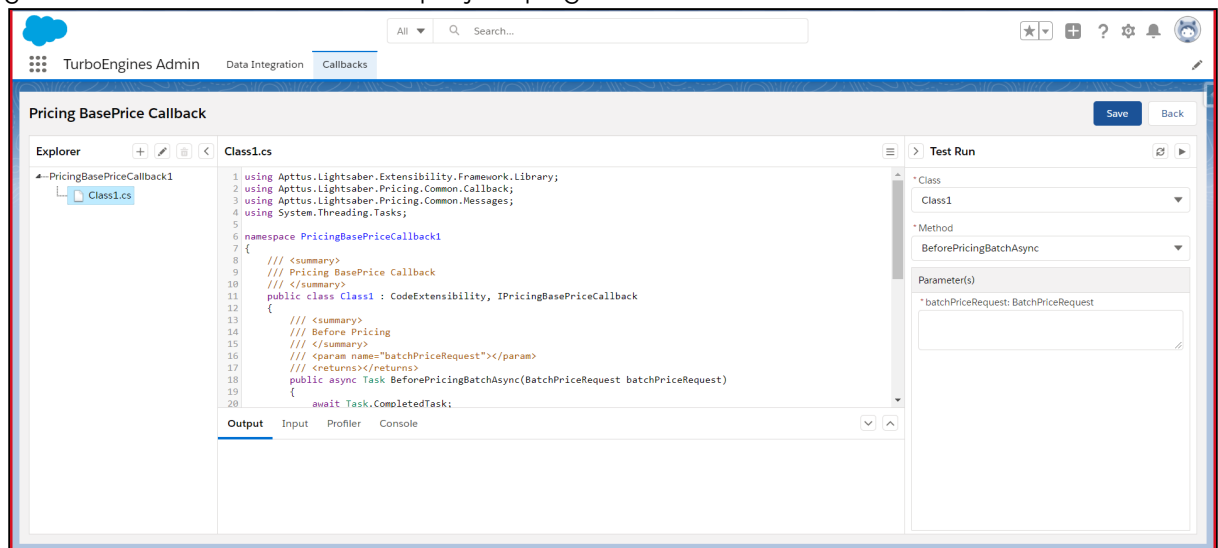
This section provides information on writing or creating a TurboPricing callback and managing an existing TurboPricing callback.

- [Creating a TurboPricing Callback](#)
- [Editing a TurboPricing Callback](#)
- [Cloning a TurboPricing Callback](#)
- [Enabling a TurboPricing Callback](#)

- [Disabling a TurboPricing Callback](#)
- [Deleting a TurboPricing Callback](#)

Creating a TurboPricing Callback

1. Log in to the Salesforce org.
2. Click **Switch to Lightning Experience**.
3. Click the **Callbacks** tab. A list of callbacks are displayed. If a callback is enabled, the project name is displayed in the **Callback Name** column.
4. Click **Manage** under the **Actions** column for the required callback to configure it. The detail page of the selected callback with a list of projects related to the callback is displayed.
5. Click **Create** to create a new project. The New Callback pop-up is displayed.
6. Enter a name for the callback project and click **Create**. A new project is created and you are redirected to the edit project page.



There is a sample template for you to create a callback.

```

using Apttus.Lightsaber.Extensibility.Framework.Library;
using Apttus.Lightsaber.Pricing.Common.Callback;
using Apttus.Lightsaber.Pricing.Common.Messages;
using System.Threading.Tasks;

namespace NewPricingBasePriceCallback
{
    /// <summary>
    /// Pricing BasePrice Callback
    /// </summary>
    public class Class1 : CodeExtensibility, IPricingBasePriceCallback
    {
        /// <summary>
        /// Before Pricing
        /// </summary>
        /// <param name="batchPriceRequest"></param>
        /// <returns></returns>
        public async Task BeforePricingBatchAsync(BatchPriceRequest
batchPriceRequest)
        {
            await Task.CompletedTask;
        }

        /// <summary>
        /// On Pricing
        /// </summary>
        /// <param name="batchPriceRequest"></param>
        /// <returns></returns>
        public async Task OnPricingBatchAsync(BatchPriceRequest
batchPriceRequest)
        {
            await Task.CompletedTask;
        }

        /// <summary>
        /// After Pricing
        /// </summary>
        /// <param name="batchPriceRequest"></param>
        /// <returns></returns>
        public async Task AfterPricingBatchAsync(BatchPriceRequest
batchPriceRequest)

```




```

    {
        await Task.CompletedTask;
    }
}


```

7. Create a callback using the template.
8. Click **Save**. If there are no errors, the code is saved successfully. If there is any error in the code, an error message is displayed. At the bottom of the editor panel, CPQ displays the reason for the error.


