Digital Commerce on Salesforce Summer 2020 Patch 1 Implementation and Deployment Guide

Table of Contents

About This Guide

With the Digital Commerce on Salesforce Implementation and Deployment Guide, you can find out how Apttus Digital Commerce works and how to install, implement and deploy Digital Commerce for your customers.

Topic	Description
What's Covered	This guide is designed to provide administrators with information on setting up data to be consumed within Apttus Digital Commerce. This guide covers the most common use cases for administration and assumes a level of familiarity with basic Salesforce.
Primary Audience	Admin users responsible for installing, implementing and deploying the Apttus Digital Commerce solution.
IT Environment	Refer to the latest Apttus Digital Commerce on Salesforce Release Notes for information on System Requirements and Supported Platforms.
Updates	For a comprehensive list of updates to this guide for each release, see the What's New topic.
Other Resources	• Digital Commerce User Guide: Refer to this guide for basic admin tasks and end user experience. • Digital Commerce SDK: Refer to this guide for technical instructions on the installation and setup of an Apttus E-Commerce storefront.

This guide describes the following tasks:

- Setting up Communities
- Adding a Storefront record
- Cloning and installing the reference template
- Local Development Setup
- Server Deployment
- Post Deployment Community Setup

Before using Digital Commerce, you must be familiar with the following:

- Basic Salesforce administration
- Salesforce and Apttus terms and definitions

What's New

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

Document	Topic	Description
Summer 20.05.17 (REV C)	Configuring TurboEngines	Updated topic to include enhancements with TurboPricing integration.
Summer 20.04.23 (REV B)	Configuring Data Sync for TurboPricing	Updated topic with additional information on selecting custom fields in the pricing master data.
Summer 2020 Patch 1 REV A	Configuring TurboEngines	New topic: New topic: A new feature for this release (TurboPricing Integration)
Summer 2020 Patch 1	Upgrading Digital Commerce	Updated information about how to upgrade Digital Commerce.
Spring 2020	Managing Permissions for Secure Guest User Access to Storefront	Updated topic. Salesforce Winter 2020 security alert introduced during Digital Commerce Winter 2019 and also applicable to Digital Commerce Spring 2020.
	Upgrading Digital Commerce	New topic. Added information about how to upgrade Digital Commerce.
	Server Deployment	Updated topic about how to include the UI as a static resource and deploy to Salesforce.
Winter 2019 Rev B	Managing Permissions for Secure Guest User Access to Storefront	New topic. Resolution added to workaround Salesforce Winter 2020 introduced security alert.
Winter 2019 Rev A	About Apttus Digital Commerce	Updated topic to conform with Apttus Documentation standards.
Winter 2019	All topics	Restructured to reflect the deployment flow.

Document	Topic	Description
Summer 2019 Rev A	About Apttus Digital Commerce	Updated topic.
	Enabling State and Country Picklists	New topic. Salesforce org level setting introduced in Spring 2019 Patch 9.
Summer 2019	• Payment Integration • Tax Integration with Avalara Tax Engine • Setting Up Multi-language Using Translations • Setting Email Notification Template for Checkout	New topics. New features for this release.
Spring 2019	Adding Custom Attributes to a Product	Existing topic. Changed config information.
Winter 2018	Adding Custom Attributes to a Product	New topic. A new feature for this release.
Summer 2018	All topics	Complete new guide.

About Apttus Digital Commerce

Digital commerce is the activity of electronically buying or selling products on online services or over the Internet. Digital commerce has three main components: Web Commerce, Mobile Commerce and Social Commerce, thus connecting channels and devices. Apttus Digital Commerce enables consumers to shop from their homes and offices, and even on the move.

As an administrator, you can use Apttus CPQ to perform configuration and pricing tasks to build the catalog page for your Digital Commerce web site. Configuration involves creating products, options, attributes, categories and associating them appropriately with each other for visibility on the Catalog page. A product can be created as a standalone product or as a bundle product with options and attributes. You can control the selection of a product on the catalog page by configuring constraint rules. You can also control the selection of attributes on the configuration page of a product by setting up attribute-based configuration for that product. Pricing enables you to set up pricing structures for the products so that the price for all products is calculated accurately. Pricing has two components: Price Lists and Price List Items. A price list controls the visibility of products to the user. A price list contains several price list items, each linked to a product. Apttus CPQ calculates the price for each product based on the applied price list, price list items, and various pricing and discounting rules.

After you have used Apttus CPQ to configure products and pricing, you can associate the price list to the storefront object within the Apttus E-Commerce package. After your catalog has been set up, the next step is to create a Storefront record. The storefront object is very basic and contains only two fields to map a storefront to a price list and logo for the guest user. The price list refers to the price list you want the guest user to access and the logo must be an ID or a URL of the logo attachment for the store. The storefront record also has a banner related list that can be used to set up banners for the jumbotron component in the reference template.

You can define the asset management functions with different data objects to track quote details until an order is fulfilled. You can set up multi-language web sites. You can apply promotions, make secure payments, and calculate tax on your cart page.

Apttus Digital Commerce allows an administrator to perform the following administrative tasks:

- Set up and activate communities
- Install Apttus E-Commerce Package and dependent packages
- Post deployment community setup
- Assign Apttus E-Commerce permission set
- Create custom field on account for the price list

- Set up single and multiple store
- Payment integration using Cybersource Payment
- Tax integration with Avalara tax engine
- Set up multi-language storefront using translations
- Set email notification template for checkout
- Enable state and country picklists
- Add a storefront record
- Add storefront promotional banners and associate it with the storefront record
- Clone and install the reference template
- Configure templates
- Set up the local development environment
- Server Deployment
- Customize your application
    - Customize HTML content and standard components
    - Add custom fields on object models
    - Add custom attributes to a product
    - Customize logic in the services
    - Customize the template page with custom field
    - Optimize Search Engine

Key Terminology

It is important to understand how terms are used when working with Apttus Digital Commerce.

Term	Description
Configure Price Quote (CPQ)	Apttus solution for configuring products, setting up pricing, and generating quotes.
Partner Commerce	Partner Commerce enables your partner company to create quotes, configure products, and manage orders for the end customer to ensure faster selling and up-time without your support.
Product Catalog	A view that allows hierarchical categorization of products for users to search through and add to their configuration.

Term	Description
Promotions	A promotion is a marketing technique that you apply to reduce the list price of a product or a service. You can create such a promotion and restrict the scope, limit, and benefits so your sales representatives apply this promotion to specific products, for specific customers, and for a limited period.
Options	A product that can be sold along with another product.
Attributes	Features of a product, such as color, size, weight, and more.
Communities	Apttus leverages Salesforce Communities to host your Digital Commerce site providing authentication and hosting features for your storefront. You can create multiple communities within your organization for multiple storefronts.
Storefront	Custom object that is part of the E-Commerce package. The storefront object maps a storefront to a price list and other basic information such as logo, banners and more.
Reference template/application	Base template provided by Apttus for further development and customization as per your requirement.
Payment integration	E-Commerce payment integration using Cybersource.
Tax integration	Tax integration using Avalara tax engine with the help of a Tax Callback class.
Translations	If your org has multiple languages enabled, use Translation Workbench to maintain your translated labels in your org. You can manage translated values for any Salesforce supported language.

Getting Started

To get started with Digital Commerce on Salesforce, you must complete the following pre-requisites:

1. Ensure that you fulfill the minimum system requirements.
2. Install E-Commerce package and all dependent packages in your Salesforce org.
3. Configure Products, Price List, and Catalog and ensure you can create a proposal in CPQ.
4. Ensure you have a User with the System Admin profile.
5. Ensure you have access to E-Commerce GIT repository (for base template and all associated libraries)
6. Ensure you have NPM (Node Package Manager) login credentials for Apttus NPM private repository.

System Requirements

Before you install and start using or customizing your Apttus Digital Commerce application, you must set up your environment for Angular development using the Angular CLI tool. This topic includes information about minimum system requirements and configuring the environment variables.

Minimum Software Requirements

Before you begin, make sure your development environment includes Node.js® and an npm package manager.

> ⚠ Command prompt must run as administrator for all command line operations.

1. Node JS v8.10.0 or higher: Install Node JS from the official Node website. For windows, download Windows installer .msi 64-bit.
2. Angular CLI v1.7.3 or higher:
   - To check if already installed, use command: *ng -v*
   - If not installed, use command: *npm install -g @angular/cli*
3. Windows Build Tools. Install python gyp using command: *npm install --global --production windows-build-tools*
4. Visual Studio Code
5. JDK / JRE 8 or Higher

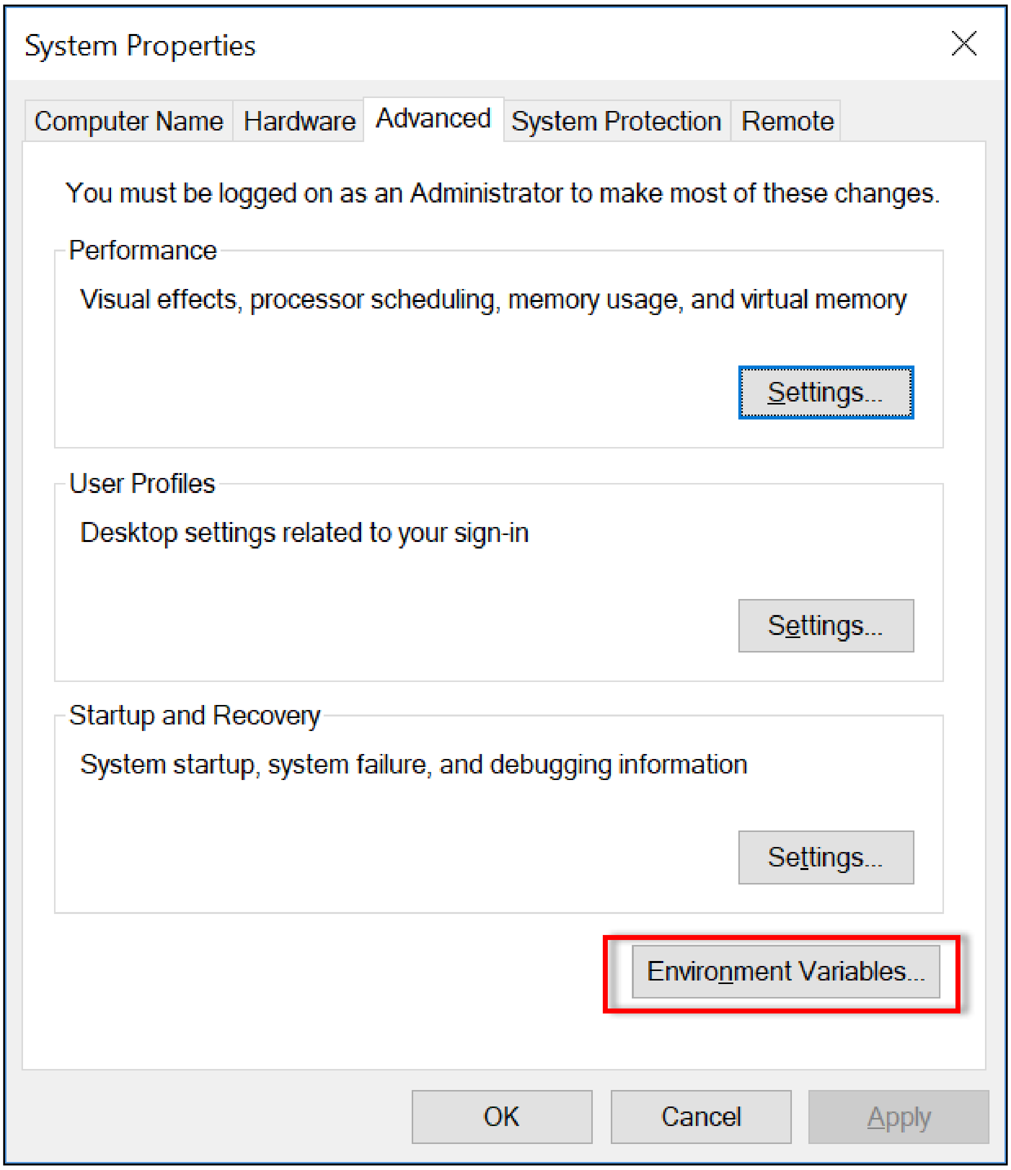
6. Apache Ant^{TM} (v1.10.3 or higher) - Install apache-ant-1.10.3-bin.zip or higher

Environment Variables

When NodeJs, JDK/JRE, and ANT are installed, it is necessary that they are accessible globally using the command prompt. Sometimes the environment variables are not configured and you may see errors stating 'node'/'java'/'ant' is not recognized as an internal or external command. You must configure the environment variables on your local system.

To configure environment variables

1. Go to This PC > System Properties > Advanced System Settings.
2. Click Environment Variables.



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

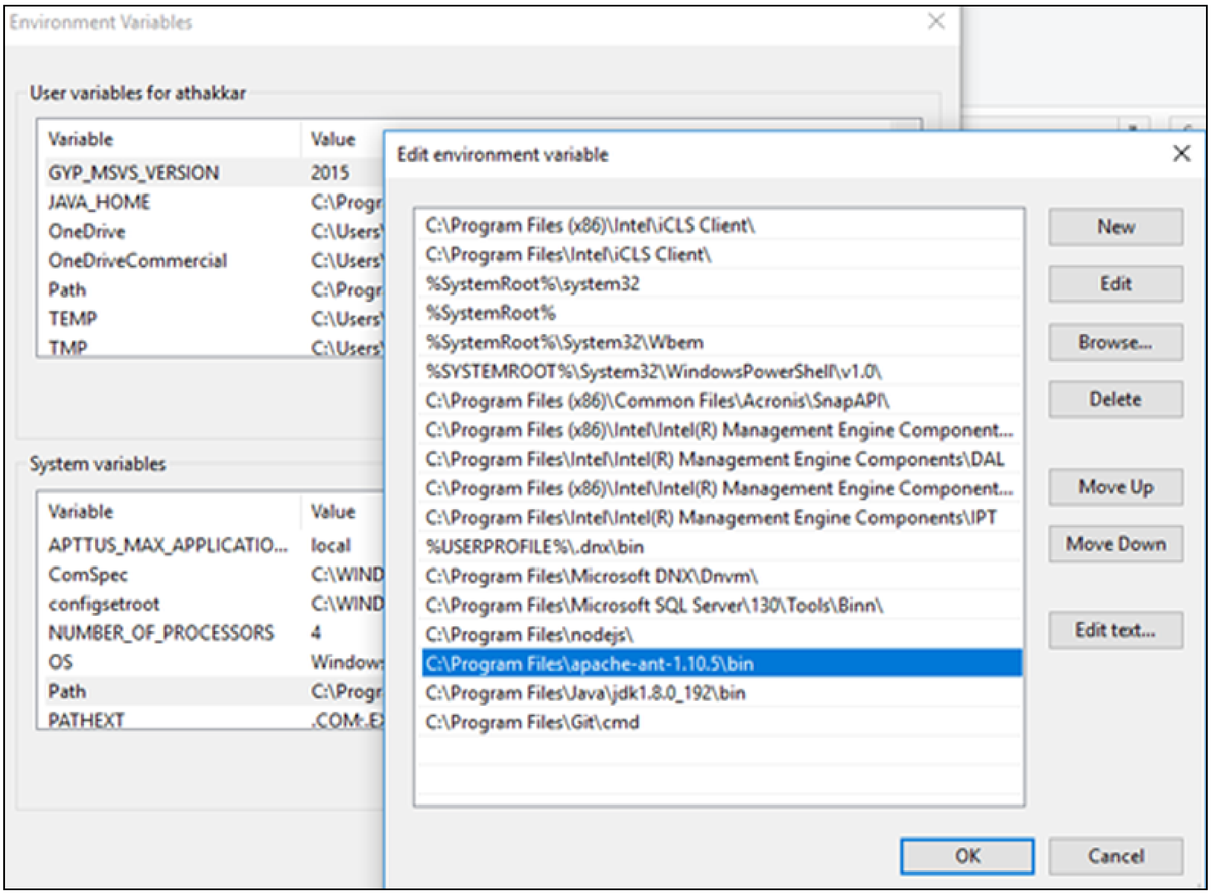
User Profiles

Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

3. Add the necessary paths into the User Variable



User variables for athakkar

Variable	Value
GYP_MSVS_VERSION	2015
JAVA_HOME	C:\Progr
OneDrive	C:\Users\
OneDriveCommercial	C:\Users\
Path	C:\Progr
TEMP	C:\Users\
TMP	C:\Users\

System variables

Variable	Value
APTTUS_MAX_APPLICATIO...	local
ComSpec	C:\WIND
configsetroot	C:\WIND
NUMBER_OF_PROCESSORS	4
OS	Window
Path	C:\Progr
PATHEXT	.COM;.E

```
C:\Program Files (x86)\Intel\iCLS Client\
C:\Program Files\Intel\iCLS Client\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\Common Files\Acronis\SnapAPI\
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\IPT
%USERPROFILE%\.dnx\bin
C:\Program Files\Microsoft DNX\Dnvm\
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
C:\Program Files\nodejs\
C:\Program Files\apache-ant-1.10.5\bin
C:\Program Files\Java\jdk1.8.0_192\bin
C:\Program Files\Git\cmd
```

Installing Apttus E-Commerce Package

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

Caution

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

Install the packages in the following order.

Order	Package	Install Center tab to access the package	Required?
1	Apttus Contract Management	Contract Management	Y
2	Apttus E-Commerce	CPQ	Y
3	Apttus Proposal Management	CPQ	Y
4	Apttus Base Library	Contract Management	Y
5	Apttus Configuration & Pricing	CPQ	Y
6	Apttus Quote/Proposal-Configuration Integration	Integrations	Y
7	Apttus Quote/Proposal-Asset Integration	CPQ	Y
8	Apttus CPQ Admin	CPQ	Y
9	Apttus CPQ API	CPQ	Y
10	Apttus Promotion Management	CPQ	Y
11	Apttus Billing Management (Required only if Cybersource Payment package is installed)	Billing	Y

Ensure that at least Apttus CPQ and E-Commerce package licenses are assigned to an admin user, community user, and guest user.

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

Downloading the Digital Commerce SDK

All the Digital Commerce SDK libraries are hosted at https://art01.apttuscloud.io/api/npm/npm-release.

After cloning the reference template, you must update the npmrc file in the reference template with the SDK library hosted URL and your own secret key.

Contact Apttus Professional Services team to get your own secret key.

Logging in to Apttus Digital E-Commerce

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

- All Apttus Digital Commerce packages are installed.
- You have administrative privileges.
- You have login credentials provided by Apttus.

To log into the application

1. Go to http://www.salesforce.com.
   Or
   If your organization is using a sandbox or test environment (for example, if you are doing user acceptance testing), go to http://test.salesforce.com instead.
2. In 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 to Salesforce**.

You have successfully logged into the application.

Configuring TurboEngines

TurboEngines is a concurrent processing engine provided by Apttus that comprises various microservices that process pricing calculations (**TurboPricing**). TurboEngines offload the computation workload from the Salesforce platform to the **Conga Flexible Compute Platform** to reduce the processing time on the cart. Processing the computation workload in the Flexible Compute Platform reduces the interaction costs and the quote turnaround time specifically during peak load or large transactions. By default, the digital commerce package deploys in the non-TurboEngines mode. You can switch TurboEngines mode as per your business setup.

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 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.

The following section provides information and step-by-step tasks for enabling TuboPricing for your organization.

- Enabling TurboPricing in an Org
- Creating a Connected App
- Preparing Tenant Information
- Setting Up the TurboPricing Endpoint URLs
- Setting Up the Pricing Execution Mode
- Configuring Data Sync for TurboPricing
- Pricing Mechanism

Enabling TurboPricing in an Org

This topic provides a summary of the necessary steps for enabling 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 on this topic, this persona is referred to as the *Tenant Admin*.

Prerequisites

- Check the "Supported Features" topic (under the User Guide) to 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 Configuration & Pricing (CPQ) in the Salesforce org to enable TurboPricing. Refer to "Packages" in *CPQ on Salesforce Summer 2020 Release Notes*.