Editing a TurboPricing Callback

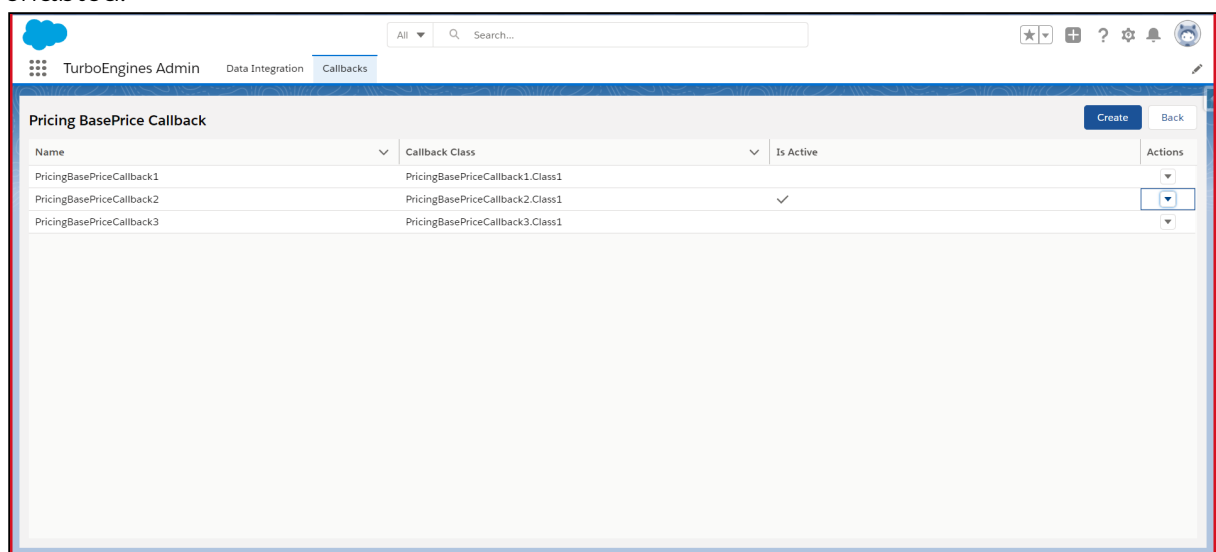
1. Log in to the Salesforce org.
2. Click **Switch to Lightning Experience**.
3. Click the **Callbacks** tab. A list of callbacks are displayed. If a callback is enabled, the project name is displayed in the **Callback Name** column.
4. Click **Manage** under the **Actions** column for the required callback to configure it. The detail page of the selected callback with a list of projects related to the callback is displayed.
5. Click the **Action** icon () for the required callback and select **Edit**. The edit project page is displayed.
6. Perform the required edits to the project. See [Creating a TurboPricing Callback](#).
7. Click **Save**.


Cloning a TurboPricing Callback

1. Log in to the Salesforce org.
2. Click **Switch to Lightning Experience**.
3. Click the **Callbacks** tab. A list of callbacks are displayed. If a callback is enabled, the project name is displayed in the **Callback Name** column.
4. Click **Manage** under the **Actions** column for the required callback to configure it. The detail page of the selected callback with a list of projects related to the callback is displayed.
5. Click the **Action** icon () for the required callback and select **Clone**. The New Callback pop-up is displayed. The name of the callback is <original_callback_name>Clone.
6. Enter a new name if you want.
7. Click **Clone**.

Enabling a TurboPricing Callback


1. Log in to the Salesforce org.
2. Click **Switch to Lightning Experience**.
3. Click the **Callbacks** tab. A list of callbacks are displayed. If a callback is enabled, the project name is displayed in the **Callback Name** column.
4. Click **Manage** under the **Actions** column for the required callback to configure it. The detail page of the selected callback with a list of projects related to the callback is displayed.
5. Click the **Action** icon () for the required callback and select **Enable**. The project is enabled.




 When you enable a project, CPQ directly impacts the processor services because of real-time update.

Disabling a TurboPricing Callback

1. Log in to the Salesforce org.
2. Click **Switch to Lightning Experience**.
3. Click the **Callbacks** tab. A list of callbacks are displayed. If a callback is enabled, the project name is displayed in the **Callback Name** column.
4. Click **Manage** under the **Actions** column for the required callback to configure it. The detail page of the selected callback with a list of projects related to the callback is displayed.

- Click the Action icon () for the required callback and select **Disable**. The project is disabled.

Deleting a TurboPricing Callback

- Log in to the Salesforce org.
- Click **Switch to Lightning Experience**.
- Click the **Callbacks** tab. A list of callbacks are displayed. If a callback is enabled, the project name is displayed in the **Callback Name** column.
- Click **Manage** under the **Actions** column for the required callback to configure it. The detail page of the selected callback with a list of projects related to the callback is displayed.
- Click the **Action** icon () for the required callback and select **Delete**. The Delete Callback pop-up prompting you to confirm deletion is displayed. Click **Yes** to delete the callback.

 You cannot delete an enabled project. First you need to disable the project and then you can delete it.

Validation Callback class

The *Validation Callback Class* provides you a mechanism to define a custom logic to be executed after the pricing. The *Validation Callback Class* is executed when the user performs after Pricing, Finalize Cart, Reprice, and Submit for Approval. Thus, you can not perform the Pricing, Finalize Cart, Reprice, and Submit for Approval actions, if the validation fails. For more information on Validation Callbacks, refer to the [Validation Callback Class](#) topic in the *CPQ on Salesforce Summer 2020 Administrator Guide*.

Helper Functions for TurboPricing Callbacks

This section describes the Helper functions available for TurboPricing callbacks.



CacheHelper

The CacheHelper function helps you cache data for faster retrievals. The *ICacheHelper* instance, which is returned when *GetCacheHelper()* is called, provides you with the following methods:

Methods

```
void Set<T>(string key, T value, string regionName);  
T Get<T>(string key, string regionName);  
bool Contains(string key, string regionName);
```

Example

```
public class Product2  
{  
    public string Id {get; set;}  
    public string Name { get; set;}  
}
```

Set Data in Cache

```
var cacheHelper = GetCacheHelper();  
var dbHelper = GetDBHelper();  
  
Product2 product = new Product2()  
{  
    Id= "RT001", Name = "Router"  
};  
await cacheHelper.Set<Product2>("Product",product);  
var productFromCache = await cacheHelper.Get<Product2>("Product");  
logHelper.LogDebug(productFromCache);
```

Console Output

```
{"Id":"RT001","Name":"Router"}
```

Get Data from Cache

```
var cacheHelper = GetCacheHelper();
Product2 product = new Product2()
{
    Id= "RT001", Name = "Router"
};
await cacheHelper.Set<Product2>("Product",product);

var productFromCache = await cacheHelper.Get<Product2>("Product");
logHelper.LogDebug(productFromCache);
```

Console Output

```
{"Id":"RT001","Name":"Router"}
```

DBHelper

The DBHelper function helps you fetch the required documents from the database based on *Query* and *FilterConditions*, and fields that you require from the documents that fulfill the conditions. You can get the instance of *IDBHelper* by calling the method *GetDBHelper()* and it provides you with the following methods:

Methods

```
Task<List<T>> FindAsync<T>(Query query);
Task<List<T>> FindAsync<T>(string entityName, Expression<Func<T, bool>>
filterCondition, int? limit, params string[] fields);
Task<List<T>> FindAsync<T>(string entityName, Expression<Func<T, bool>>
filterCondition, params string[] fields);
DBStatisticsInfo GetDBStatistics();
```

The first method helps you fetch the documents based on a *Query*, which comprises of the *EntityName* (*Name of the collection*), list of *FilterConditions*, *Fields* to fetch, and *Limit* of results to retrieve.

The *GetDBStatistics()* provides you the information on the total number of queries and the time taken for each query during callback execution.

Note

GetDBStatistics() method should be used only when you are debugging or troubleshooting the issue. Once the debugging/troubleshooting is done, you must remove it from your callback code.

Definitions

```
public class Query
{
    public string EntityName { get; set; }
    public List<FilterCondition> Conditions { get; set; }
    public string[] Fields { get; set; }
    public int? Limit { get; set; }
}

public class FilterCondition
{
    public string FieldName { get; set; }
    public ConditionOperator ComparisonOperator { get; set; }
    public object Value { get; set; }
}

ComparisonOperator :
    EqualTo,
    GreaterThan,
    GreaterThanOrEqual,
    In,
    LessThan,
    LessThanOrEqual,
    NotEqualTo,
    NotIn,
    Contains
```

- *EntityName* is the name of the collection you want to fetch the records from, or the records of the entity that you want to query upon.
- *Conditions* are a list of *FilterConditions*, which compare a field or a column's value with a specified *Value*.
- *Fields* are the columns of the entity you want in your query results.
- *Limit* is the number of records you want to limit your result to.