Enabling TurboPricing

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

Step	Task	Owner	Description
Pre-Provisioning Tasks			

Step	Task	Owner	Description
1	Set up Connected App in your org	Tenant Admin	Create a connected app to provide authentication and authorization to TurboPricing Data Sync Service.
2	Prepare pre-provisioning tenant information	Tenant Admin	Gather all required information for provisioning your TurboConfig or TurboPricing org. Provide this information to Apttus Technical Support to begin the provisioning process.
Post-Provisioning Tasks ⓘ 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.			
3	Configure Services	Tenant Admin	Update the following settings: 1. Set up the Pricing Execution Mode 2. Set up the TurboPricing endpoint URL
4	Configure data sync settings for TurboPricing	Tenant Admin	Complete the data sync for post-provisioning. • For TurboPricing
5	Sync data to TurboEngines	TurboEngines Administrator (can be Tenant Admin)	Set up and schedule or activate data sync to sync pricing master data.

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 TurboPricing services.

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.

Field	Description
Enable OAuth Settings	Select this to define the OAuth settings. For example, *TurboPricing*. 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.
Callback URL	The **Callback URL** is generated automatically when you select the field **Enable for Device Flow**. For example, *https://test.salesforce.com/services/oauth2/success* 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, the application generates **Consumer Key** and **Consumer Secret**. You must provide the values of **Consumer Key** and **Consumer Secret** to the Technical Support team.

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 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 the Technical Support team.

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

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

Configuration	Required for Service	Description
OrgId	TurboPricing	This is the Salesforce Organization ID of the org to be provisioned for TurboEngine service. To locate your Organization ID: 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).
Org Type	TurboPricing	Org type to be provisioned (sandbox or production)
Tenant Name	TurboPricing	The one word tenant name used for the tenant endpoint (for example, *customername-sandbox*)
Consumer Key	TurboPricing	The consumer key (`client-id` in OAuth 2.0) generated from your Connected App. Refer to Creating a Connected App.
Consumer Secret	TurboPricing	The secret key (`client-secret` in OAuth 2.0) generated from your Connected App. Refer to Creating a Connected App.

Configuration	Required for Service	Description
Salesforce User Name	TurboPricing	Admin username for the org to be provisioned with read/write access to CPQ (used by the Technical Support team for verifying settings)
Salesforce Password	TurboPricing	Password for the Salesforce admin user.
Authority	TurboPricing	The URL used to verify session Id for TurboPricing (*login.salesforce.com*, *test.salesforce.com*, or a custom Salesforce domain)
InstanceURL	TurboPricing	The URL given by the UI after logging into the org to be provisioned (for example, *customerturbo.my.salesforce.com*)

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

Setting Up the TurboPricing Endpoint URLs

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

To add *APTS_PricingServiceOverrideURI*

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_PricingServiceOverrideURI.*
4. In the **Value** field, enter the TurboPricing endpoint URL (without https://).
5. Click **Save**.

To add *APTS_PricingServiceUrl*

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_PricingServiceUrl.*
4. In the **Value** field, enter the TurboPricing endpoint URL (with https://).
5. In the **Code** field, enter the Org ID.

> ⚠ You must convert the 15 characters Org ID to 18 characters and add it in the lowercase.

6. 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

The TurboEngines Data Sync administrator user interface allows administrators to manage, modify, and monitor data sync operations of master data between Conga and various consumer endpoints. Administrators can make changes to data sync consumer profiles by adding, updating, or deleting objects and fields for sync. They can also monitor data sync jobs status and run history and manually trigger data sync for any given consumer profile.

The standard and custom objects that are synced from Salesforce to TurboEngines are defined as Consumer Profiles. Consumer Profiles are a master list and format definition for objects, fields, and the related objects and fields to be synced to a specific consumer endpoint. These profiles are displayed in the Consumer list under the Data Integration tab.

Configure the Custom Fields at Consumer Profiles

Before performing data sync, you must configure any custom fields at consumer profiles.

To configure custom fields

1. Open the Data Sync Admin UI. Go to the **Salesforce App Launcher > TurboEngines Admin**.
2. Go to the **Data Integration** tab. From the list of Consumers, click **Manage Sync** next to Pricing Master Data Tables.

3. From the list of Objects, scroll down to the object from which you want to select fields to be synced, click the down arrow button next to it and click **Manage Objects**.
4. Under the Available Fields, click **Managed Fields**. The Managed Fields dialog appears.
5. From the list of fields, select the custom fields that you want to sync.
6. Click **Submit**.

Post-Provisioning Tasks

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 Digital Commerce and TurboPricing. Before the initial data sync, you must configure settings to 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

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 the Technical Support Team
5. Click **Save**.

To configure the CSD Trusted Site

1. Go to **Setup** > **Administration Setup** > **Security Controls** > **CSD 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 the previous section (To 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 on the list. Click the drop-down and 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

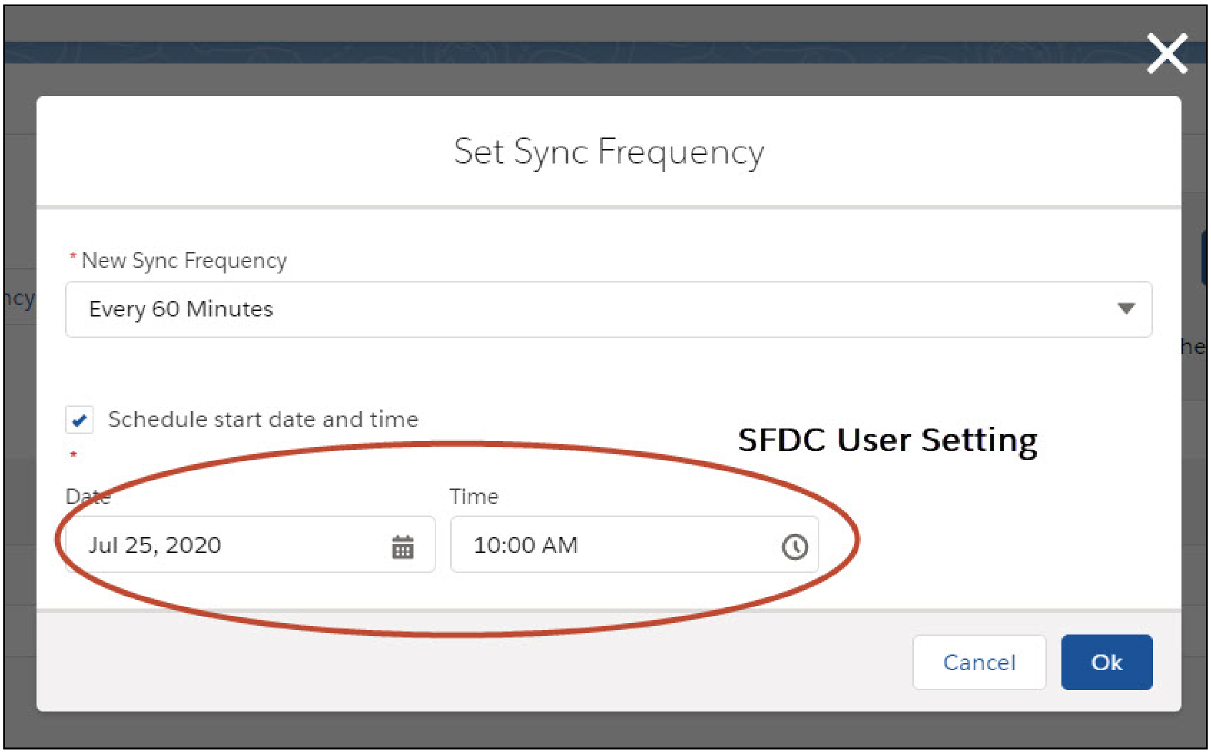
1. Go to **Setup > Profiles**.
2. Search for the profile you want to configure and click **Edit**.
3. Under Custom Settings, make sure the following tabs are set as "Default On":
   - **Data Integration**: This tab serves as the starting point for managing all consumer profiles.

- **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*.



Set Sync Frequency

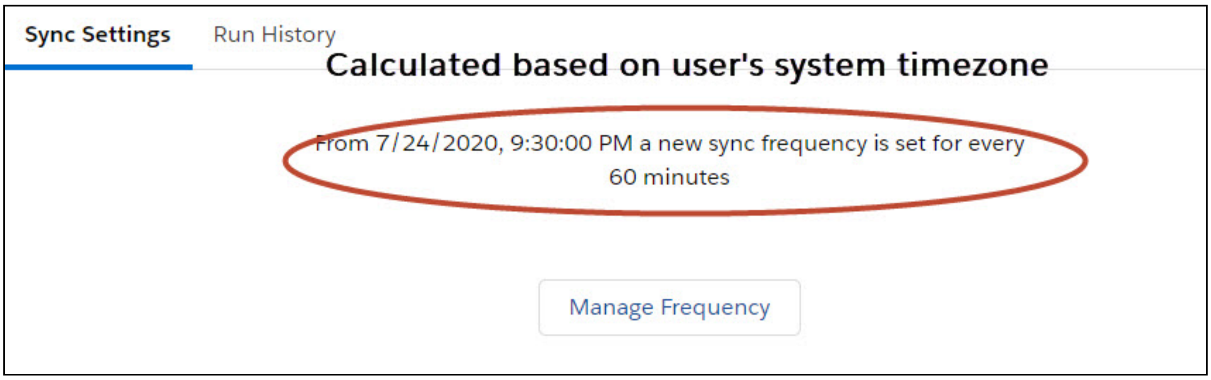
* New Sync Frequency: Every 60 Minutes

☑ Schedule start date and time

Date: Jul 25, 2020

Time: 10:00 AM

However, the scheduled date and time displayed by the Data Sync Admin UI is calculated based on the user's *system timezone*.



From 7/24/2020, 9:30:00 PM a new sync frequency is set for every 60 minutes

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 synchronized, the next scheduled sync date and time are 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 the latest *TurboEngines Data Sync Administrator Guide* on the Documentation portal.

Limited Support for Special Characters in Product Search Functionality

Support	Characters
Not supported	/ \ * %
May work	( ) [ ] { } + - _ = @ " $ &
Not certified	<> ^ ~ ! : ; # ` . ,

Pricing Mechanism

You can change the pricing of products using the following flags at the template level:

- skipPricing
- skipRules

The Digital Commerce SDK consists of the following flag to change pricing. These flags are available in the environment file.

Flag	Description
skipPricing	By default, this flag is set to True. When the user changes an existing cart, Digital Commerce skips pricing meaning pricing still runs in the background making another network call, thus scalable to large carts. Setting this flag to False allows Digital Commerce to perform pricing when the user changes an existing cart, meaning all CRUD operations such as update quantity, delete a line item, data change along with pricing on the cart are performed in a single network call. This is ideal for a cart with up to 50 line items. For more than 50 line items, it is recommended you set the skipPricing flag to True. ⚠ When skipPricing - True, the system can use CPQ pricing engine or TurboPricing engine, whereas skipPricing = False, the system uses CPQ pricing engine only.
skipRules	By default, this flag is set to False. When the user changes an existing cart, Digital Commerce evaluates the rules. Setting this flag to True skips rules when the user changes an existing cart.

You may use these flags with the following combinations to allow Digital Commerce to make changes to the cart with or without pricing parameters.

- skipPricing = True and skipRules = False
- skipPricing = False and skipRules = True
- skipPricing = False and skipRules = False

E-Commerce Package Objects

Digital Commerce is comprised of multiple packages as mentioned in the installation section and this section lists out the objects inside the package.

Object Name	Purpose of the Object
Storefront	Represents the storefront details.
Storefront Banner	Represents the storefront banner details.
Api Cache	Stores record information such as Line Item Ids and Attribute metadata for quick loading thus improving performance. When changes are made in the metadata of an object you are using, you should always clean the Api Cache objects.

Upgrading Digital Commerce

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

> ⓘ If you do not have Digital Commerce installed, you can contact Apttus Support to request an installation link, then perform the standard installation as described in Installing Apttus E-Commerce Package.

Preparing for Upgrade

Before you upgrade to Digital Commerce on Salesforce Summer 2020 Patch 1, you must ensure the following:

- You go through *Digital Commerce on Salesforce Feature by Release* to know about the new features, enhancements, and deprecated features in Digital Commerce since your existing release. After you upgrade Digital Commerce to Summer 2020 Patch 1, you cannot roll back to any previous release.
- You have supported platforms and system requirements.
- You have access to the **Install Center** on the Apttus Community Portal for Digital Commerce managed package and dependent packages.
- You have access to the npm-release repository for Digital Commerce SDK.
- You have administrator privileges to your Salesforce org.
- You need not back up your configurations. All configurations you performed since you installed your existing release will remain intact after the upgrade.

Upgrading to Digital Commerce on Salesforce Summer 2020 Patch 1

This section describes step-by-step instructions to upgrade from Spring 2020 and Winter 2019 to Summer 2020 Patch 1.

Upgrading Digital Commerce Spring 2020 to Summer 2020 Patch 1

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

Order	Product	*Version Name | Version Number*
1	Apttus Contract Management	9.3.0432.5 | 9.432.5
2	Apttus E-Commerce	4.0.52 | 4.52 Spring 2020
3	Apttus Digital Commerce SDK	2001.0.125 (2001)
4	Apttus Proposal Management	9.2.0206 | 9.206
5	Apttus Base Library (Pre-requisite for Installing Apttus Configuration & Pricing)	1.0.36 | 1.36
6	Apttus Configuration & Pricing	12.0.1715 | 12.1715
7	Apttus Quote/Proposal Configuration Integration	11.2.0312 | 11.312
8	Apttus CPQ Admin	11.1.39 | 11.39
9	Apttus CPQ API	11.2.0099 | 11.99
10	Apttus Promotion Management (Required if you are using Promotions)	1.0.0000 | 1.0
11	Apttus Quote/Proposal-Asset Integration (Required if you are using Assets Based Ordering)	6.5.0014 | 6.14
12	Apttus Order Management	1.0.0000 | 1.0

Order	Product	*Version Name | Version Number*
13	Apttus Billing Management	5.1.0158 | 5.158

2. Ensure that you have the following packages and dependent packages to upgrade to Summer 2020 Patch 1. These packages are required to utilize the new features and enhancements of Summer 2020 Patch 1.

Order	Product	*Version Name | Version Number*
1	Conga Contract Lifecycle Management	11.1.0547 | 11.547
2	Conga Digital Commerce	4.1.85.2 | 4.85.2 Summer 2020 Patch 1
3	Conga Digital Commerce SDK	2002.1.3 (2002)
4	Conga Quote Management	10.1.0221 | 10.221
5	Conga Base Library (Pre-requisite for Installing Conga Configuration & Pricing)	1.1.93 | 1.93
6	Conga Configuration & Pricing	12.1.1787.6 | 12.1787.6
7	Conga Quote Configuration Integration	12.1.0332 | 12.332
8	Conga CPQ Setup	12.1.61 | 12.61
9	Conga CPQ API	12.1.0104 | 12.104
10	Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
11	Conga Quote -Asset Integration (Required if you are using Assets Based Ordering)	6.5.0014 | 6.14

Order	Product	*Version Name | Version Number*
12	Conga Order Management (Required only if Partner Commerce package is installed)	1.0.0000 | 1.0
13	Conga Billing	6.2.238.1 | 6.238.1

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 above. For detailed information on installing managed packages, see Installing Apttus E-Commerce Package.
4. After the upgrade is complete, perform the post-upgrade tasks.

Upgrading Digital Commerce Winter 2019 to Summer 2020 Patch 1

1. Go to **Setup** > **Installed Packages** and ensure that your Salesforce org has the following Winter 2019 packages installed.

Order	Product	*Version Name | Version Number*
1	Apttus Contract Management	9.3.0432.5 | 9.432.5
2	Apttus E-Commerce	3.25 | 3.25 Winter 2019
3	Apttus Digital Commerce SDK	1903.0.0 (1903)
4	Apttus Proposal Management	9.1.0197 | 9.197
5	Apttus Configuration & Pricing	11.2.1675 | 11.1675
6	Apttus Quote/Proposal Configuration Integration	11.2.0310 | 11.310
7	Apttus CPQ Admin	11.1.39 | 11.39
8	Apttus CPQ API	11.2.0096 | 11.96

Order	Product	*Version Name | Version Number*
9	Apttus Promotion Management (Required if you are using Promotions)	1.0.0000 | 1.0
10	Apttus Quote/Proposal-Asset Integration (Required if you are using Assets Based Ordering)	6.5.0014 | 6.14
11	Apttus Order Management	1.0.0000 | 1.0
12	Apttus Billing Management	5.1.158 | 5.158

2. Ensure that you have the following packages and dependent packages to upgrade to Summer 2020 Patch 1. These packages are required to utilize the new features and enhancements of Summer 2020 Patch 1.

Order	Product	*Version Name | Version Number*
1	Conga Contract Lifecycle Management	11.1.0547 | 11.547
2	Conga Digital Commerce	4.1.85.2 | 4.85.2 Summer 2020 Patch 1
3	Conga Digital Commerce SDK	2002.1.3 (2002)
4	Conga Quote Management	10.1.0221 | 10.221
5	Conga Base Library (Pre-requisite for Installing Conga Configuration & Pricing)	1.1.93 | 1.93
6	Conga Configuration & Pricing	12.1.1787.6 | 12.1787.6
7	Conga Quote Configuration Integration	12.1.0332 | 12.332
8	Conga CPQ Setup	12.1.61 | 12.61
9	Conga CPQ API	12.1.0104 | 12.104

Order	Product	*Version Name | Version Number*
10	Conga Promotions (Required if you are using Promotions)	1.0.0000 | 1.0
11	Conga Quote -Asset Integration (Required if you are using Assets Based Ordering)	6.5.0014 | 6.14
12	Conga Order Management (Required only if Partner Commerce package is installed)	1.0.0000 | 1.0
13	Conga Billing	6.2.238.1 | 6.238.1

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 above. For detailed information on installing managed packages, see Installing Apttus E-Commerce Package.
4. After the upgrade is complete, perform the post-upgrade tasks.

Upgrading Digital Commerce SDK

This section lists the process of migrating your E-Commerce application to the Summer 2020 Patch 1 release.

To upgrade Digital Commerce SDK from Spring 2020 to Summer 2020 Patch 1

Refer to the Digital Commerce on Salesforce Summer 2020 SDK, and go to **Upgrade Information > Upgrade ecommerce template from spring v2001 to summer v2002.1.3**. Perform the following tasks:

1. Update the Apttus Libraries
2. Remove ConstraintSideMenuModule
3. Replace ProductConfigurationSummaryModule with ConfigurationSummaryModule and Removing CmsModule & ConstraintAlertModule
4. Remove CMS Related Code From the Summary Component

5. Remove the CMS Code from the Product-detail Component

To upgrade Digital Commerce SDK from Winter 2019 to Summer 2020 Patch 1

Refer to the Digital Commerce on Salesforce Summer 2020 SDK, and go to **Upgrade Information > Upgrade ecommerce template from winter v1903 to summer v2002.1.3**. Perform the following tasks:

1. Update Apttus Libraries
2. Replace ConstraintIconModule and ConstraintSideMenuModule with ConstraintRuleModule
3. Import BsModalRef from 'ngx-bootstrap/modal'
4. Replace apt-cr-side-menu with apt-constraint-rule-sidebar
5. Replace ProductConfigurationSummaryModule with ConfigurationSummaryModule
6. Update the Cart Model
7. Remove detailPageAlert from apt-cr-constraint-alert
8. Fix the Cart List Component
9. Pass the aboEnabled Flag to the add-to-cart component from product-detail.component.ts
10. Update the Quote Model

Performing the Post Upgrade Tasks

After you upgrade to this Digital Commerce release, consider the following options and requirements:

- Managing Permissions for Secure Guest User Access to Storefront
- All upgrade scenarios must work the same way as the previous release.
    - Layout changes: Validate layout changes.
    - RemoteAPIcalls: All API calls must be functional.

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

Setting Up Communities

To deploy the Digital Commerce code, you must set up and enable a Community.