The following are a few examples of different Query Models:

Query Example

```
Query query = new Query()
{
    EntityName = "Account",
    Fields = new string[]{ "Id", "Name", "Type" },
    Limit = 5,
    Conditions = new List<FilterCondition>()
    {
        new FilterCondition()
        {
            ComparisonOperator = ConditionOperator.EqualTo,
            FieldName = "Type", Value = "Ship-To"
        }
    }
};
```

DB Call Example

```
var dbHelper = GetDBHelper();
List<AccountQueryModel> account = await dbHelper.FindAsync<AccountQueryModel>(query);
```

Output

After you execute *FindAsync*, it retrieves data from the database and returns a list of *AccountQueryModel*.

Complex Query Example

```

Query query = new Query()
{
    EntityName = "Account",
    Fields = new string[] { "Id", "Name", "Type","BillingCity__c",
"BillingCountryCode__c" },
    Limit = 2,
    Criteria = new Expression(ExpressionOperator.AND)
};

FilterCondition nestedConditionOne = new FilterCondition()
{
    ComparisonOperator = ConditionOperator.EqualTo,
    FieldName = "Type",
    Value = "Sold-To"
};
query.Criteria.AddCondition(nestedConditionOne);

Expression complexExpression = new Expression(ExpressionOperator.OR);
FilterCondition complexExpressionConditionOne = new FilterCondition()
{
    ComparisonOperator = ConditionOperator.In,
    FieldName = "BillingCity__c",
    Value = new string[] { "BEDFORD PARK", "FREMONT" }
};
FilterCondition complexExpressionConditionTwo = new FilterCondition()
{
    ComparisonOperator = ConditionOperator.EqualTo,
    FieldName = "BillingCountryCode__c",
    Value = "US"
};
complexExpression.AddCondition(complexExpressionConditionOne);
complexExpression.AddCondition(complexExpressionConditionTwo);
query.Criteria.AddFilter(complexExpression);

```


DB Call Example

```
var dbHelper = GetDBHelper();  
List<AccountQueryModel> accountList = await  
dbHelper.FindAsync<AccountQueryModel>(query);  
return account;
```

Result

After you execute *FindAsync*, it retrieves data from the database and returns a list of *AccountQueryModel*.

Note:


For better performance, consider the following scenarios when executing database queries.

1. Apply more filters to narrow down the search results.
2. Fetch only the required columns or fields.
3. Do not execute queries inside *for* loop.

HttpHelper

The *HttpHelper* function helps you with HTTP calls covering CRUD operations.

To get an instance of the *IHttpHelper*, you must call *GetHttpHelper()*, which returns you an instance of *IHttpHelper* allowing you to call the following methods:

-  It is not recommend to call salesforce API from Custom Code, due to performance reasons.

Methods

```

HttpContent GetHttpContentFromXml<T>(T payload);
Task<T> GetAsync<T>(string requestUri);
Task<HttpResponseMessage> GetAsync(string requestUri);
Task<T> PostAsync<T>(string requestUri, HttpContent content);
Task<HttpResponseMessage> PostAsync(string requestUri, HttpContent content);
Task<T> PutAsync<T>(string requestUri, HttpContent content);
Task<HttpResponseMessage> PutAsync(string requestUri, HttpContent content);
Task<HttpResponseMessage> DeleteAsync(string requestUri);

```

Example

```

var httpHelper = GetHttpHelper();
var logHelper = GetLogHelper();

string url = "https://jsonplaceholder.typicode.com/todos/1";
var response = await httpHelper.GetAsync<SampleResponse>(url);
logHelper.LogDebug(response);

url = "https://jsonplaceholder.typicode.com/todos";
response.id = 0;
response.title = "Callback";
var contentToPost = new
    StringContent(JsonConvert.SerializeObject(response), Encoding.UTF8, "application/
    json");
var postResult = await httpHelper.PostAsync<SampleResponse>(url, contentToPost);
logHelper.LogDebug(postResult);

```

Console Output

```

Complete-GetAsync(https://jsonplaceholder.typicode.com/todos/1), HttpStatus=OK,
Time=285.534
{"userId":1,"id":1,"title":"delectus aut autem","completed":false}
Complete-PostAsync(https://jsonplaceholder.typicode.com/todos), HttpStatus=Created,
Time=400.073
{"userId":1,"id":201,"title":"Callback","completed":false}

```

LogHelper

The LogHelper function helps you with the tracing or logging. The *ILogHelper* instance returned when *GetLogHelper()* is called provides you with the following methods. Log is created in Jaeger at runtime and while authoring it creates log in-memory, which is available on the **Console** tab on the TurboPricing Callback Administrator User Interface (edit page).

Methods

```
void LogCritical(string message, Exception ex);
void LogDebug(string message);
void LogDebug(object obj);
void LogError(string message, Exception ex);
void LogInformation(string message);
void LogTrace(string message);
void LogWarning(string message);
```

Example

```
public class AccountQueryModel
{
    public string Id {get; set;}
    public string Name { get; set;}
    public string Type {get; set;}
}
```

```
var logHelper = GetLogHelper();
logHelper.LogTrace("Trace Statement");
logHelper.LogDebug("Debug Statement");
logHelper.LogWarning("Warning Statement");
logHelper.LogInformation("Info Statement");
AccountQueryModel account = new AccountQueryModel(){
    Id= "0013i0000043m9eAAA",
    Name = "Account_B5F83C"
};
logHelper.LogDebug(account);
logHelper.LogError("Error Message", new Exception("SystemException"));
```

Console Output

```
Trace Statement
Debug Statement
Warning Statement
Info Statement
{"Id":"0013i0000043m9eAAA","Name":"Account_B5F83C","Type":null}
Error Message
{
  "ClassName": "System.Exception",
  "Message": "SystemException",
  "Data": null,
  "InnerException": null,
  "HelpURL": null,
  "StackTraceString": null,
  "RemoteStackTraceString": null,
  "RemoteStackIndex": 0,
  "ExceptionMethod": null,
  "HResult": -2146233088,
  "Source": null,
  "WatsonBuckets": null
}
```

MetadataHelper

The MetadataHelper function helps you fetch the metadata for an entity for a specified RecordType. The *IMetadataHelper* instance returned when *GetMetadataHelper()* is called performs the following method:

Method

```
Task<string> GetRecordTypeIdByName(string entityName, string recordTypeName);
```

Example

```
var metadataHelper = GetMetadataHelper();
var logHelper = GetLogHelper();

var recordTypeId = await metadataHelper.GetRecordTypeIdByName("Product2", "Master");
logHelper.LogDebug(recordTypeId);
```

Console Output

```
0120000000000000AAA
```

PricingHelper

The PricingHelper function helps you perform pricing related operations such as apply rounding and updating the Price Method for line items. To get the instance of *IPricingHelper*, you must call *GetPricingHelper()*, which provides the following methods:

Methods

```
decimal? ApplyRounding(decimal? value, int precision, RoundingMode roundingMode);
void UpdatePrice(LineItemModel lineItemModel);
```

Example

```
var pricingHelper = GetPricingHelper();
var dbHelper = GetDBHelper();

var roundValue = pricingHelper.ApplyRounding(1.657m, 2, RoundingMode.DOWN);
logHelper.LogDebug(roundValue);
roundValue = pricingHelper.ApplyRounding(1.657m, 2, RoundingMode.UP);
logHelper.LogDebug(roundValue);
```

Console Output

```
1.65  
1.66
```

VaultHelper

The VaultHelper function helps you fetch the KeyVault details for an application. The *IVaultHelper* instance returned when *GetVaultHelper()* is called fetches KeyVault values using the following method:

Method

```
Task<Dictionary<string, object>> GetValue(string AppName);
```

Example

```
var vaultHelper = GetVaultHelper();  
var logHelper = GetLogHelper();  
  
var vault = await vaultHelper.GetValue("MyApp");  
logHelper.LogDebug(vault["AppKey"]);  
logHelper.LogDebug(vault["AppSecrets"]);
```

Console Output

```
"Your_Key"  
"Your_Secrets"
```

To set and update value in vault

Using the Vault API, you can set and update value in vault, which is stored in encrypted format. You cannot retrieve vault value using any API. You can retrieve vault values only using the VaultHelper function.

1. **Create:** POST: /pricing/api/admin/Vault

It creates an application with the specified key value name and application name.

Body:

```
{
  "AppName": "MyApp",
  "Secrets": {
    "MySecretsKey1" : "MyValue1",
    "MySecretsKey2" : "MyValue2"
  }
}
```

You can create multiple applications and store multiple secrets as KeyValue in a single application using the POST API and retrieve those secret values using the VaultHelper function by passing AppName.