- Enabling Communities
- Creating Communities in your Org
- Creating Community Users
- Creating Sites
- Activating the Community

The Apttus E-Commerce platform leverages a Salesforce Community to provide authentication and hosting features for guest users. After the E-Commerce package is deployed, the next step is to create a Salesforce Community. At minimum, you just need the community URL. However, if you intend to support guest users, you will need to enable that within the community settings. After deployment, the angular library will provide a Visualforce page that you can set as the default page for all page settings within the community. For example: home, login, forgot password, change password, and more). Being that it is a single page application that is designed to handle all incoming requests.

Enabling Communities

You must enable communities to create communities.

To enable community

1. Go to **Setup** > **Customize** > **Communities** > **Communities Settings** and select **Enable communities**.
2. Select a domain name for your communities, and click **Check Availability** to see if the domain name is available.
3. If you see a success message, click **Save**.

The Communities page is refreshed and displays All Communities section.

Creating Communities in your Org

You can create communities using a wizard. This wizard helps you select community templates that suit your business requirement.

To create a community

1. Go to **Setup** > **Customize** > **Communities** > **All Communities** and click **New Community**.
2. Select from one of the Standard Community templates. For example, Salesforce Tabs + Visualforce.
3. Click **Get Started**.
4. Type a community **Name**. **URL** displays the domain name of your community. In **Optional**, type a suffix for your community and click **Create**. For example, Name = E-Commerce, Optional = ecomm.

Your community is created.

Creating Community Users

You can create community users and enable them to access your community.

To create community users

1. Add a user to an account as a contact record and enable the contact record as a customer user or partner user, depending on your business relationship.
2. Refer to Create Community Users for detailed information.

Creating Sites

Sites enable you to create your E-Commerce websites that are directly integrated with your Salesforce org—without requiring users to log in with a username and password. To create sites, refer to Create Sites for detailed information.

Activating the Community

After you set up a community, you must activate the community.

To activate a community

1. Go to **Setup** > **Customize** > **Communities** and click **All Communities**.

2. Click **Workspaces** and under My Workspaces click **Administration**.
3. From the Settings page, click **Activate**.

Your community space is now activated.

Post Deployment Community Setup

Setting Up the Default Page

You can set up the default page for your community. This eliminates the need to suffix your community URL with the storefront you created.

To set up a default page

1. Go to **Setup** > **Customize** > **Communities** and click **All Communities**.
2. Click **Workspaces** and under My Workspaces, click **Administration**.
3. Select **Pages** and from **Community Home**, select Visualforce page.
4. Search and select the visualforce page you deployed.
5. Click **Save**.

Now when you go to your community URL, your storefront is displayed.

Granting User Access to Community via Profiles

You can enable users to access community through profiles based on the level of access you want to grant. Ensure that you provide access to at least an admin user, community user (e-commerce or partner) and guest user.

To enable users to access a community

1. Go to **Setup** > **Customize** > **Communities** and click **All Communities**.
2. Click **Workspaces** and under My Workspaces, click **Administration** and select **Members.**
3. Under Select Profiles section, from Available Profiles column, select a profile and add it to Selected Profiles column.
4. Under the Select Permission Sets section, from Available Permission Sets column, select a permission set and add it to Selected Permission Sets column.rch and select the visualforce page you deployed.
5. Click **Save**.

Enabling Self Registration

You can enable self registration and other user management tasks from the community administration page.

To enable self registration

1. Go to **Setup** > **Customize** > **Communities** and click **All Communities**.
2. Click **Workspaces** and under My Workspaces, click **Administration** and select **Login & Registration.**
3. For **Login**, select Visualforce page, and search and select the visualforce home page you deployed. For example: store
4. From the **Password** section, do the following:
   - For **Forgot Password**, select Visualforce page, and search and select the visualforce home page you deployed. For example: store
   - For **Change Password**, select Visualforce page, and search and select the visualforce page you deployed for a password change request. For example: storepassword.

   > ⚠ This is based on the Salesforce behavior of handling sessions. You cannot use the same visualforce page for both Home page and Change Password page.
5. From the Registration section, do the following:
   a. To enable self registration, select Allow external users to self-register.
   b. From Page, select Visualforce page, and search and select the visualforce home page you deployed. For example: store
   c. From Assign Registering Users To, set up the default Profile and Account for the self-registration.
6. Click **Save**.

Setting Up Guest Users

You can set up guest users for your community. The concept of guest users is simply hiding access to certain pages.

To set up a guest user

1. Go to **Setup** > **Customize** > **Communities** and click **All Communities**.
2. Click **Workspaces** and under My Workspaces, click **Administration**.
3. Select **Pages,** select the Force.com section. This section takes you to the underlying site record for your Salesforce community.
4. Under the Site Visualforce Pages section, ensure the pages are listed for the guest user to access. The pages that are not listed cannot be accessed by a guest user.
5. Once done, click **Public Access Settings** where you can see the guest user profile for our storefront. This displays what a guest user can access and manage object and field level permissions.
6. Click **Save**.

Apttus E-Commerce Permission Set

The E-Commerce package comes with a basic permission set for providing the necessary access to users. The permission set is named 'Apttus Ecommerce' and should be assigned to users access the e-commerce storefront. If you would like to make any changes to the permissions, you may clone the permission set and make any changes necessary.

Using the managed permission set may prevent a portal user to login to Digital Commerce. This occurs because the managed permission set does not have API Enabled Permission by default. You must create an unmanaged permission set and enable API permission.

To enable API permission

1. Clone Apttus Ecommerce permission set and name it Apttus Ecommerce Custom.
2. Go to **Setup** > **Administration Setup** > **Manage Users** > **Permission Sets** and select the permission set. In this case, Apttus Ecommerce and Apttus Ecommerce Custom.
3. Go to **System Permissions** and select to check **API Enabled**.
4. Click **Save**.

Everytime you assign Apttus Ecommerce permission set to a user, you MUST assign Apttus Ecommerce Custom permission set as well.

Creating Custom Field on Account for the Price List

PriceListId gets added to CPQ on the Account object. Create a custom field on the Account for the Price List with API name PriceListId__c. You can use this Price List in the Storefront record, to set up your Storefront with Categories, Products and more.

Setting Up Single and Multiple Store

You can set up a single store within a community as well as multiple stores within the same community. Communities are used to segment the users. For example, if you want users to view all your storefronts, you can create one community with multiple storefronts. In case, you want to restrict set of users to different stores, you must create separate communities to restrict access.

You can achieve this by creating different visualforce pages and control access through profiles and permissions sets.

Managing Permissions for Secure Guest User Access to Storefront

Salesforce Winter 2020 introduced several security alerts, which add default security to the records that power a storefront and therefore restrict what guest users can access on the storefront.

- An optional sharing setting, **Secure guest user record access**, is available that affects guest user access in the following ways:
  - Changes the organization-wide default (OWD) external access settings to apply only to community users. By default, guest users have no access to any records used for the storefront.
  - Adds a sharing rule type, **Guest user access, based on criteria**, which can grant only Read access to records, based on criteria you specify.
  - Restricts guest users from being included in public groups.
  - Restricts records from being manually shared with a guest user.
- An optional Communities setting, **Reassign new records created by guest users to the default owner**, is available that restricts a guest user from owning any Object records, such as Apttus_Config2__ProductConfiguration__c. This means that, after

enabling the setting, a guest user can only view or add to a cart created *before* the setting was enabled, and cannot create a cart.

- An optional Communities setting, **Let guest users see other members of this community**, is available that is disabled by default. This setting does not affect any storefront functionality.
- The View All Users permission is disabled by default for any new org.

To resolve this, the security alerts are *mandatory* since February 2020. The security alerts are opt-in until that date.

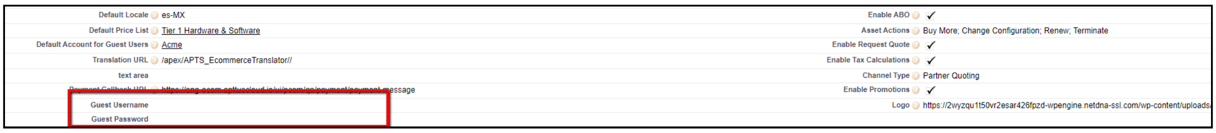
If you allow guest users to perform any of the following tasks, we recommend that you install the appropriate patch for your version of Apttus Digital Commerce and immediately complete the following required workaround steps to maintain guest user access on your storefront.

- Self-register
- Browse your storefront
- Add products to a cart
- Complete checkout

The solution for all versions of Apttus Digital Commerce involves creating a dedicated licensed user to act on behalf of the guest users for a storefront. The standard Community guest user can still be used, however, those users cannot perform the actions mentioned above.

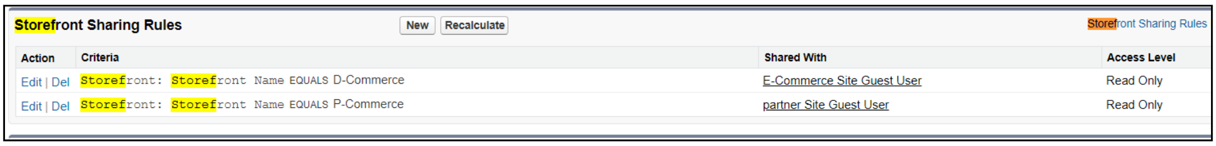
To maintain guest user access on your storefront

1. Click All tabs and click Storefronts.
2. In the Storefront record, enter the Guest Username and Guest Password for the guest user account. These fields are used for all guest users.



Default Locale	es-MX	Enable ABO	✓
Default Price List	Tier 1 Hardware & Software	Asset Actions	Buy More; Change Configuration; Renew; Terminate
Default Account for Guest Users	Acme	Enable Request Quote	✓
Translation URL	/apex/APTS_EcommerceTranslator//	Enable Tax Calculations	✓
text area		Channel Type	Partner Quoting
Payment Callback URL	https://eng-ecom.apttuscloud.io/ui/pcom/qc/payment/payment-message	Enable Promotions	✓
Guest Username		Logo	https://2wyzqu1t50vr2esar426fpzd-wpengine.netdna-ssl.com/wp-content/uploads
Guest Password			

3. Assign the Apttus Ecommerce permission set to the user you created in Step 2. This grants access to the various objects and fields necessary to perform the actions within the storefront.
4. Add a remote site setting with your community URL as the approved domain.
5. Create storefront sharing rules to allow the guest user to read the username and password from the storefront object for the guest user.



Storefront Sharing Rules

Action	Criteria	Shared With	Access Level
Edit | Del	Storefront: Storefront Name EQUALS D-Commerce	E-Commerce Site Guest User	Read Only
Edit | Del	Storefront: Storefront Name EQUALS P-Commerce	partner Site Guest User	Read Only

Important

Create all the above-mentioned sharing rules for both Digital Commerce and Partner Commerce storefronts.

Digital Commerce Application Caching

The API Cache object that comes with the managed package stores record information such as Line Item Ids and Attributes metadata for quick loading thus improving performance.

The API Cache object caches the Query String and does not rebuild it again once it is cached. When changes are made in the metadata that is part of the query string, you should always clean the Api Cache object.

Cleaning the application cache was done by managed package but now it should be customer-specific unmanaged code

You must provide CRUD permission to the API Cache object.

Digital Commerce has a cache in place that stores dynamic queries so that the APIs do not have to rebuild them every time they are used. Every time an API is called, the default (50) cached records are pulled into the memory proactively to more intelligently speed up the API. There was no way to override the number of cached records.

A new admin setting named APTS_ApiCacheHydrateSize is introduced to override the default batch size.

Payment Integration with Cybersource Payment

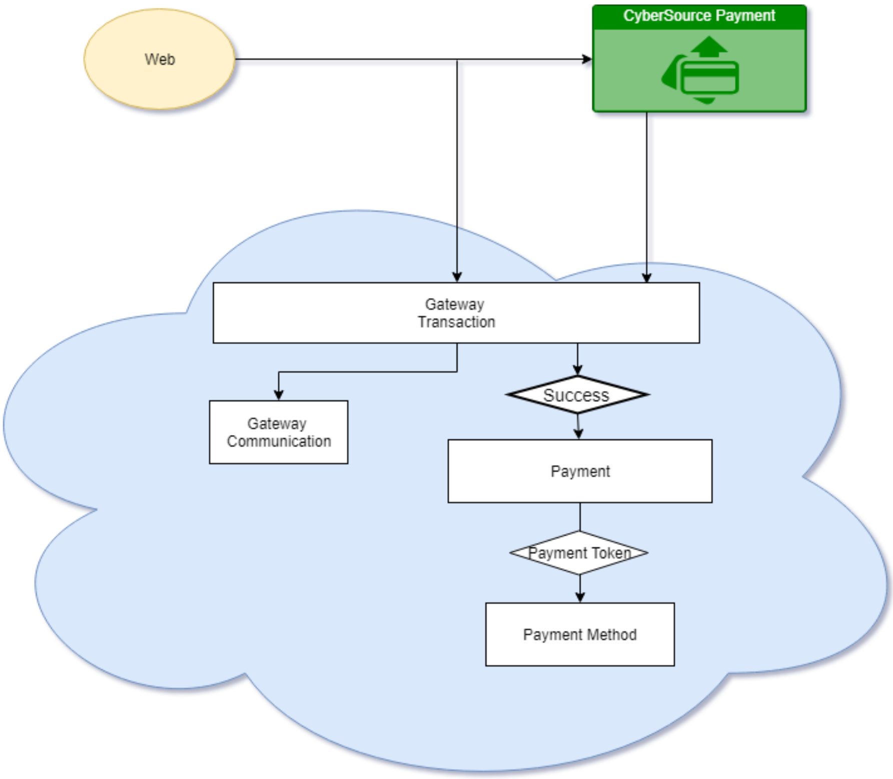
⚠ **Prerequisite**

You must have already installed the Apttus CPQ and Apttus Billing Management packages.

The following four Billing objects store record entries that are used by the Cybersource unmanaged package for every transaction.

Object Name	Purpose
Payment	When a transaction is successfully completed with a success message, a new entry is made in the Payment object with regards to the order.
Payment Method	This object stores the data for saving payment detail for future use.
Gateway Transaction	This object creates a new record every time a payment request to Cybersource is sent and updates the status when a response is received.
Gateway Communication	This is a child object of Gateway Transaction. This object creates a new entry for each request and response payload.

The following diagram illustrates the relationship of Salesforce objects:



To implement payment integration using Cybersource

1. Download and install the Cybersource unmanaged package from the repository. This package includes all the related objects, labels, classes, and permissions with some rest API resources. Based on these resources the payment integration is executed through classes.
2. You must create a Cybersource account with the following two Secure Acceptance profiles.
   a. Hosted API - This is used to display IFrame in your Digital commerce site.
   b. Checkout API - This is used to perform transactions by using payment method in a silent way.
   c. Set up Payment Integration using Cybersource.
3. Create profiles in the APTS Cybersource Profile Details custom setting. This custom setting is part of the unmanaged package.
4. Define custom labels for Hosted profile and Checkout/Silent Order Post profile.

Creating a Cybersource Account with the Secure Acceptance Method

Prerequisite

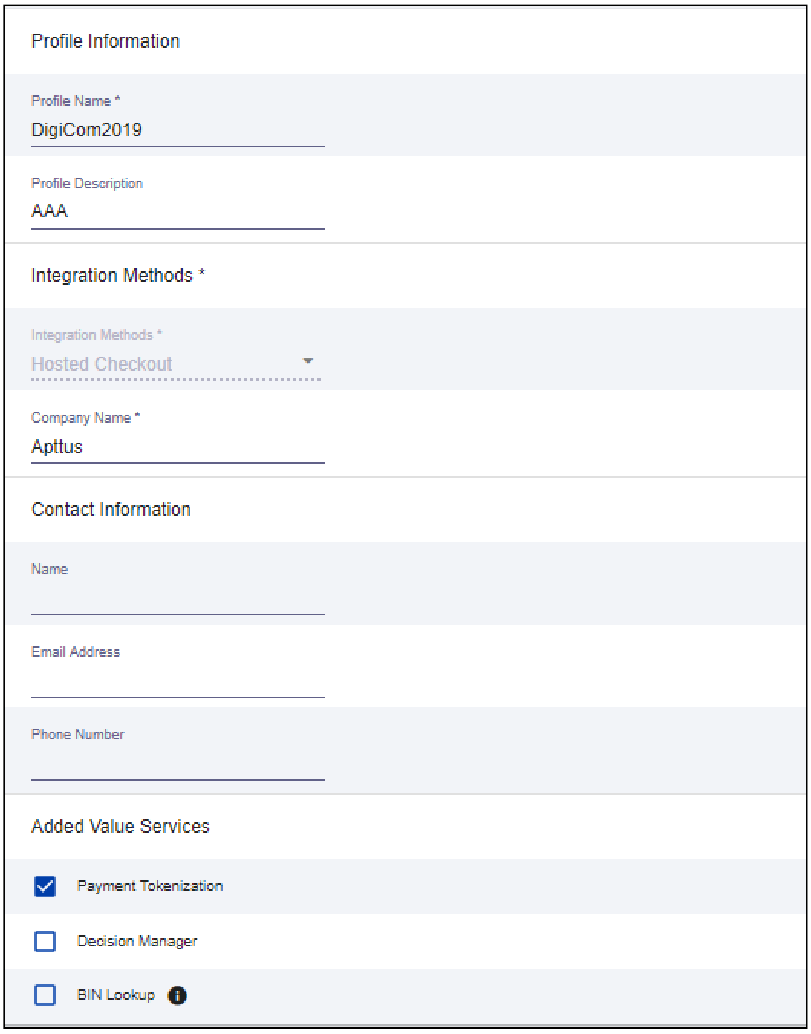
You must subscribe for a Cybersource subscription account.

To integrate Cybersource payment gateway with Apttus, you must create a Cybersource account and create your own profile under secure acceptance setting.

To create profiles under secure acceptance setting

1. Log in to your Cybersource account.
2. From the Dashboard, under Payment Configuration, select Secure Acceptance Setting.
3. From the Secure Acceptance Setting page, click New Profile. The Create Profile panel appears.
4. Enter the profile name and description, the company name, the integration method used to process notifications and the contact to whom they will be sent, and any additional services. This information is available in the General Information tab after you create the profile.

   ⚠ The integration method cannot be changed later.



Profile Information

Profile Name *
DigiCom2019

Profile Description
AAA

Integration Methods *

Integration Methods *
Hosted Checkout

Company Name *
Apttus

Contact Information

Name

Email Address

Phone Number

Added Value Services

☑ Payment Tokenization

☐ Decision Manager

☐ BIN Lookup

5. Click Submit. The Create Profile panel closes and the Edit Profile page appears.

The profile is created as "Inactive." You can complete profile values to promote it using the steps in Editing Secure Acceptance Profiles at any time.

To edit a secure acceptance profile

You must make changes to an inactive profile, then promote the changes to the active profile.

1. On the left navigation pane, click the Payment Configuration icon.
2. Click Secure Acceptance Settings. The Secure Acceptance Settings page appears.
3. Click the Inactive Profiles tab.
4. In the Profile Name column, click the name of the profile you want to edit. The Edit Profile page appears.
   You can also select an Active profile, and click the Edit icon. Business Center automatically takes you to the Inactive version of the profile.

5. Click the tab containing the information you want to update and make changes as necessary.

Tab Name	Description
General Information	Enter basic information about the name and format of the profile.
Payment Settings	Select accepted card types, checkout methods, and reversal preferences.
Security	Generate access key and security key that you need to pass for transaction.
Payment Form	Set up checkout steps, and which fields to include in billing and order review forms.
Notifications	Designate where to send transaction data.
Customer Response	Customize response and error messages.
Branding (Optional)	Add your branding logos, fonts, and colors.

6. When you are done, click Save.
7. Click the Promote Profile icon to add changes to the Active profile.

Defining Payment Methods

After creating a profile, configure payment methods.