2. **Update:** PUT /pricing/api/admin/Vault/{appName}

It updates an existing application vault information with a new value.

Body:

```
{
  "AppName": "MyApp",
  "Secrets": {
    "MySecretsKey1" : "MyNewValue1",
    "MySecretsKey2" : "MyNewValue2"
  }
}
```

3. **Remove:** DELETE /pricing/api/admin/Vault/{appName}

It removes secrets from the vault storage.

Pricing Callback Class for TurboPricing

Pricing callback provides extensibility points in the Pricing Engine which can be used to extend or override existing behavior of the Pricing Engine based on customer requirements. Pricing callback classes allow you to add pricing logic to the cart that cannot be achieved by out-of-the-box pricing mechanisms, such as Price Rulesets and Price Matrices. The pricing callbacks are invoked during pricing action and execute the various pricing contact methods. The Pricing Callback is executed for each bundled product or standalone product in the cart. A separate transaction call is initiated for each product in the cart. The Pricing Callback is invoked when you are on the cart page.

To use the Pricing Callback you must create a custom .NET class that implements the following interfaces

Interface	Description
<i>IPricingBasePriceCallback</i>	This interface provides you a mechanism to define custom logic to be executed before, during and after Base Price Calculation
<i>IPricingTotallingCallback</i>	This interface provides you a mechanism to define custom logic to be executed before, during and after adjustment Calculation

You must only use the data synced with TurboPricing from Salesforce to write the custom logic. You can use the Helper Functions to retrieve the synced data from TurboPricing.

The following sections describe the interfaces and helper functions:



Pricing Base Price Callback Interface

The *IPricingBasePriceCallback* interface provides you a mechanism to define a custom logic to be executed before, during, and after **Base**

Price calculation. The *IPricingBasePriceCallback* is executed in batches which consists of the batch line items. TurboPricing calls this interface for every batch.

The following methods are available in the *IPricingBasePriceCallback* interface:

Method Signature	Description
<i>Task BeforePricingBatchAsync(BatchPriceRequest batchPriceRequest)</i>	You can use this method to define custom logic that must be executed before Base Price is calculated.
<i>Task OnPricingBatchAsync(BatchPriceRequest batchPriceRequest)</i>	You can use this method to define custom logic that must be executed during the Base Price calculation. You can use the price list items to write the custom logic.
<i>Task AfterPricingBatchAsync(BatchPriceRequest batchPriceRequest)</i>	You can use this method to define custom logic that must be executed after the Base Price is calculated.

Example Code


```

namespace Apttus.Lightsaber.Customer.Pricing
{
    public class PricingBasePriceCallback : CodeExtensibility,
    IPricingBasePriceCallback
    {
        public async Task AfterPricingBatchAsync(BatchPriceRequest batchPriceRequest)
        {
            await Task.CompletedTask;
        }

        public async Task BeforePricingBatchAsync(BatchPriceRequest
batchPriceRequest)
        {
            var batchLineItems = batchPriceRequest.LineItems.SelectMany(x =>
x.ChargeLines).Select(s => new LineItem(s)).ToList();

            //Example, setting custom field on line item before PLI resolution
            foreach(var batchLineItem in batchLineItems) {
                decimal extendedQuantity = batchLineItem.GetQuantity();
                decimal quantity = batchLineItem.GetQuantity();

                if(batchLineItem.IsOptionLine())
                {
                    LineItem rootBundleLineItemModel = new
LineItem(batchLineItem.GetRootParentLineItem().GetPrimaryLineItem());
                    decimal bundleQuantity = rootBundleLineItemModel.GetQuantity();
                    extendedQuantity = bundleQuantity * batchLineItem.GetQuantity();
                }

                batchLineItem.APTS_Extended_Quantity__c = extendedQuantity;
            }

            //You can also query DB here, and perform initial setup such as creating
required Dictionary, List and so on for later use in the callback code.

            await Task.CompletedTask;
        }

        public async Task OnPricingBatchAsync(BatchPriceRequest batchPriceRequest)
        {
            var batchLineItems = batchPriceRequest.LineItems.SelectMany(x =>
x.ChargeLines).Select(s => new LineItem(s)).ToList();

```

```

        foreach(var batchLineItem in batchLineItems) {
            PriceListItemModel priceListItemModel =
batchLineItem.GetPriceListItem();
            PriceListItem priceListItemEntity = priceListItemModel.Entity;

            if(batchLineItem.PriceListId != priceListItemEntity.PriceListId) {
                batchLineItem.APTS_Is_Contract_Pricing__c = true;
            }
        }
        await Task.CompletedTask;
    }
}
}

```

Pricing Totalling Callback Interface

The *IPricingTotallingCallback* interface provides you a mechanism to defined custom logic to be executed before, during, and after adjustment calculation.