1. From the Edit Profile page, go to the Payment Settings tab.
2. Click Add Card Types to add cards of your choice.

> ⚠ If you want to use Electronic Check (eCheck) or Pay Pal method, you must raise a support ticket first with Cybersource. Cybersource implements your requirement, post which eCheck and Pay Pal options are displayed. You can also select account types as per your requirement.

3. Click the setting icon next to the card type to add supported currencies and required fields. The card setting page appears.

> ⚠ The default currency is US Dollars (USD). To add more currencies, you must raise a support ticket with Cybersource.

4. Select the options of your choice and click Submit.

Creating Access and Security Key

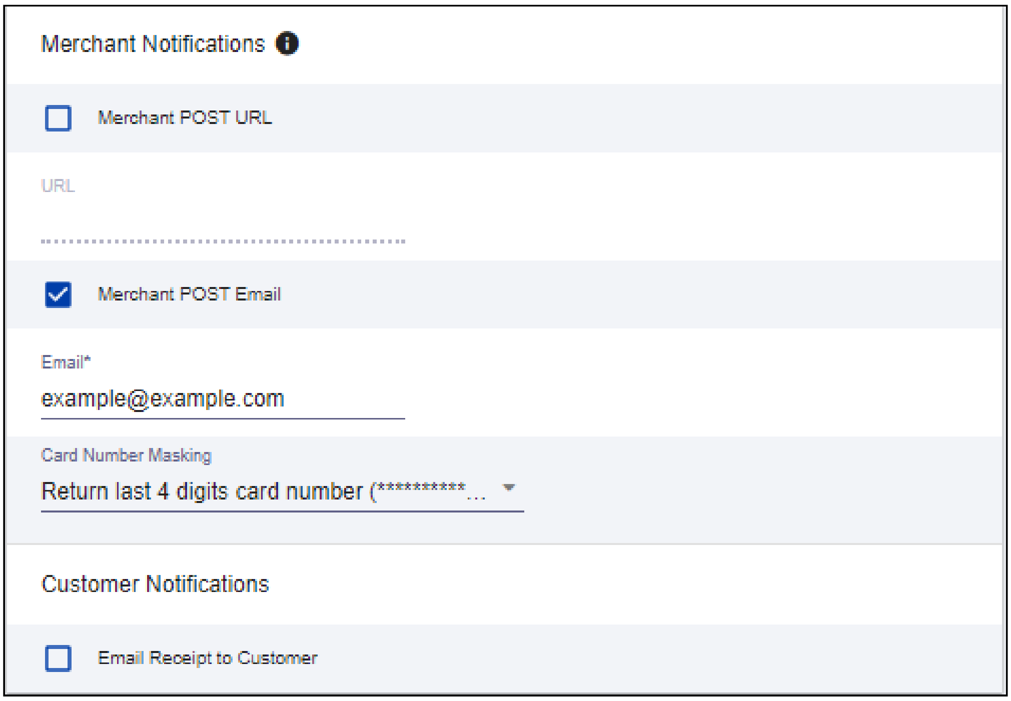
You must create your access key and security key that is needed for transactions.

1. From the Edit Profile page, go to the Security tab.
2. Click the Create icon.
3. Enter a Key Name and click Create.

An access key and security key is generated. You must enter the keys in the APTS_CyberSource_Profile_Details__c Cybersource custom Setting.

Configuring Transaction Responses

You can configure transaction responses from the Notification tab. You can define whether to receive responses through email or a post on API. Currently Email notifications is supported.



Merchant Notifications

□ Merchant POST URL

URL

☑ Merchant POST Email

Email*: example@example.com

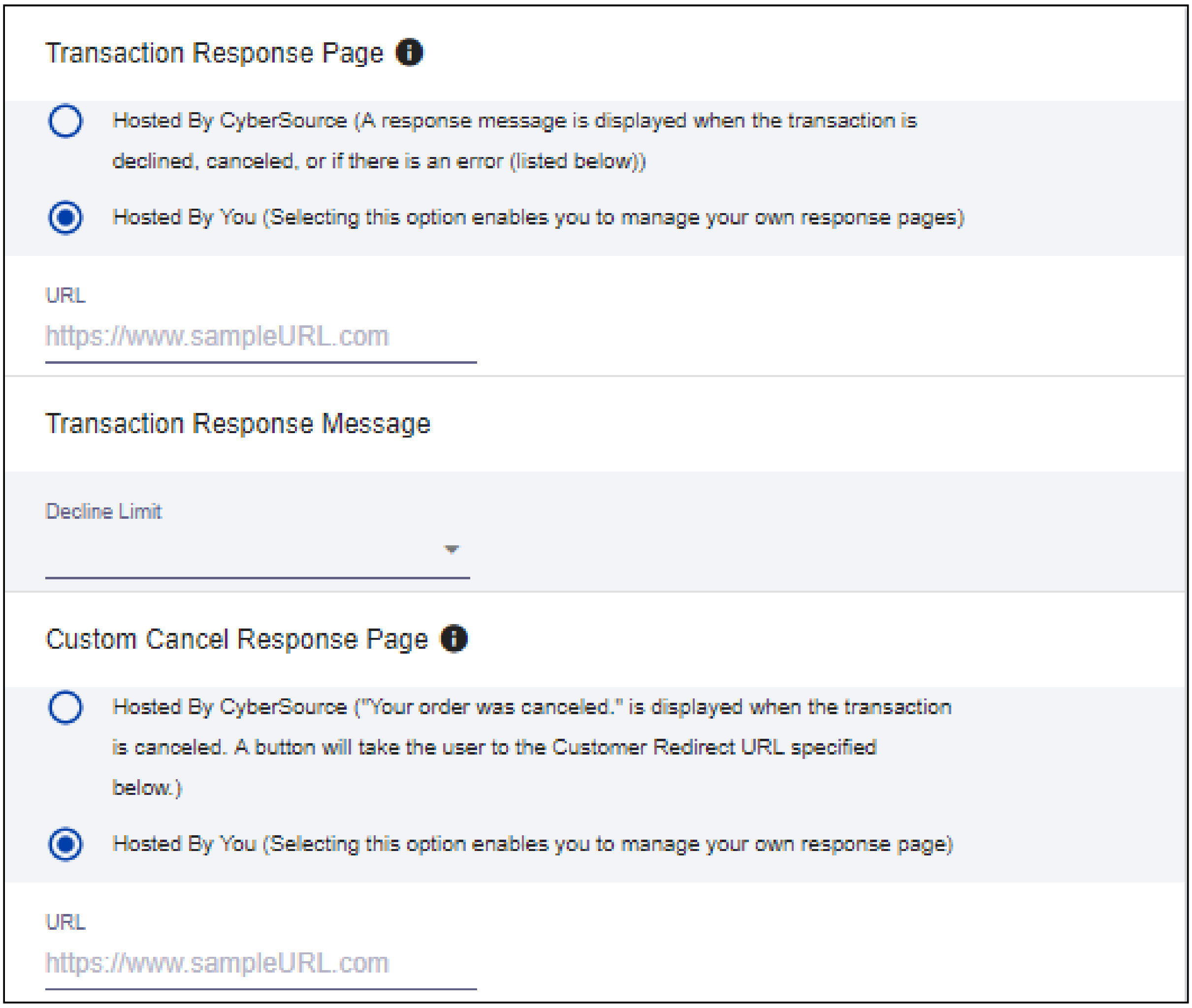
Card Number Masking: Return last 4 digits card number (**********...

Customer Notifications

□ Email Receipt to Customer

Configuring Customer Responses

Customer response setting allows you to configure the redirect page after the transaction is completed/canceled.



Transaction Response Page

○ Hosted By CyberSource (A response message is displayed when the transaction is declined, canceled, or if there is an error (listed below))

● Hosted By You (Selecting this option enables you to manage your own response pages)

URL
https://www.sampleURL.com

Transaction Response Message

Decline Limit

Custom Cancel Response Page

○ Hosted By CyberSource ("Your order was canceled." is displayed when the transaction is canceled. A button will take the user to the Customer Redirect URL specified below.)

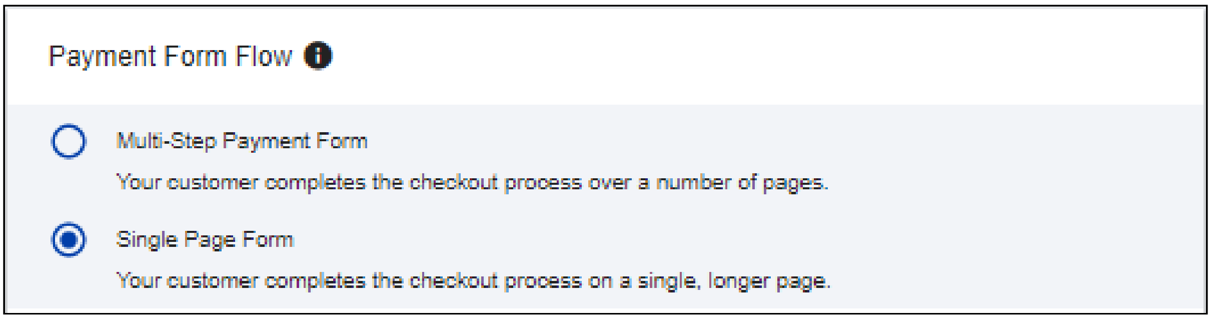
● Hosted By You (Selecting this option enables you to manage your own response page)

URL
https://www.sampleURL.com

Configuring The Payment Form

You must configure your cybersource secure acceptance page from the Payment Form tab.

1. From the Edit Profile page, go to the Payment Form tab.
2. Select the Payment Form Flow.

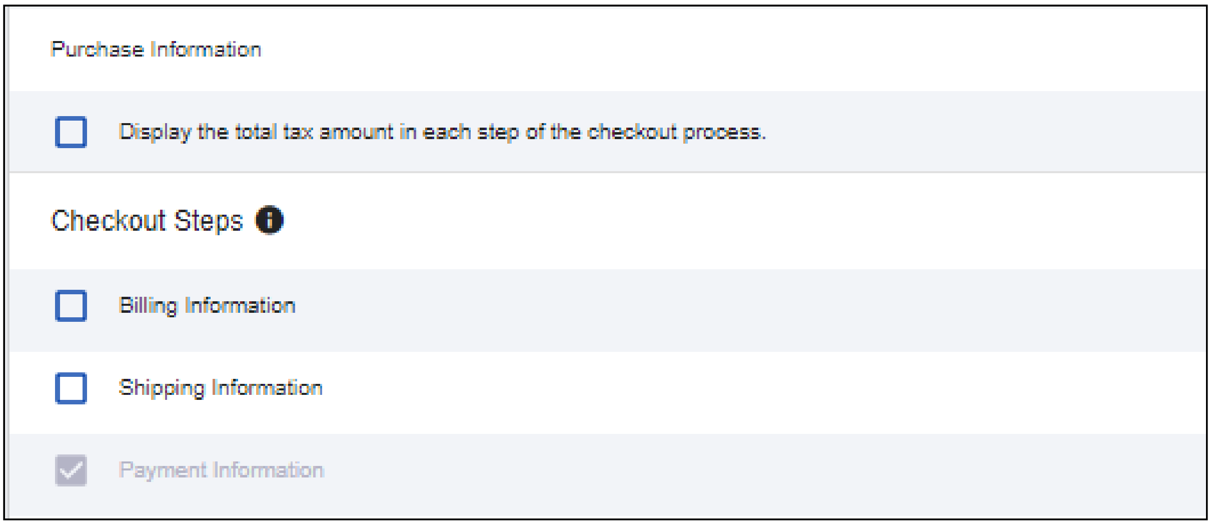


Payment Form Flow

○ Multi-Step Payment Form
Your customer completes the checkout process over a number of pages.

● Single Page Form
Your customer completes the checkout process on a single, longer page.

3. Configure payment and checkout page information.



Purchase Information

□ Display the total tax amount in each step of the checkout process.

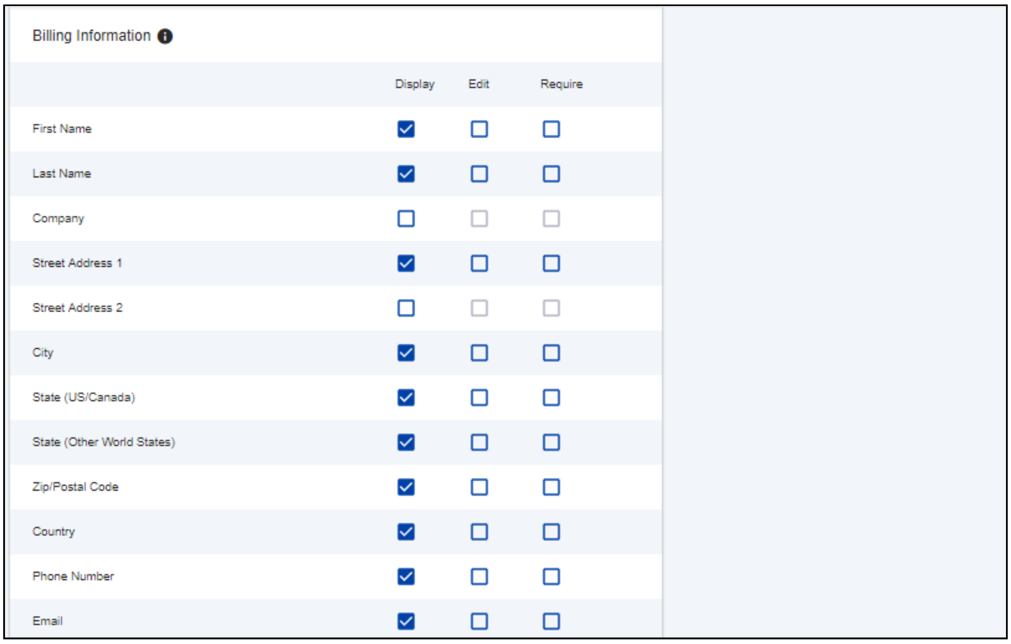
Checkout Steps

□ Billing Information

□ Shipping Information

☑ Payment Information

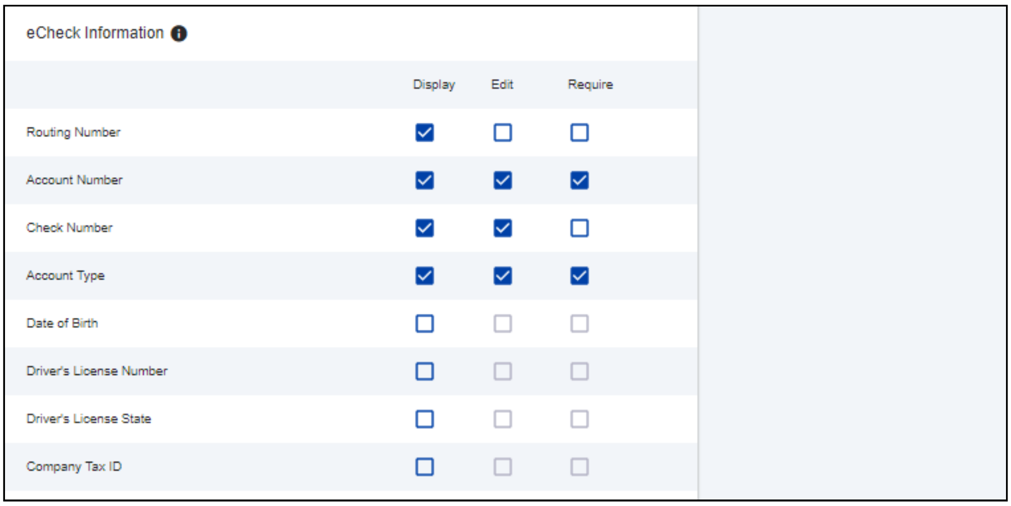
4. Select relevant information to display on billing page.



Billing Information

	Display	Edit	Require
First Name	☑	□	□
Last Name	☑	□	□
Company	□	□	□
Street Address 1	☑	□	□
Street Address 2	□	□	□
City	☑	□	□
State (US/Canada)	☑	□	□
State (Other World States)	☑	□	□
Zip/Postal Code	☑	□	□
Country	☑	□	□
Phone Number	☑	□	□
Email	☑	□	□

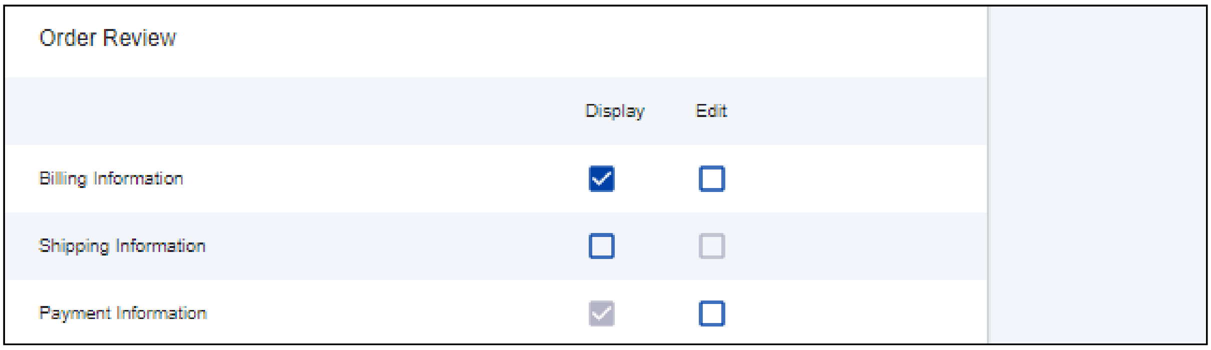
5. Select relevant information to display on echeck page.



eCheck Information

	Display	Edit	Require
Routing Number	☑	☐	☐
Account Number	☑	☑	☑
Check Number	☑	☑	☐
Account Type	☑	☑	☑
Date of Birth	☐	☐	☐
Driver's License Number	☐	☐	☐
Driver's License State	☐	☐	☐
Company Tax ID	☐	☐	☐

6. Configure the settings for review page.



Order Review

	Display	Edit
Billing Information	☑	☐
Shipping Information	☐	☐
Payment Information	☑	☐

7. Click Save.

Creating Profiles in Custom Settings

Now that Cybersource settings is complete, you must create profiles in the APTS CyberSource Profile Details custom settings. The Cybersource unmanaged package includes Cybersource custom setting named APTS_CyberSource_Profile_Details__c. This Custom Setting stores the access_Key, Secret_Key, and Profile ID from Payments. You must create two records in the custom settings, one for hosted profile and another for checkout profile.

To create hosted profile in custom setting

1. Go to Setup > Develop > Custom Settings and click Manage for APTS CyberSource Profile Details.
2. Click New.
3. In Name, enter the name for the profile. For example: System Properties.
4. In Access Key, enter the access key that you generated in the Security tab in Cybersource.
5. In Ifram URL, enter the URL provided by Cybersource during subscription.
    - For Test Transactions, you may use: https://testsecureacceptance.cybersource.com/embedded/pay
    - For Live Transactions you may use: https://secureacceptance.cybersource.com/embedded/pay or the URL provided by Cybersource.
6. In Merchant ID, enter the merchant ID. The merchant ID is same as the Organization ID that you used to create the Cybersource account.
7. In Profile ID, enter the profile ID. You can get the Profile ID as soon as you create a secure acceptance profile.
8. In Secret Key1 and Secret Key2, enter the secret key that you generated in the Security tab in Cybersource.

    > ⚠ Due to Salesforce limitations of 250 characters, this secret key is divided into two fields. You must manually break the key into two and enter in these fields.

9. If you are using a test environment, select Is TestEnvironment.

    > ⚠ Checking this flag sets the default order amount to $100 as Cybersource may display an error for bigger amount in test environments.

10. Click Save.

To create checkout profile in custom setting

1. Go to Setup > Develop > Custom Settings and click Manage for APTS CyberSource Profile Details.
2. Click New.
3. In Name, enter the name for the profile. For example: Silent Checkout Profile.
4. In Access Key, enter the access key that you generated in the Security tab in Cybersource.
5. In Ifram URL, enter the URL provided by Cybersource during subscription.