The *IPricingTotallingCallback* is invoked once for each pricing request to calculate the total.

The following methods are available in the *IPricingTotallingCallback* interface:

Method Signature	Description
<i>Task</i> <i>BeforePricingCartAdjustmentAsync(Aggregate CartRequest aggregateCartRequest)</i>	You can use this method to define custom logic that must be executed before the adjustment is calculated.
<i>Task</i> <i>AfterPricingCartAdjustmentAsync(Aggregate CartRequest aggregateCartRequest)</i>	You can use this method to define custom logic that must be executed after the adjustment is calculated.
<i>Task</i> <i>OnCartPricingCompleteAsync(AggregateCart Request aggregateCartRequest)</i>	You can use this method to define custom logic that must be executed after the pricing is calculated completely.

Example Code

```

namespace Apttus.Lightsaber.Customer.Pricing
{
    public class PricingTotallingCallback : CodeExtensibility,
    IPricingTotallingCallback
    {
        public async Task BeforePricingCartAdjustmentAsync(AggregateCartRequest
        aggregateCartRequest)
        {
            var cartLineItems =
            aggregateCartRequest.CartContext.LineItems.SelectMany(x => x.ChargeLines).Select(s =>
            new LineItem(s)).ToList();

            foreach(LineItem cartLineItem in cartLineItems)
            {
                if(cartLineItem.IncentiveBasePrice.HasValue &&
                cartLineItem.IncentiveBasePrice.Value != 0) {

                    decimal sellingTerm =
                    cartLineItem.GetValueOrDefault(LineItemPropertyNames.SellingTerm, 1);
                    decimal lineItemQ = cartLineItem.GetQuantity();

                    decimal? unitIncentiveAmount = cartLineItem.BasePrice -
                    cartLineItem.IncentiveBasePrice;

                    cartLineItem.APTS_Unit_Incentive_Adjustment_Amount__c =
                    unitIncentiveAmount;
                    cartLineItem.IncentiveBasePrice = cartLineItem.BasePrice -
                    unitIncentiveAmount;
                    cartLineItem.IncentiveAdjustmentAmount = unitIncentiveAmount *
                    lineItemQ * sellingTerm * -1;
                }
            }

            await Task.CompletedTask;
        }

        public async Task AfterPricingCartAdjustmentAsync(AggregateCartRequest
        aggregateCartRequest)
        {
            //Example, Set custom fields on line item based on the adjusted price
            await Task.CompletedTask;
        }
    }
}

```

```
public async Task OnCartPricingCompleteAsync(AggregateCartRequest
aggregateCartRequest)
{
    var cartLineItems =
aggregateCartRequest.CartContext.LineItems.SelectMany(x => x.ChargeLines).Select(s =>
new LineItem(s)).ToList();
    foreach(var cartLineItem in cartLineItems)
    {
        if(cartLineItem.NetPrice < 1000)
        {
            cartLineItem.APTS_Deal_Color = "Red";
        }
        else
        {
            cartLineItem.APTS_Deal_Color = "Green";
        }
    }
    await Task.CompletedTask;
}
}
```

Frequently Asked Questions (TurboConfig)

What is TurboConfig and how does it work?

TurboConfig is a configuration engine to process product configuration rules while configuring products and finalizing a quote. TurboConfig offloads the computation workload from the Salesforce platform to the Apttus Flexible Compute Platform built using microservices. The benefit of the TurboConfig is that users can sell complex configurations much faster because of significantly-optimized processing time. Also, it allows customers to expand the solution to other business units and sell faster.

For example, in a TurboConfig-enabled flow, when the Sales Representative adds a product or a favorite configuration to cart, the application of complex constraint rules associated with them is offloaded to the Apttus Flexible Compute Platform to process for faster response.

When do I need to use TurboConfig?

The Salesforce platform has limitations (such as heap size, CPU timeout limits, number of SOQL limits, and view state) that result in slower response times and usability issues. TurboConfig handles such complex rules and processes a volume of rules significantly faster.

TurboConfig is recommended when you have a large number of rules or highly complex configuration rules to be applied while selecting a product or configuring a bundle.