- For Test Transactions, you may use: https://testsecureacceptance.cybersource.com/silent/embedded/pay
- For Live Transactions you may use: https://secureacceptance.cybersource.com/silent/embedded/pay or the URL provided by Cybersource.

a. In Merchant ID, enter the merchant ID. The merchant ID is same as the Organization ID that you used to create the Cybersource account.

6. In Profile ID, enter the profile ID. You can get the Profile ID as soon as you create a secure acceptance profile.
7. In Secret Key1 and Secret Key2, enter the secret key that you generated in the Security tab in Cybersource.

   > ⚠ Due to Salesforce limitations of 250 characters, this secret key is divided into two fields. You must manually break the key into two and enter in these fields.

8. If you are using a test environment, select Is TestEnvironment.

   > ⚠ Checking this flag sets the default order amount to $100 as Cybersource may display an error for bigger amount in test environments.

9. Click Save.

To configure the callback URL for Payment Iframe in custom setting

1. Go to Setup > Develop > Custom Settings and click Manage for APTS CyberSource Profile Details.
2. Click System Properties.
3. In Payment Callback URL, enter the URL to redirect the customer in the Digital Commerce application after payment is done.
4. Click Save.

Defining Custom Labels

The Cybersource unmanaged package comes with two labels where you must enter the custom setting profile you created. These labels use relevant values while performing payment transactions.

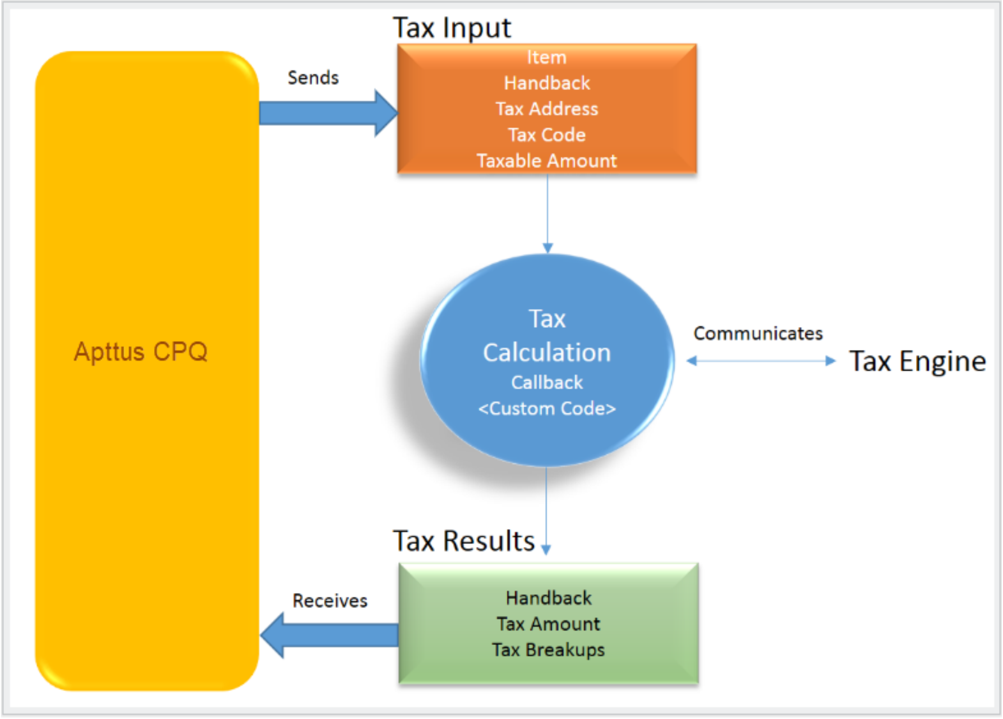
To define custom labels

1. Go to Setup > Create > Custom Labels and click Edit for cyberSource_Active_CustomSetting.
2. Enter a short description.
3. In Value, enter the name for the hosted profile you created in the APTS CyberSource Profile Details custom setting. For example: System Properties.
4. Click Save.
5. Now to go to Setup > Create > Custom Labels and click Edit for cyberSource_Active_Silent_CustomSetting.
6. In Value, enter the name for the checkout profile you created in the APTS CyberSource Profile Details custom setting. For example: Silent Checkout Profile.
7. Click Save.

Tax Integration with Avalara Tax Engine

You can communicate with a Tax Engine of your choice with the help of a Tax Callback class. For this, you should know the Request Fields that are sent to Callback and further to the Tax Engine. Following diagram gives an overview of how information is sent from Product to the Tax Engine and received back after processing.

Apttus Tax Engine



Tax input is a container which holds the following fields:

- **Item:** Contains an Invoice Line Item or a Credit Memo Line Item.
- **Handback**: A generic wrapper class that can be used to pass an additional field value. Set the value for this field to *TaxInputRelatedObjects*. This class contains the parent Invoice or Credit Memo based on whether the item contains an Invoice Line Item or a Credit Memo Line Item.
- **Tax Address**: The address specified as the Shipping Address of the Ship To account. If there is no Shipping Address mentioned in the Ship To account, then the Billing Address of the Ship To account is used.
- **Tax Code:** This value is taken from the product PLI.
- **Taxable Amount**: This is the amount to apply the tax to.

When communicating with a Tax Engine, you must note that:

- The implementation of the Tax Callback must return a *TaxResultHandback* object in the *Handback* field of a Tax Result.
- The implementation of the Tax Callback must determine a *commit mode* by checking if the status of the Invoice is *Approved Pending.*

You must register a **Tax Callback class** which is called for tax calculation on Invoice generation.

Setting Up Tax Integration

⚠ **Prerequisite**

You must have already installed Apptus CPQ packages.

To set up tax integration, you must complete the following tasks:

- Install the tax integration package and define the class in custom settings
- Enable tax calculations
- Create Tax Code and Tax Certificate
- Set up Tax Calculations
- Set up Tax Breakups

To set up tax integration with Avalara tax engine

1. Download and install Avalara Tax Integration with Ecommerce unmanaged package from the repository.
2. Go to Custom Settings and click Manage for Config Custom Classes.
3. Click New.
4. In Name, enter System Properties.
5. In Tax Callback Class, enter APTS_APTSTaxCallBack. This class is part of the Avalara Tax Integration with Ecommerce unmanaged package.
6. Click Save.

To enable tax calucalations

1. Go to the Storefronts tab. Choose and click your storefront name to open your storefront record.
2. Click Edit and set the Enable Tax Calculations flag to True.
3. Click Save.

In Apttus Digital Commerce at the time of Quote creation or Order generation tax needs to be calculated for taxable line items or order line items. You must create a Tax Code and a Tax Certificate record.

To create tax code and tax certificate record

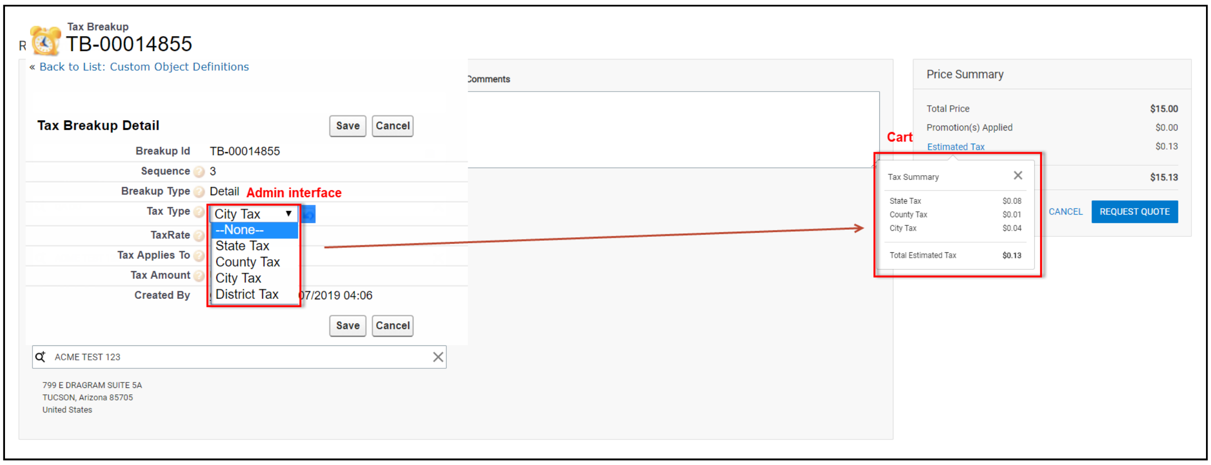
1. Go to the Tax Code tab and click New.
2. In Name, enter a name for the tax code.
3. In Code, enter the tax code provided by Avalara.
4. Enter a meaningful Description for the tax code.
5. Click Save. You can define this code in your price list item.
6. Go to the Tax Certificate tab and click New.
7. Enter a Number for the tax certificate.
8. Enter a meaningful Description for the tax certificate.
9. Select the Effective Date and Expiration Date for the tax certificate.
10. Click Save. You can define the newly created tax certificate in your Account object.

By default, not all products are taxable. To make a product taxable, you must set the Taxable flag to True and enter the tax code on the Price List Item.

To set up tax calculation

1. Go to the Price List Item for line item you want to calculate tax for.
2. Set the Taxable flag to True and enter the Tax Code.
3. Click Save.

In Apttus Digital Commerce, when you select a product that is taxable and you request a quote, the system calculates the tax and displays into the cart. The breakup of the tax is defined in Salesforce from the line item level. The percentage of the tax break up is passed on from Avalara. The mechanism here is that the API picks account address and passes it to Avalara. Based on the region/address, Avalara passes tax percentage information which is then computed in Salesforce and displayed in the Tax Breakup object. Refer to the image below for the type for tax breakups populated on the Digital Commerce site.



Tax Breakup
TB-00014855
« Back to List: Custom Object Definitions

Tax Breakup Detail

Save Cancel

Breakup Id TB-00014855
Sequence 3
Breakup Type Detail
Tax Type City Tax
TaxRate
Tax Applies To
Tax Amount
Created By 07/2019 04:06

Save Cancel

ACME TEST 123

799 E DRAGRAM SUITE 5A
TUCSON, Arizona 85705
United States

Comments

Price Summary

Total Price	$15.00
Promotion(s) Applied	$0.00
Estimated Tax	$0.13
	$15.13

Tax Summary

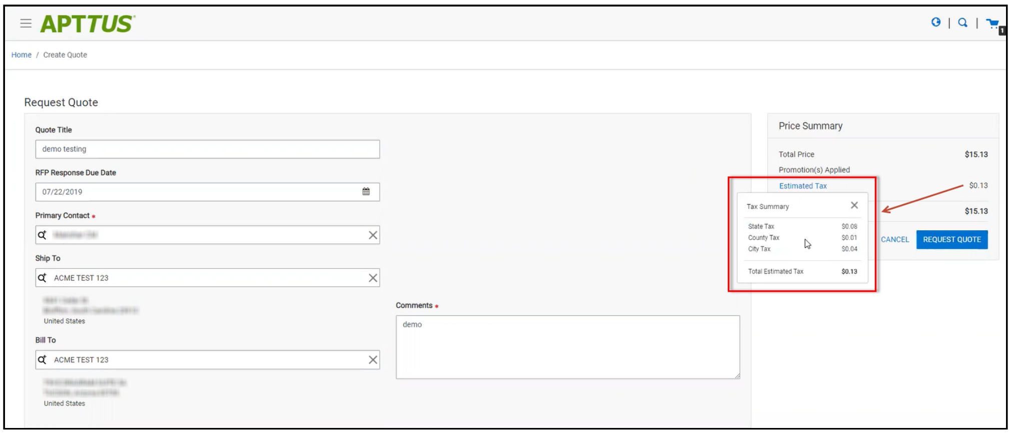
State Tax	$0.08
County Tax	$0.01
City Tax	$0.04
Total Estimated Tax	$0.13

Now when the order is generated, you get order tax breakup for orders and proposal tax break up for proposals. These tax breakups are attached to respective line items.

You must ensure that the Account object has a valid Billing and Shipping Address and the Tax certificate is defined.

Use Case:

1. From the Product Catalog, select a Product and add it to the cart.
2. Click Request Quote. The Request Quote page appears.
   Tax for the selected product is calculated and displayed.
3. To view the tax break up, click Estimated Tax.



Home / Create Quote

Request Quote

Quote Title
demo testing

RFP Response Due Date
07/22/2019

Primary Contact *
[redacted]

Ship To
ACME TEST 123
[redacted]
United States

Bill To
ACME TEST 123
[redacted]
United States

Comments *
demo

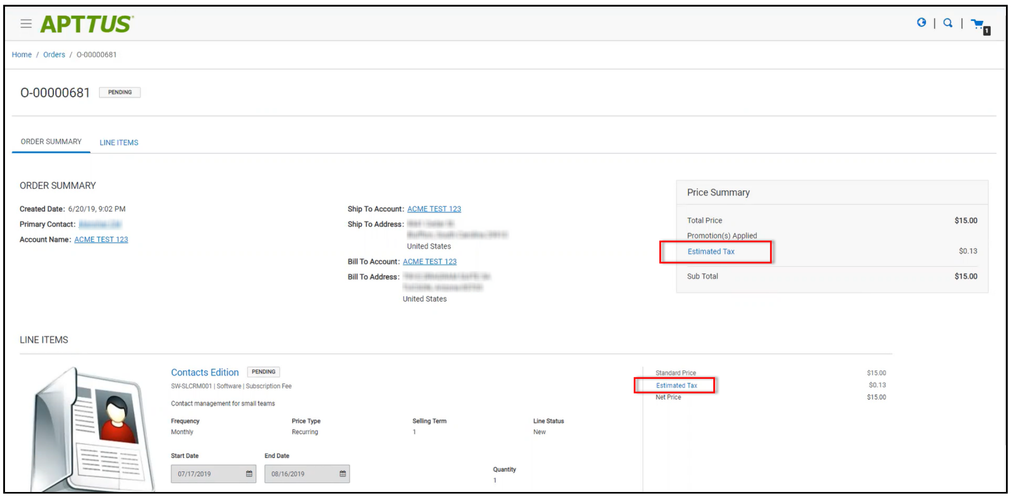
Price Summary

Total Price	$15.13
Promotion(s) Applied	
Estimated Tax	$0.13
	$15.13

Tax Summary

State Tax	$0.08
County Tax	$0.01
City Tax	$0.04
Total Estimated Tax	$0.13

4. You can either Place an Order or Request a Quote. If you Request a Quote, you can view the tax summary from My Accounts > Quotes. If you place an Order, you can view the tax summary from My Accounts > Orders.



Home / Orders / O-00000681

O-00000681 PENDING

ORDER SUMMARY LINE ITEMS

ORDER SUMMARY

Created Date: 6/20/19, 9:02 PM
Primary Contact: [redacted]
Account Name: ACME TEST 123

Ship To Account: ACME TEST 123
Ship To Address: [redacted]
United States
Bill To Account: ACME TEST 123
Bill To Address: [redacted]
United States

Price Summary

Total Price	**$15.00**
Promotion(s) Applied	
Estimated Tax	$0.13
Sub Total	**$15.00**

LINE ITEMS

Contacts Edition PENDING
SW-SLCRM001 | Software | Subscription Fee
Contact management for small teams

Frequency	Price Type	Selling Term	Line Status
Monthly	Recurring	1	New

Start Date 07/17/2019
End Date 08/16/2019
Quantity 1

Standard Price	$15.00
Estimated Tax	$0.13
Net Price	$15.00

Setting Up Multi-Language Using Translations

You can set up multi-language support for your Digital Commerce site. Apttus provides you three ways to achieve this:

- Translations from the SDK
- Translations from Salesforce
- Translations using Translation Workbench

Translation can be set up for the following:

- Static Labels or text (Page Titles, Header, Footer, Tabs, Descriptions)
- Field Label translations
- Data translations (product, category, product attribute group)

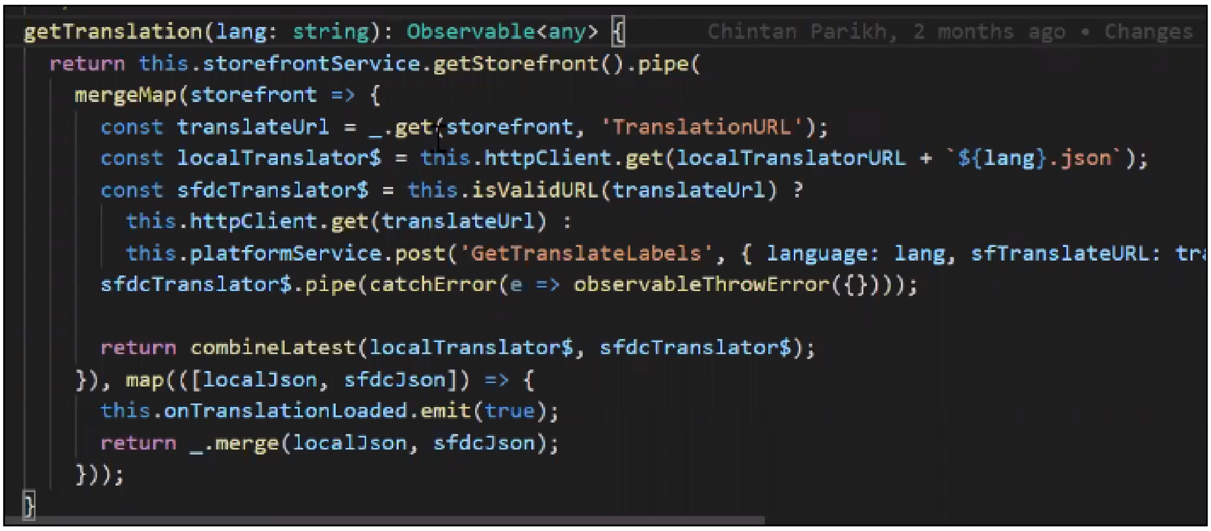
The default language is English. You can set a different language from the User Profile.

To set a default language for a user

1. Go to Setup > Administration Setup - Managed Users and click Users.
2. Click Edit next to the User.
3. In Locale, select a locale. For example: Spanish (Mexico).
4. In Language, select your preferred language for the user. For example: Spanish.
5. Click Save.

Setting Up Translations using SDK

The Apttus E-Commerce package consists of a Translator Loader. This Translator loader contains a folder named assets > i18n that consists json files used to define translations. Be default, only English language (en_US.json) is packaged with the base template. In order to add more languages, you can clone the English language json file and modify it to accommodate your preferred language. In the translator-loader.service.ts file, the translation get method checks for the translation URL from the Storefront record. Priority is set to check if the Translation URL is defined on the Storefront object. If not, the system checks for the files in the SDK. The system selects the URL from the local json files. If both, the Storefront and the SDK, do not have the URL, the system sets the default language as English. Refer to the get method code snippet. If the translation URL is not present on the Storefront object, the system uses SDK to translate labels, fields, and data.