For example

- If you have more than 100 constraint rules (inclusion, exclusion, recommendation and replacement rules) applicable across standalone and bundle products
- If you have more than 50 field expressions applicable across products and bundles
- If you have more than 100 product attribute value rules applicable across bundles
- If you have a complex bundle structure that includes more than 500 options and several option groups
- If you have complex bundles rules such as min/max, custom filter callback, repeat inclusions.

How do I enable TurboConfig?

You must have a license for TurboPricing or TurboConfig to enable either service. If you do not have a license, please contact your Apttus Account Executive before you begin. After you acquire a license TurboConfig instances will be provisioned for you.

For detailed instructions on how to enable TurboConfig, refer to [instructions on how to turn on TurboConfig](#).

Which version of CPQ should I be on to use TurboConfig?

You must be on CPQ on Salesforce Summer 2020 release or above.

What are the supported features in TurboConfig?

TurboConfig is released for limited availability in the Summer 2020 release. For a complete list of supported features on TurboConfig, refer to the [feature matrix](#).

Is the TurboConfig available for all products or only select products?

You can use TurboConfig for all or select products. However, you must publish the product before you use TurboConfig to configure the product. For instructions on how to publish products to TurboConfig, refer to the [instructions here](#).

How do I publish products so that I can use TurboConfig for those products?

You can publish products individually or in a group using the Publish Product REST API. For instructions on how to publish products to TurboConfig, refer to the [instructions here](#).

Is there a way to automate publishing of all products and changes on a regular basis?

In this release, the only way to publish a product or a group of products is by invoking a REST API. In the upcoming releases, Apttus provides an option to sync product data to TurboConfig at a regular frequency.

Does TurboConfig work on existing quote or configurations, which were created using a different constraint rule execution mode?

Yes, if you have quotes in progress and if you have configured quotes using the *Client* execution mode, you can process the quotes using the *CMS* execution mode. However, if you have created a quote using the *CMS* execution mode when you switch to the *Client* execution mode, you may have to delete the line items and add them again.

How do I switch from Server Side/Client Side constraint rules to TurboConfig?

For detailed instructions on how to enable TurboConfig, refer to [instructions on how to turn on](#). Also, refer to the [feature parity matrix](#) before you switch to TurboConfig. Note that custom callbacks are not supported in this release.

When I refresh my Salesforce Sandbox org, should I change any config settings?

When you refresh your sandbox, you must reconfigure TurboConfig after the refresh. Follow the onboarding process to enable TurboConfig in your sandbox after you have refreshed the sandbox.

I have done some customization in my org such as added formula fields, workflow rules, callbacks. Do my customizations work when I switch to TurboConfig?

There is no impact on any customizations you may have done on CPQ Objects. However, if you have written any configuration callbacks such as Option Filter Callbacks, you will be required to migrate your callback to TurboConfig using the microservice callback

framework. Note that the callbacks are not supported in TurboConfig in the Summer 2020 release. Refer to the supported [feature matrix](#) before switching to TurboConfig engine.

Is TurboConfig security and privacy complaint?

TurboEngines run in a secure multi-tenant environment and TurboEngines are designed to provide full security and privacy with your data. The services are hosted in either Microsoft Azure or IBM cloud, which are ISO 27001/2, SOC 1/2, GDPR compliant. Apttus takes advantage of data encryption and access control features enabled by the cloud service provider. If you have any questions or need details, contact Apttus Technical Support.

Apttus Copyright Disclaimer

Copyright © 2021 Apttus Corporation ("Apttus") and/or its affiliates. All rights reserved.

No part of this document, or any information linked to or referenced herein, may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written consent of Apttus. All information contained herein is subject to change without notice and is not warranted to be error free.

This document may describe certain features and functionality of software that Apttus makes available for use under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not, in any form, or by any means, use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part of the software. Reverse engineering, disassembly, decompilation of, or the creation of derivative work(s) from, the software is strictly prohibited. Additionally, this document may contain descriptions of software modules 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.

U.S. GOVERNMENT END USERS: Apttus software, including any operating system(s), integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Neither the software nor the documentation were developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Apttus and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Apttus and X-Author are registered trademarks of Apttus and/or its affiliates.

The documentation and/or software may provide links to Web sites and access to content, products, and services from third parties. Apttus is not responsible for the availability of, or any content provided by third parties. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the

relationship is directly between you and the third party. Apttus is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Apttus is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

For additional resources and support, please visit <https://community.conga.com>.

DOC ID: ATESUM20AGREVB20200923