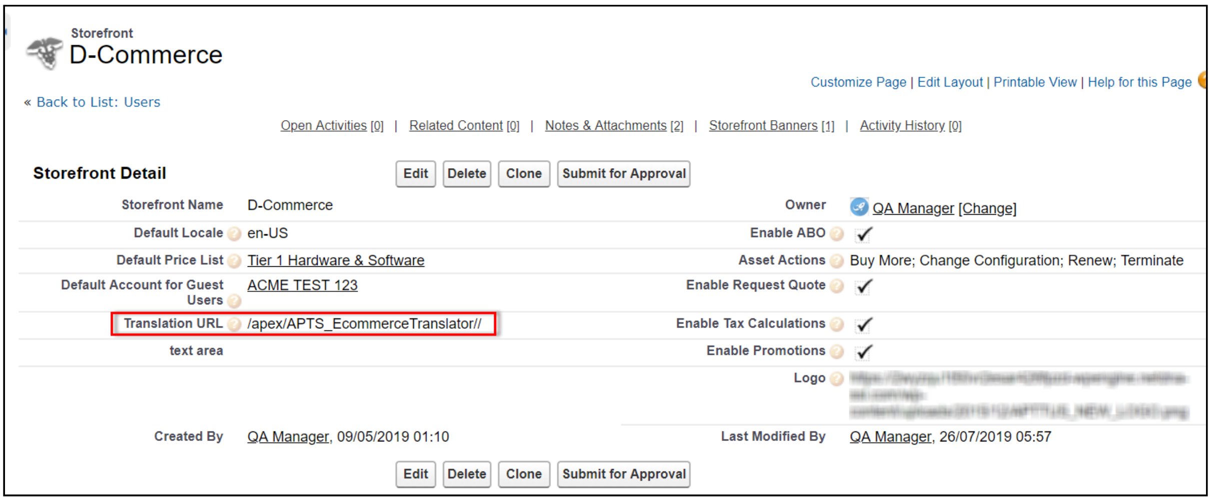
```
getTranslation(lang: string): Observable<any> {       Chintan Parikh, 2 months ago • Changes
  return this.storefrontService.getStorefront().pipe(
    mergeMap(storefront => {
      const translateUrl = _.get(storefront, 'TranslationURL');
      const localTranslator$ = this.httpClient.get(localTranslatorURL + `${lang}.json`);
      const sfdcTranslator$ = this.isValidURL(translateUrl) ?
        this.httpClient.get(translateUrl) :
        this.platformService.post('GetTranslateLabels', { language: lang, sfTranslateURL: tr
      sfdcTranslator$.pipe(catchError(e => observableThrowError({})));

      return combineLatest(localTranslator$, sfdcTranslator$);
    }), map(([localJson, sfdcJson]) => {
      this.onTranslationLoaded.emit(true);
      return _.merge(localJson, sfdcJson);
    }));
}
```

Setting Up Translations Using Salesforce

The translation URL must be present on the Storefront record. Go to the Storefront record > Translation URL = /apex/APTS_EcommerceTranslation. This URL points to a static page that contains all the translations. See example code snippet.

Example Code Snippet for E-Commerce Translation

```
<apex:page contentType="application/json; charset=utf-8" language="{!
$CurrentPage.parameters.language}">

{
  "ERROR": {
    "MIN-OPTIONS": "{!$Label.eCom_MIN_OPTIONS}",
    "CART": {
      "TOO_MANY_ATTEMPTS": "{!$Label.ecom_TOO_MANY_ATTEMPTS}",
      "PRICE_CHANGE": "{!$Label.eCom_PRICE_CHANGE}",
      "PRICE_CHANGE_TOASTR_TITLE": "{!$Label.eCom_PRICE_CHANGE_TOASTR_TITLE}"
    },
    "APPLICATION_ERROR_TOASTR_TITLE": "{!$Label.eCom_APPLICATION_ERROR_TOASTR_TITLE}"
  },
  "SUCCESS": {
    "CART": {
      "ITEM_ADDED_TOASTR_MESSAGE": "{!$Label.eCom_ITEM_ADDED_TOASTR_MESSAGE}",
      "ITEM_ADDED_TOASTR_TITLE": "{!$Label.eCom_ITEM_ADDED_TOASTR_TITLE}"
    }
  },
  "ASSETS": {
    "CHANGECONFIGURATION_SUCCESS": "{!
$Label.eCom_Assets_ChangeConfigurationSuccess}",
    "CHANGECONFIGURATION_START_DATE": "{!
$Label.eCom_Assets_ChangeConfigurationStartDate}",
    "CHANGECONFIGURATION_END_DATE": "{!
$Label.eCom_Assets_ChangeConfigurationEndDate}"
  },
  "MY_ACCOUNT": {
    "LAST_LOGIN_DATE": "{!$Label.eCom_LAST_LOGIN_DATE}"
  },
  "BUTTON": {
    "Change Configuration": "{!$Label.eCom_Button_ChangeConfiguration}"
  },
  "PRODUCT_CARD": {
    "INSTALLED_PRODUCT": "{!$Label.eCom_PRODUCT_CARD_INSTALLED_PRODUCT}",
    "STANDARD_PRICE": "{!$Label.eCom_PRODUCT_CARD_STANDARD_PRICE}"
  },
  "PAGINATION": {
    "FIRST": "{!$Label.eCom_PAGINATION_FIRST}"
  },
  "COMMON": {
    "ORDERS": "{!$Label.eCom_COMMON_ORDERS}",
    "USERNAME": "{!$Label.eCom_COMMON_USERNAME}",
    "QUANTITY": "{!$Label.eCom_COMMON_QUANTITY}",
    "CHANGE_CONFIGURATION": "{!$Label.eCom_Common_ChangeConfiguration}",
    "ADD_TO_CART": "{!$Label.eCom_Add_to_Cart}"
  },
  "CONSTRAINT_POPOVER": {
    "PRODUCTS_INCLUDED_EXCLUDED_HEADING": "{!
$Label.eCom_PRODUCTS_INCLUDED_EXCLUDED_HEADING}"
  },
  "CART": {
    "PAYMENT": {
      "PAYMENT_TITLE": "{!$Label.eCom_CART_PAYMENT_TITLE}"
    },
    "CART_SUMMARY": {
      "QUANTITY": "{!$Label.eCom_CART_SUMMARY_QUANTITY}"
    },
    "CHECKOUT": "{!$Label.eCom_CHECKOUT}",
    "BILLING_AND_SHIPPING_INFORMATION": "{!
$Label.eCom_BILLING_AND_SHIPPING_INFORMATION}"
  },
  "MANAGE_CART": {
    "CART_SUMMARY": {
      "CART_SUMMARY_TITLE": "{!$Label.eCom_Cart_Summary_Title}",
      "SUB_TOTAL": "{!$Label.eCom_Sub_Total}"
    },
    "CART_TABLE": {
      "ITEMS_IN_YOURCART": "{!$Label.eCom_ITEMS_IN_YOUR_CART}"
    }
  },
  "FOOTER": {
    "PRODUCTS": "{!$Label.eCom_FOOTER_PRODUCTS}"
  },
  "HEADER": {
    "SUBMIT": "{!$Label.eCom_HEADER_SUBMIT}",
    "ENTER_YOUR_SEARCH_TERM": "{!$Label.eCom_ENTER_YOUR_SEARCH_TERM}",
    "LAST_LOGIN": "{!$Label.eCom_HEADER_LAST_LOGIN}",
    "HOME": "{!$Label.eCom_HEADER_HOME}",
    "LOG_OUT": "{!$Label.eCom_HEADER_LOG_OUT}",
    "LOG_IN": "{!$Label.eCom_HEADER_LOG_IN}"
  },
  "PROMOTION": {
    "PROMO_CODE": "{!$Label.eCom_PROMOTION_PROMO_CODE}",
    "PROMOTION_APPLIED": "{!$Label.eCom_PROMOTION_APPLIED}",
    "APPLIED_PROMOTION": "{!$Label.eCom_APPLIED_PROMOTION}"
  },
  "LOGIN": {
    "USERNAME": "{!$Label.eCom_LOGIN_USERNAME}",
    "SIGN_IN": "{!$Label.eCom_LOGIN_SIGN_IN}",
    "INCORRECT_CREDENTIALS_TOASTR_MESSAGE": "{!
$Label.eCom_INCORRECT_CREDENTIALS_TOASTR_MESSAGE}"
  },
  "PRODUCT_DETAILS": {
    "PRODUCT_DETAIL": "{!$Label.eCom_PRODUCT_DETAIL}",
    "PRODUCT_CODE": "{!$Label.eCom_PRODUCT_CODE}",
    "STANDARD_PRICE": "{!$Label.eCom_PRODUCT_DETAILS_STANDARD_PRICE}",
    "UPDATE_CONFIGURATION": "{!$Label.eCom_PRODUCT_DETAILS_UPDATE_CONFIGURATION}"
  },
  "INSTALLED_PRODUCTS": {
    "PRODUCT_FAMILY": "{!$Label.eCom_INSTALLED_PRODUCTS_PRODUCT_FAMILY}"
  },
  "MINI_CART": {
    "YOUR_CART_IS_EMPTY": "{!$Label.eCom_MINI_CART_YOUR_CART_IS_EMPTY}"
  }
}
</apex:page>
```


Storefront
D-Commerce

Customize Page | Edit Layout | Printable View | Help for this Page

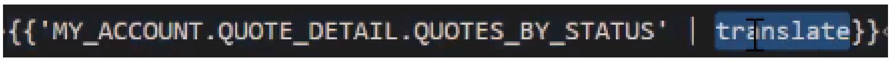
« Back to List: Users

Open Activities [0] | Related Content [0] | Notes & Attachments [2] | Storefront Banners [1] | Activity History [0]

Storefront Detail

Storefront Name	D-Commerce	Owner	QA Manager [Change]
Default Locale	en-US	Enable ABO	✓
Default Price List	Tier 1 Hardware & Software	Asset Actions	Buy More; Change Configuration; Renew; Terminate
Default Account for Guest Users	ACME TEST 123	Enable Request Quote	✓
Translation URL	/apex/APTS_EcommerceTranslator//	Enable Tax Calculations	✓
text area		Enable Promotions	✓
		Logo	[redacted]
Created By	QA Manager, 09/05/2019 01:10	Last Modified By	QA Manager, 26/07/2019 05:57

Wherever you are displaying a label on the UI, a translation pipe is used.



```
{{'MY_ACCOUNT.QUOTE_DETAIL.QUOTES_BY_STATUS' | translate}}
```

This translation pipe calls the translation loader service, which uses the translation get method to look for the translation URL whether it is in the code base or defined on the storefront object. The translation get method looks at the URL, reads the file, gets all the key values and displays the response on the template.

Defining translations on Salesforce using Custom Labels

For the translation URL defined on the Storefront record, you must define custom labels.

To define custom labels

1. Go to Setup > App Setup > Create and click Custom Labels.
2. Click New Custom Label.
3. Enter a Short Description for the custom label you are creating.
4. Enter a Name for the custom label.
5. The default language is English.
6. In Categories, enter text to categorize the label.
7. In the Value text box, enter text. This value can be translated into any language that Salesforce supports.
8. Click Save.

Translations for custom labels determine what text to display for the label’s value when a user’s default language is the translation language.

1. Go to Setup > App Setup > Create and click Custom Labels.
2. Select the name of the custom label to open.
3. In the Translations related list, click New to enter a new translation or Edit next to the language to change a translation.
4. Select the Language you are translating into.
5. Enter the Translation Text. This text overrides the value specified in the label's Value field when a user's default language is the translation language.
6. Click Save.

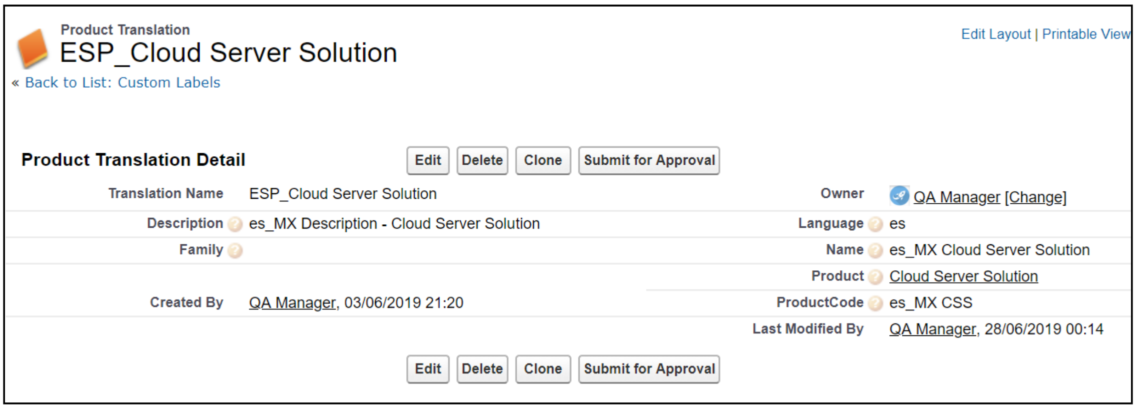
Defining translations for custom objects

Salesforce supports Product Translations, Category Translations, and Attribute Group Translations.

To define translations for Products

1. Go to the Product Translations tab and click New. - OR - Go to the Product tab, select a product for which you want to define translation, and from the Product Translations related list, click New Product Translations.
2. Enter a Translation Name.
3. Enter translated text for the following fields:
    - Description
    - Language
    - Name
    - Product
    - ProductCode

> ⚠ Currently, Family is not supported.



Product Translation

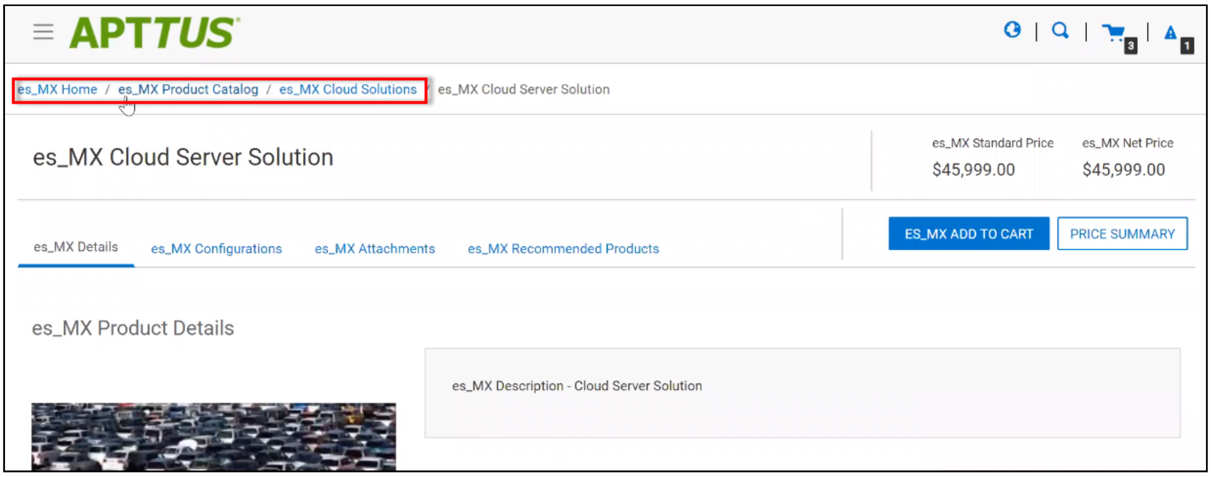
ESP_Cloud Server Solution

« Back to List: Custom Labels

Product Translation Detail

Translation Name	ESP_Cloud Server Solution	Owner	QA Manager [Change]
Description	es_MX Description - Cloud Server Solution	Language	es
Family		Name	es_MX Cloud Server Solution
		Product	Cloud Server Solution
Created By	QA Manager, 03/06/2019 21:20	ProductCode	es_MX CSS
		Last Modified By	QA Manager, 28/06/2019 00:14

Now when you browse a product on the cart for which you have defined translations in Salesforce, the translated text is displayed.



APTTUS
3
1
es_MX Home / es_MX Product Catalog / es_MX Cloud Solutions / es_MX Cloud Server Solution

es_MX Cloud Server Solution

es_MX Standard Price	es_MX Net Price
$45,999.00	$45,999.00

es_MX Details
es_MX Configurations
es_MX Attachments
es_MX Recommended Products
ES_MX ADD TO CART
PRICE SUMMARY

es_MX Product Details

es_MX Description - Cloud Server Solution

⚠ For illustration purposes, the translated categories are prefixed with es_ in the above image. In the actual scenario, the correct language is displayed.

Similarly, you can define category translations.

To define translations for Categories

1. Go to the Product tab, select a product for which you want to define the category translation.
2. Go to the Categories related list and click the category classification Id.
3. From the Product Classification page, click the Category.
   - OR -
4. Go to the Categories tab and select the Category for which you want to define translations.
5. From the Category Translations related list, click New Category Translation.
6. Enter a Category Translation Name.
7. Search and select the Category Hierarchy.
8. Enter a Language and a Label.
9. Select a currency.
10. Click Save.

To define translations for Product Attribute Group

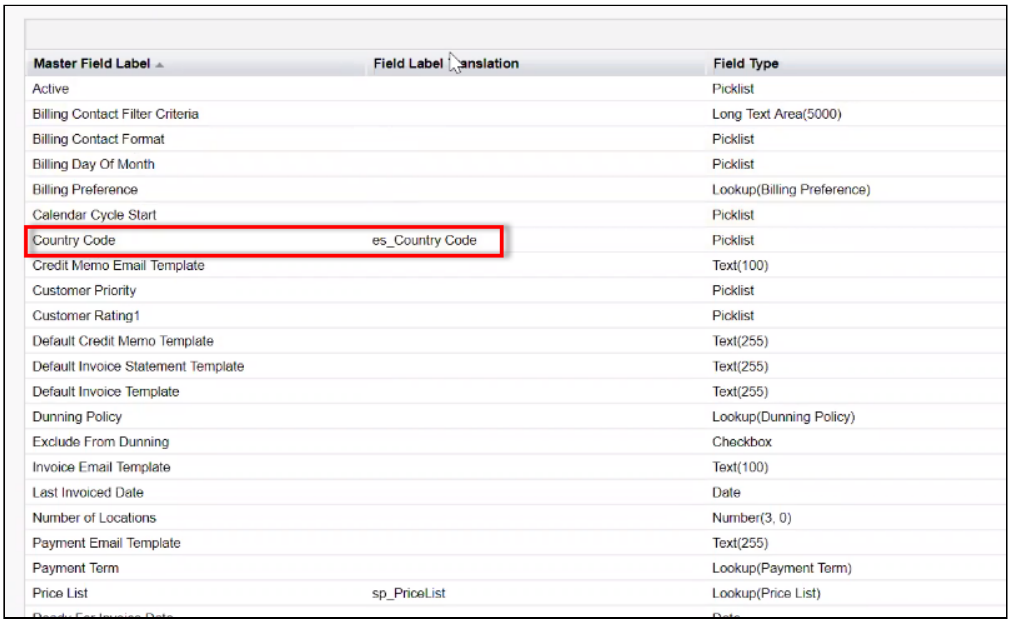
1. Go to the Product Attribute Groups tab and select a product attribute group for which you want to define the attribute group translation.
2. Go to the Attribute Group Translations related list and click New Attribute Group Translation.
3. Enter a Translation Name.
4. The Product Attribute Group field is already populated.
5. In Name, enter the translated attribute group name.
6. Enter a Language.
7. Select a currency.
8. Click Save.

Setting up Translations using Translation Workbench

You can also translate fields and objects using Translation Workbench. You can set up languages for your translations in the Translation Workbench.

To set up translation using Translation Workbench

1. Go to Setup > Administration Setup > Translation Workbench and click Translation Settings
2. Add a Language you want to support for your site.
3. To make the translations available to the users in that language, click Active.
4. Click Save.
5. Now go to Setup > Administration Setup > Translation Workbench and click Translate.
6. Select Language.
7. Select Custom Field as the Setup Component. This way you can translate the fields on an object.
8. Select an Object. A list of all the available fields in the selected object is displayed.
9. For the fields that you want to set the translations, double-click the Field Label Translation.



Master Field Label	Field Label Translation	Field Type
Active		Picklist
Billing Contact Filter Criteria		Long Text Area(5000)
Billing Contact Format		Picklist
Billing Day Of Month		Picklist
Billing Preference		Lookup(Billing Preference)
Calendar Cycle Start		Picklist
Country Code	es_Country Code	Picklist
Credit Memo Email Template		Text(100)
Customer Priority		Picklist
Customer Rating1		Picklist
Default Credit Memo Template		Text(255)
Default Invoice Statement Template		Text(255)
Default Invoice Template		Text(255)
Dunning Policy		Lookup(Dunning Policy)
Exclude From Dunning		Checkbox
Invoice Email Template		Text(100)
Last Invoiced Date		Date
Number of Locations		Number(3, 0)
Payment Email Template		Text(255)
Payment Term		Lookup(Payment Term)
Price List	sp_PriceList	Lookup(Price List)
Ready For Invoice Date		Date

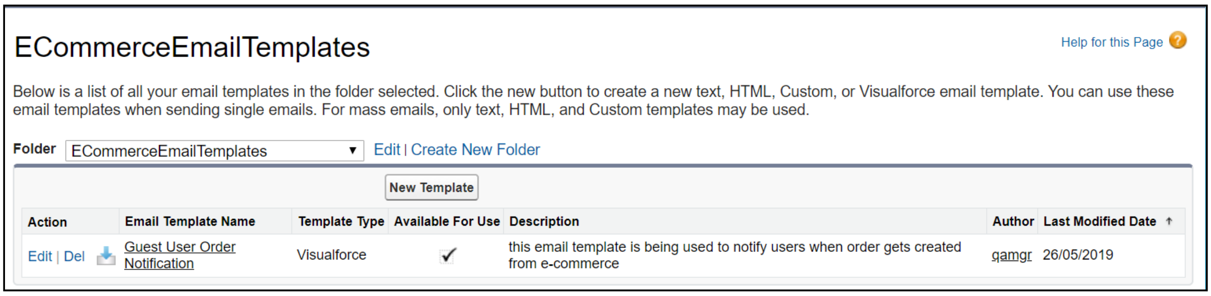
10. Click Save.

Setting Email Notification Template for Checkout

As a guest user or logged in user, you receive a confirmation email after you place an order upon checkout.

To set up email notification

1. Go to Setup > Administration Setup and click Classic Email Templates.
2. When you install E-Commerce package, a folder named ECommerceEmailTempalates is available.



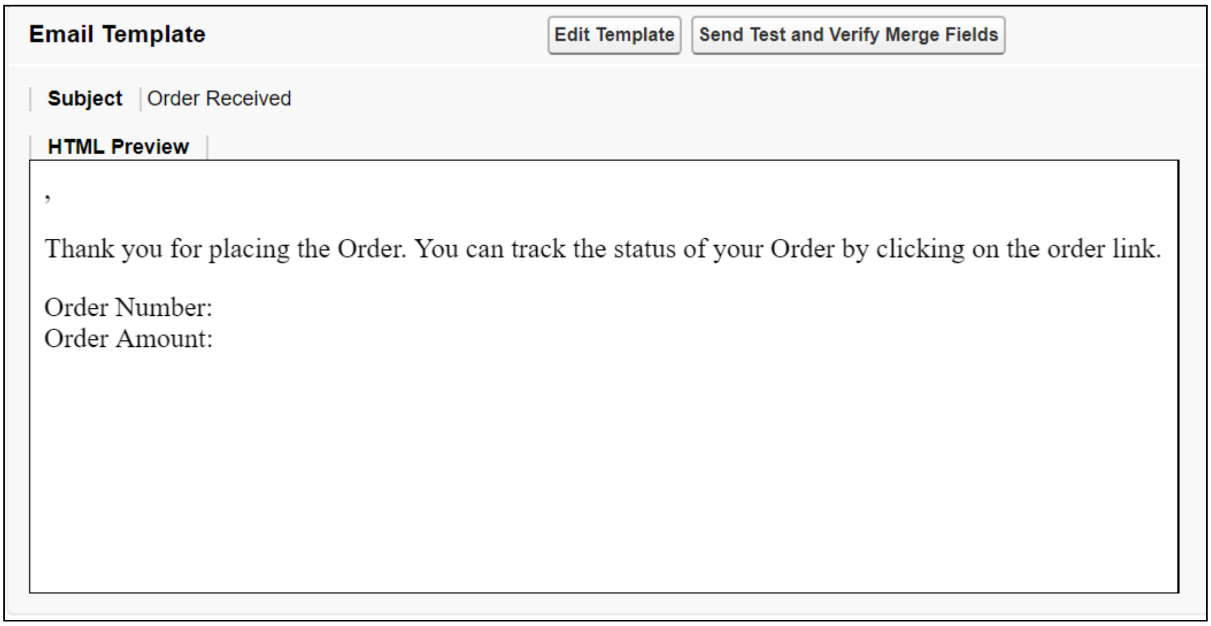
ECommerceEmailTemplates

Below is a list of all your email templates in the folder selected. Click the new button to create a new text, HTML, Custom, or Visualforce email template. You can use these email templates when sending single emails. For mass emails, only text, HTML, and Custom templates may be used.

Folder ECommerceEmailTemplates

Action	Email Template Name	Template Type	Available For Use	Description	Author	Last Modified Date ↑
Edit | Del	Guest User Order Notification	Visualforce	✓	this email template is being used to notify users when order gets created from e-commerce	qamgr	26/05/2019

3. In Folder, select ECommerceEmailTempalates.
4. Click Edit for Guest User Order Notification. This is packaged with the E-Commerce package too. This email template is being used to notify users when order gets created from E-Commerce.
5. Under the Email Template related list, you can view the default email template.



Subject Order Received

HTML Preview

,

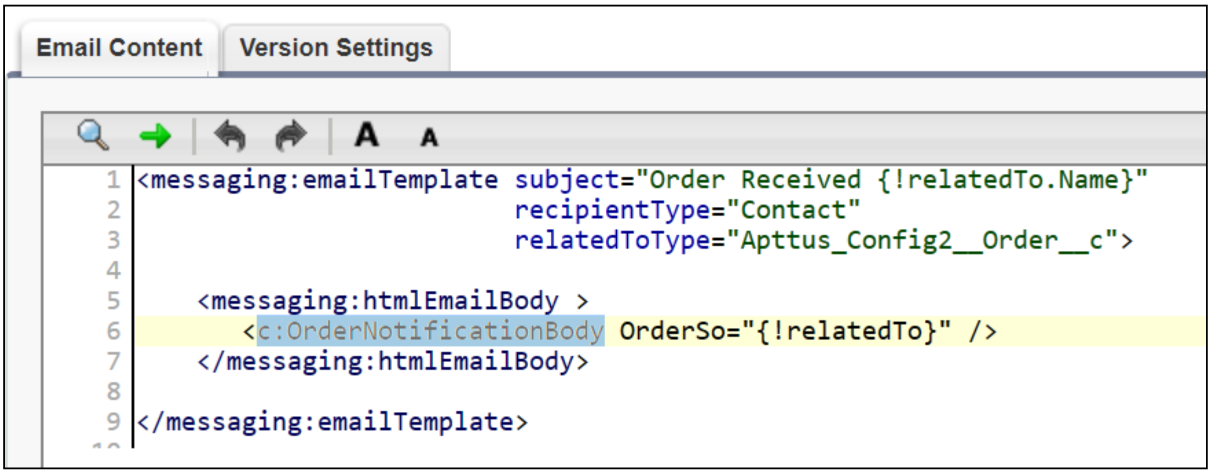
Thank you for placing the Order. You can track the status of your Order by clicking on the order link.

Order Number:
Order Amount:

6. Click Save.

This template is the default template. You may choose to edit the template as per your requirement.

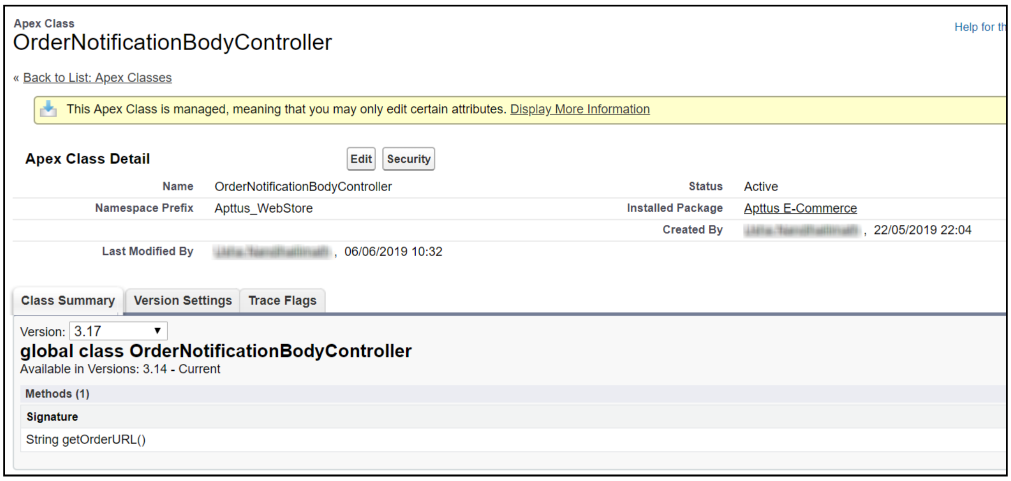
1. Click Edit Template.



```
<messaging:emailTemplate subject="Order Received {!relatedTo.Name}"
                         recipientType="Contact"
                         relatedToType="Apttus_Config2__Order__c">

    <messaging:htmlEmailBody >
       <c:OrderNotificationBody OrderSo="{!relatedTo}" />
    </messaging:htmlEmailBody>

</messaging:emailTemplate>
```

2. This template has a default component that comprises of the body of the email template. The component is highlighted in the image above.
3. To change the body of the template, you can define your own component and within this component you can use OrderNotificationBodyController.getOrderURL(); to fetch the relative Order URL of your Digital Commerce application.

a. Click Setup > App Setup > Develop and click Apex Classes.
b. Browse and click to open OrderNotificationBodyController.



Apex Class

OrderNotificationBodyController

This Apex Class is managed, meaning that you may only edit certain attributes. Display More Information

Apex Class Detail

Name	OrderNotificationBodyController	Status	Active
Namespace Prefix	Apttus_WebStore	Installed Package	Apttus E-Commerce
		Created By	[redacted] , 22/05/2019 22:04
Last Modified By	[redacted] , 06/06/2019 10:32		

Version: 3.17

global class OrderNotificationBodyController

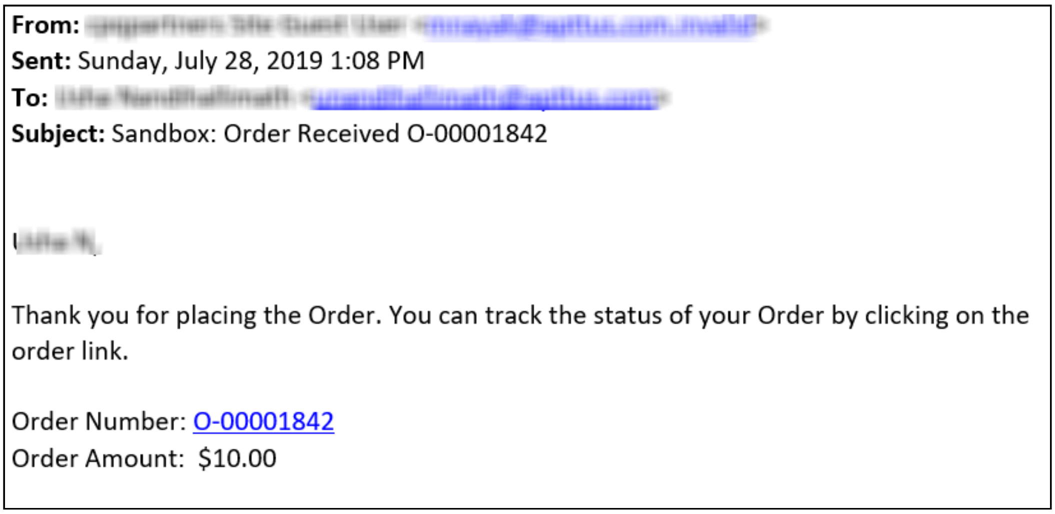
Available in Versions: 3.14 - Current

Methods (1)

Signature
String getOrderURL()

The class that is called is OrderNotificationBodyController.getOrderURL();

See example email below:



From: [redacted]
Sent: Sunday, July 28, 2019 1:08 PM
To: [redacted]
Subject: Sandbox: Order Received O-00001842

[redacted]

Thank you for placing the Order. You can track the status of your Order by clicking on the order link.

Order Number: O-00001842
Order Amount: $10.00

Enabling State and Country Picklists

Your Digital Commerce site may require State and Country picklists. You must configure these picklists on the org level which enables the State and Country fields to display the picklists on the Contact object.

To enable state and country picklists

1. Go to **Setup**, in the Quick Find box, search for *State and Country/Territory Picklists* and click **State and Country/Territory Picklists**. The State and Country/Territory Picklists setup page appears.
2. On the State and Country/Territory Picklists setup page, follow the step-by-step instructions to enable the State and Country picklists.

For complete information, refer Enable and Disable State and Country Picklists.

Adding a Storefront Record

The Store object is created during the deployment process. The Store object is the only supplementary object to the CPQ code. You must create a store record. After installing the managed package, there is no way you can access the Storefront object. You must add a Storefront tab to access the Storefront object.

Apart from the underlying catalog, the E-Commerce package comes with a store object and tab to map a storefront to a catalog. If you are using an Apttus MDO org, there may already be a 'store' object installed. This object is deprecated in favor of the 'Storefront' object that comes with the E-Commerce package.

After your catalog has been setup within Apttus, the next step is to create a 'Storefront' record. The storefront object is very basic and contains only a couple fields to map a storefront to a price list and logo for the guest user. The price list should look up to the price list you want the guest user to access and the logo should be an id or a url of the logo attachment for the store. The storefront record also has a 'banner' related list that can be used to setup banners for the jumbotron component in the reference template. Remember the name of the storefront you created. This will be used in a later step to associate with a storefront codebase.

To add a storefront tab

1. Go to **Setup** > **Create** > **Tabs** and click **New**.
2. From **Object**, select Storefront.
3. From **Tab Style**, select a style and click **Next**.
4. Click **Next** and click **Save**.

The Storefronts tab is created. You can now create a Storefront record.

To add a storefront record

1. Click **All tabs** and click **Storefronts**.
2. Click **New**.
3. For **Storefront Name**, type a mandatory name for your storefront.
4. For **Default Price List**, select a price list. This is the default price list used for guest users in the storefront.
5. For **Default Account for Guest Users**, type an account name as default account for orders placed by guest users from E-Commerce.

6. For **Logo**, use Notes and Attachments to attach an image file. Copy and paste the image ID in the Logo field to reference the image. You can also type a URL to reference an externally hosted logo image.

   > If you do not see the fields above, add them by editing the layout.

7. From **Currency**, select a mandatory currency for your storefront.
8. To enable Asset Based Ordering capabilities for your storefront, select **Enable ABO**.
9. If you enabled Asset Based Ordering, use the **Asset Actions** field to enable them for your storefront.
10. To enable Promotion capabilities for your storefront, select **Enable Promotions**.
11. To enable request quote capabilities for the storefront, select **Enable Request Quote**.
12. Click **Save**.

Adding Storefront Promotional Banners

Storefront Banners are custom objects that are deployed with the managed package. You can add as many banners as you want.

To create storefront banners

1. In the Storefront record, from the Storefront Banners related list, click **New Storefront Banner**.
2. The Storefront field is auto-populated.
3. Enter the following information:

Field	Description
Title	Title for the banner.
Subtitle	Subtitle or sub heading for the banner.
Link	Specify the page that opens, when clicked.
Image	This field references a custom id or a URL.

4. From **Currency**, select a mandatory currency for your storefront banner.
5. Click **Save**. The Storefront Banner details page appears.

To add an image to the Storefront Banner

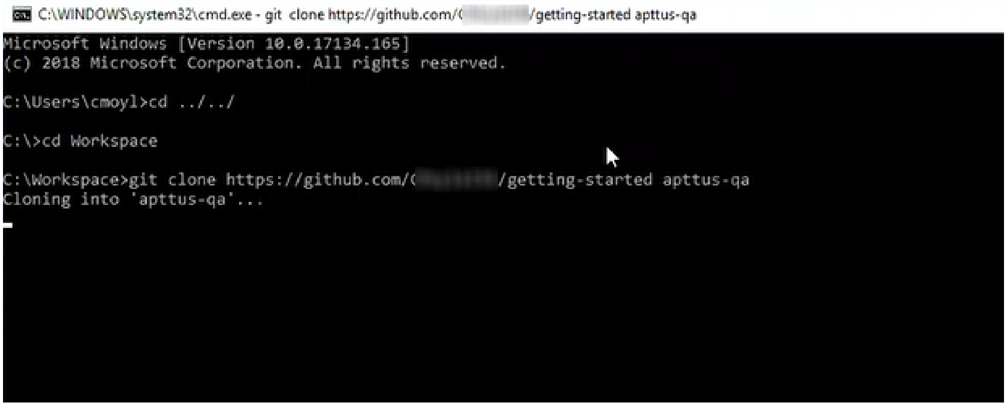
1. In the Storefront record, from the Note & Attachments related list, click **New Note**.
2. For **Title**, enter a title of the Banner image.
3. Click **Save**.
4. Click **Attach File** to attach a banner image and follow the instructions.

Cloning the Reference Templates

You can clone a template from Apttus standard repository and customize as per your specific requirement.

To clone a template from Apttus repository

1. Open command prompt and type *git clone,* the *URL* of the template and a new name for your cloned template. See example below.



C:\WINDOWS\system32\cmd.exe - git clone https://github.com/[redacted]/getting-started apttus-qa

```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\cmoyl>cd ../../

C:\>cd Workspace

C:\Workspace>git clone https://github.com/[redacted]/getting-started apttus-qa
Cloning into 'apttus-qa'...
_
```

2. Press **Enter** on your keyboard.

Your template is cloned.

You can now open the code for the cloned template.

Installing the Reference Template

After cloning a template, the first task is to run the npm install command. The npm install command installs all the dependencies under the package.json folder. One of the dependencies under the package.json folder is the apttus/ecommerce library that bundles all the components library and the data access layers. The npm install process sets up the E-Commerce SDK with your Salesforce instance.

Pre-requisite: You must ensure that the following tasks are completed before you run the npm install command:

- Install required packages on your Salesforce org
- Set up a community, if not already done
- Create a custom field on Accounts for Price List
- Create a Store object and a store record within it
- Ensure you have angular-cli installed on your local machine in order to use the templates

To install the cloned template

1. Open the code of the template.
2. Type **npm install** and press **Enter** on your keyboard. The system prompts with series of questions for your templates to get connected with a Salesforce org.
3. For **Would you like to connect with a Salesforce instance**?, type Yes.
4. For **Which angular project are you using?**, type the name of your project. For example, getting-started.
5. For **What is your salesforce administrator name**, type the username of your Salesforce org.
6. For **What is your salesforce password with the security token**, type the password of your Salesforce org.
7. For **What is your salesforce endpoint,** by default it is https://login.salesforce.com. You can change it to your sandbox or leave it as is.
8. For **Would you like to add a CORS entry to your org to allow localhost development**, type Yes. This adds a whitelist for local host so you can run development in your local machine instead of just the Salesforce instance.
9. For **Which domain will be used for api calls?**, type the domain URL.
10. For **Which community are you using?**, type the name of your community.
11. For **Would you like to deploy the standard ecommerce permission set? (Y/n)**, type Yes to deploy the permission set that is installed with your managed package.

Your reference template along with all dependencies is now installed on your Salesforce org.

Configuring Templates

You can open the repository folder on your local machine to check the setup.

> ❗ A new folder named node_modules is created. You should never modify anything in the node_modules folder. This folder is not part of the repository. All the third party dependencies get installed in this folder. Whenever you run an npm command the dependencies are overridden in this folder.

Salesforce Credentials

As part of the setup, sf.json file is created that contains the credentials for deploying the application. If you want to deploy your Storefront in a different org, you can update the credentials in this sf.json file.

Configuration Parameters

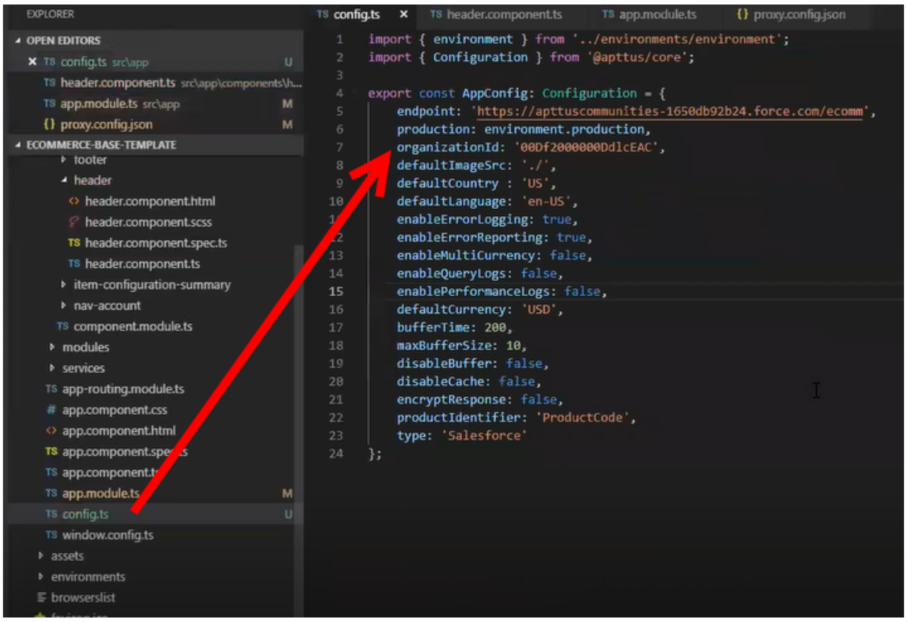
During the npm install phase, a configuration file named config.ts is automatically created. This is a runtime configuration for the application. This file is different from the sf.json file. The config.ts file contains runtime specific variables that helps the application to operate. You must set it up with the following parameters:

Name	Type	Required	Default Value	Description
production	True/false	Yes	-	Specify the environment where you want the application to run.
organizationId	True/false	Yes	-	Specify the org Id of the application. For details, see What is your Salesforce.com Organization Id? section.

Name	Type	Required	Default Value	Description
defaultImageSrc	String	No	-	Specify the URL of the default image to use when no image is found.
defaultCountry	String	No	‘US’	The default country code is "US".
defaultLanguage	String	No	‘en-US’	The default locale is "en-US".
enableErrorLogging	True/false	No	True	Set this to True, in non-production mode, to send error logs to Apttus.
enableErrorReporting	True/false	No	True	When set to True, in non-production mode, it shows a model window to provide user feedback to Apttus.
enableMultiCurrency	True/false	No	False	If using a multi-currency enabled org, set to true to enable currency fields on models.
enableQueryLogs	True/false	No	True	Set to true to print query requests and results in the browser console.
enablePerformanceLogs	True/false	No	True	Set to true to print performance metrics of requests in the browser console.
defaultCurrency	String	No	‘USD’	The default currency to use for guest users. Defaults to USD.

Name	Type	Required	Default Value	Description
bufferTime	Number (milliseconds)	No	200	The number in milliseconds to wait before sending requests. A larger number will batch requests into a single network callout, but may decrease performance.
maxBufferSize	Number	No	10	The maximum number of requests to batch into a single callout.
disableBuffer	True/False	No	False	When set to true, will disable buffered requests entirely. 1 request = 1 network callout.
disableCache	True/false	No	False	When set to true, data returned from requests will not be cached.
encryptResponse	True/false	No	True	When set to true, responses from the server will be encrypted. ⚠ Encryption payload is limited to 1 MB.
productIdentifier (Deprecated)	String	No	'Id'	The API name of the field on the product to use as the unique identifier in the application.
cartRetryLimit	Number	No	10	Maximum number of times the Apttus CPQ reprice operation should be attempted when the state of the cart changes.

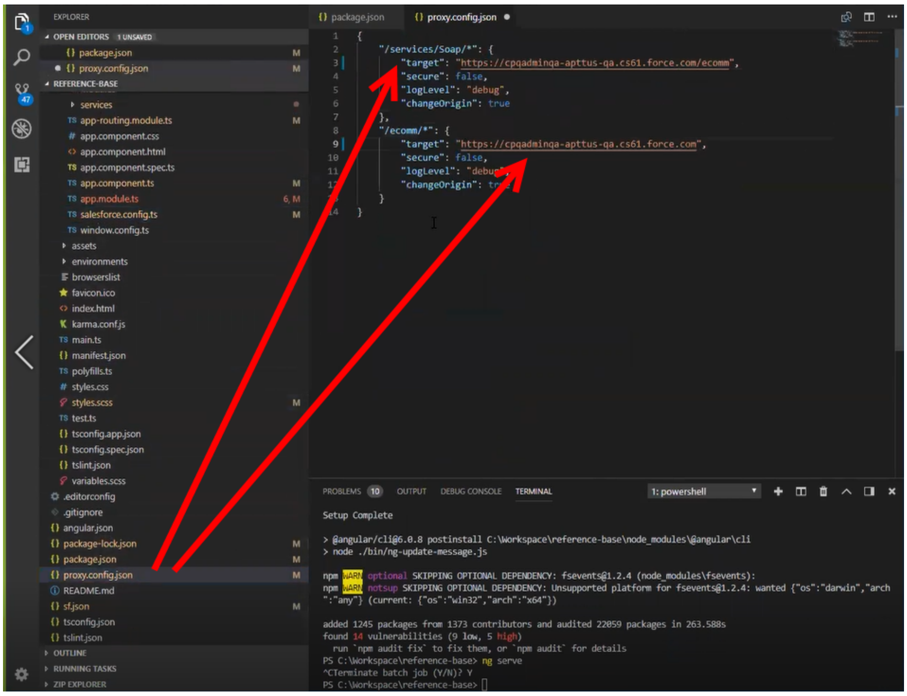
Name	Type	Required	Default Value	Description
debounceTime	Number (milliseconds)	No	200	Amount of time the application should wait for user input before batching DML operations.
storageAccount (Deprecated)	String	No	-	
tenantId (Deprecated)	String	No	-	
tenant (Deprecated)	String	No	-	
proxy	String	No	-	The login proxy to use to bypass CORS restrictions on login.
useIndexedDB	True/false	No	False	When set to true, data cache will be persisted in browser IndexedDB.
storefront	String	Yes	-	
expandDepth	Number	No	10	The maximum number of queries that should be executed to populate subquery data.
queryDepth	Number	No	2	The maximum depth of relationships to be used in a query.



```
import { environment } from '../environments/environment';
import { Configuration } from '@apttus/core';

export const AppConfig: Configuration = {
    endpoint: 'https://apttuscommunities-1650db92b24.force.com/ecomm',
    production: environment.production,
    organizationId: '00Df2000000DdlcEAC',
    defaultImageSrc: './',
    defaultCountry : 'US',
    defaultLanguage: 'en-US',
    enableErrorLogging: true,
    enableErrorReporting: true,
    enableMultiCurrency: false,
    enableQueryLogs: false,
    enablePerformanceLogs: false,
    defaultCurrency: 'USD',
    bufferTime: 200,
    maxBufferSize: 10,
    disableBuffer: false,
    disableCache: false,
    encryptResponse: false,
    productIdentifier: 'ProductCode',
    type: 'Salesforce'
};
```

Setting Up Proxy for Local Development

There is also a proxy.config.json file that gets installed in the root directory as part of the deployment. To configure the proxy.config.json file, you must provide the Community URL in both the target locations. This allows you to make SOAP API calls from your local development server (for functionality like login and reprice cart). Populate the 'target' attributes in that file with the instance URL of your community.



```
{
    "/services/Soap/*": {
        "target": "https://cpqadminqa-apttus-qa.cs61.force.com/ecomm",
        "secure": false,
        "logLevel": "debug",
        "changeOrigin": true
    },
    "/ecomm/*": {
        "target": "https://cpqadminqa-apttus-qa.cs61.force.com",
        "secure": false,
        "logLevel": "debug",
        "changeOrigin": true
    }
}
```

```
Setup Complete

> @angular/cli@6.0.8 postinstall C:\Workspace\reference-base\node_modules\@angular\cli
> node ./bin/ng-update-message.js

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch
":"any"} (current: {"os":"win32","arch":"x64"})

added 1245 packages from 1373 contributors and audited 22059 packages in 263.588s
found 14 vulnerabilities (9 low, 5 high)
  run `npm audit fix` to fix them, or `npm audit` for details
PS C:\Workspace\reference-base> ng serve
^CTerminate batch job (Y/N)? Y
PS C:\Workspace\reference-base> 
```

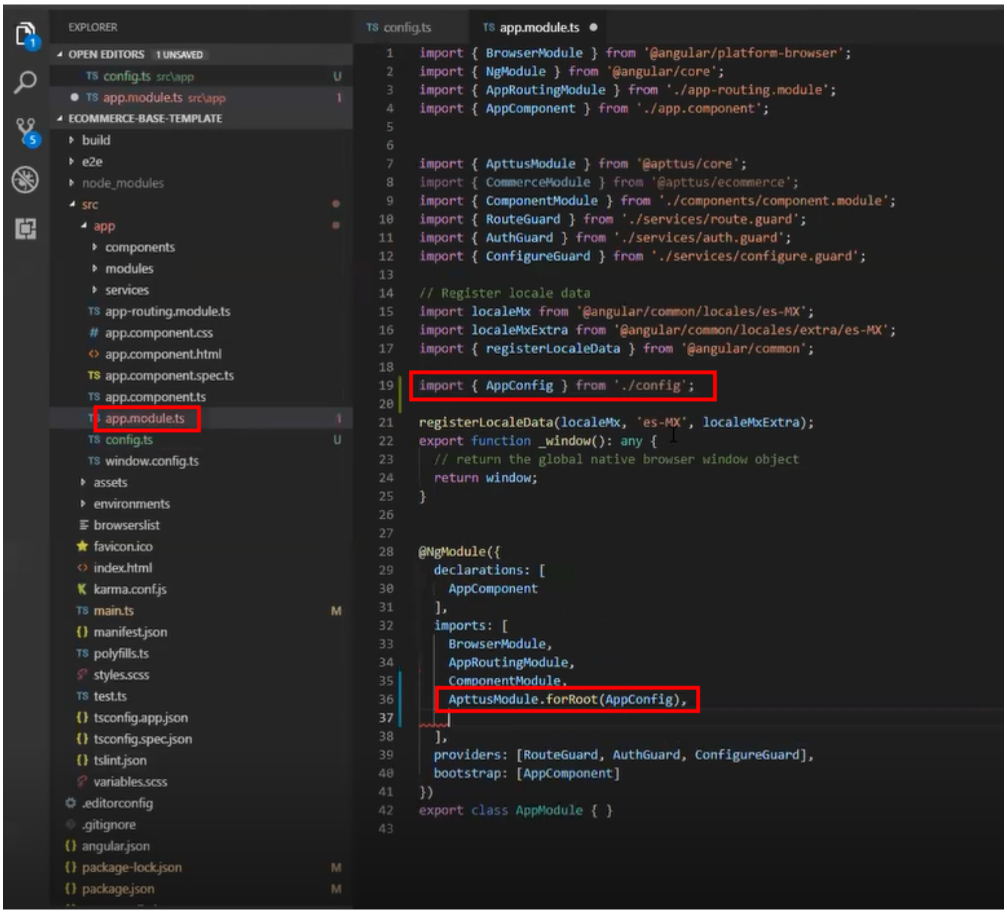
All the code for the templates is open sourced into the library we just cloned and installed. It consists of all the modules such as cart page, home page, account page and more. We can go into any template, for example, Home page layout where you can see some of the components of the Apttus underlying component library. For example, apt-jumbotron. This is a component library that gets installed as part of the apttus/ecommerce. You dont need to essentially build a code for this component. This is a component with the npm package. You just need to modify a single line of HTML code to reference and use it.

You can reference your configuration and import them into your main application module from the app.module.ts file. This app.module.ts file contains an ApttusModule and a CommerceModule. These two modules are getting referenced from the underlying libraries. You must import them into the root module or application to use them.

Importing the app modules in the root module

Add two lines for the following:

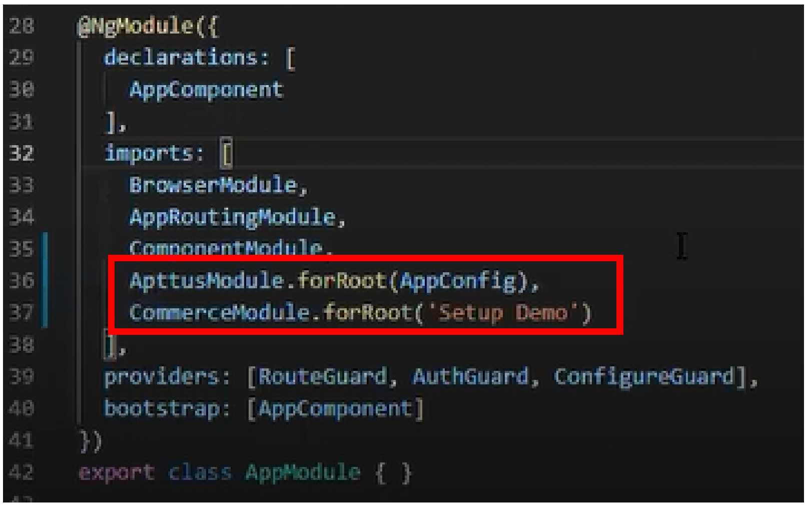
- ApttusModule.forRoot - Apttus module runs all the underlying state management, caching, communication with the Salesforce org. You must import the configuration from the config.ts file.



```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';


import { ApttusModule } from '@apttus/core';
import { CommerceModule } from '@apttus/ecommerce';
import { ComponentModule } from './components/component.module';
import { RouteGuard } from './services/route.guard';
import { AuthGuard } from './services/auth.guard';
import { ConfigureGuard } from './services/configure.guard';

// Register locale data
import localeMx from '@angular/common/locales/es-MX';
import localeMxExtra from '@angular/common/locales/extra/es-MX';
import { registerLocaleData } from '@angular/common';

import { AppConfig } from './config';

registerLocaleData(localeMx, 'es-MX', localeMxExtra);
export function _window(): any {
  // return the global native browser window object
  return window;
}


@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ComponentModule,
    ApttusModule.forRoot(AppConfig),
    
  ],
  providers: [RouteGuard, AuthGuard, ConfigureGuard],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- CommerceModule.forRoot - In the forRoot method of the CommerceModule declare the storefront that you want to use. For example, CommerceModule.forRoot('Tier 1')

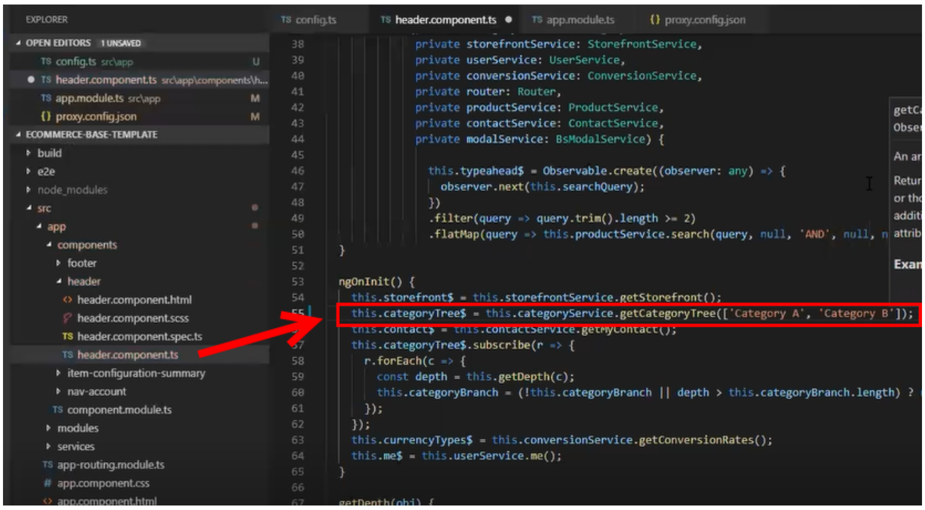
This defines the Storefront for your application.



```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ComponentModule,
    ApttusModule.forRoot(AppConfig),
    CommerceModule.forRoot('Setup Demo')
  ],
  providers: [RouteGuard, AuthGuard, ConfigureGuard],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Setting Up Subset of Categories

You can set up specific categories or subset of categories from the header.component.ts file.



```
            private storefrontService: StorefrontService,
            private userService: UserService,
            private conversionService: ConversionService,
            private router: Router,
            private productService: ProductService,
            private contactService: ContactService,
            private modalService: BsModalService) {

              this.typeahead$ = Observable.create((observer: any) => {
                observer.next(this.searchQuery);
              })
              .filter(query => query.trim().length >= 2)
              .flatMap(query => this.productService.search(query, null, 'AND', null, n
  }

  ngOnInit() {
    this.storefront$ = this.storefrontService.getStorefront();
    this.categoryTree$ = this.categoryService.getCategoryTree(['Category A', 'Category B']);
    this.contact$ = this.contactService.getMyContact();
    this.categoryTree$.subscribe(r => {
      r.forEach(c => {
        const depth = this.getDepth(c);
        this.categoryBranch = (!this.categoryBranch || depth > this.categoryBranch.length) ?
      });
    });
    this.currencyTypes$ = this.conversionService.getConversionRates();
    this.me$ = this.userService.me();
  }

  getDepth(obj) {
```

Turning Off Sentry for a Customer

Our angular application is wired into Sentry, any errors that occur for customers are sent to sentry. You can view those in sentry, along with stacktrace and debug them. If customers dont want such information to be sent to us, you may consider turning it off for customers.

Go to config.ts and do the following:

- enableErrorLogging - Set this to False.
- enableErrorReporting - Set this to False.

Local Development Setup

Now you can run your application locally on your local machine by running the ***ng serve*** command. This runs the E-Commerce site locally in your local machine against the configuration that you set up during the npm install phase.

Navigate to http://localhost:4200/ to see how your application looks like. The application displays categories, products, pricing and more based on the Price List selected in the Storefront record. The app is automatically reloaded if you change any of the source files.

Bootstrap Theme Changes

Bootstrap and ngx Bootstrap (angular wrapper to Bootstrap) is installed during the npm install phase. The templates are built around the Bootstrap as a UI framework. You can change to any other mechanism if you dont want to use Bootstrap.

A variables.scss file is installed in the template. If you want to do some quick and easy theme change to your template, you can modify this file as per your requirement. All this follows standard Bootstrap framework construct to modify cards, dropdowns, forms, buttons and more.

Server Deployment

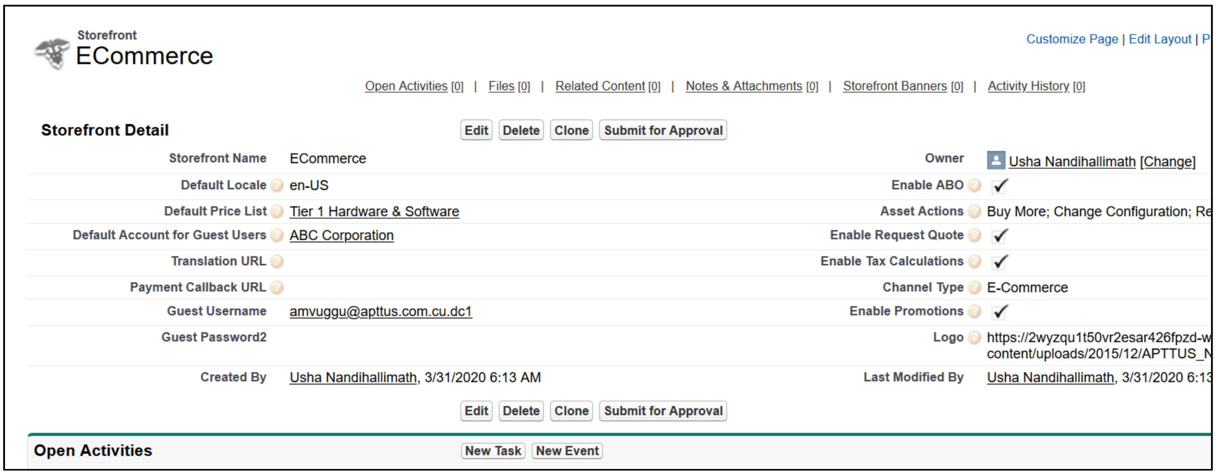
After everything looks correct in your local machine, you are now ready to deploy your E-Commerce application on your Salesforce Org.

Deploying Standard Template

Along with the managed package you get a standard template.

To deploy the standard template

1. Create a Storefront Record pertaining to your desired template. To deploy the ecommerce template, you must name your storefront **ECommerce**. All other settings are optional.

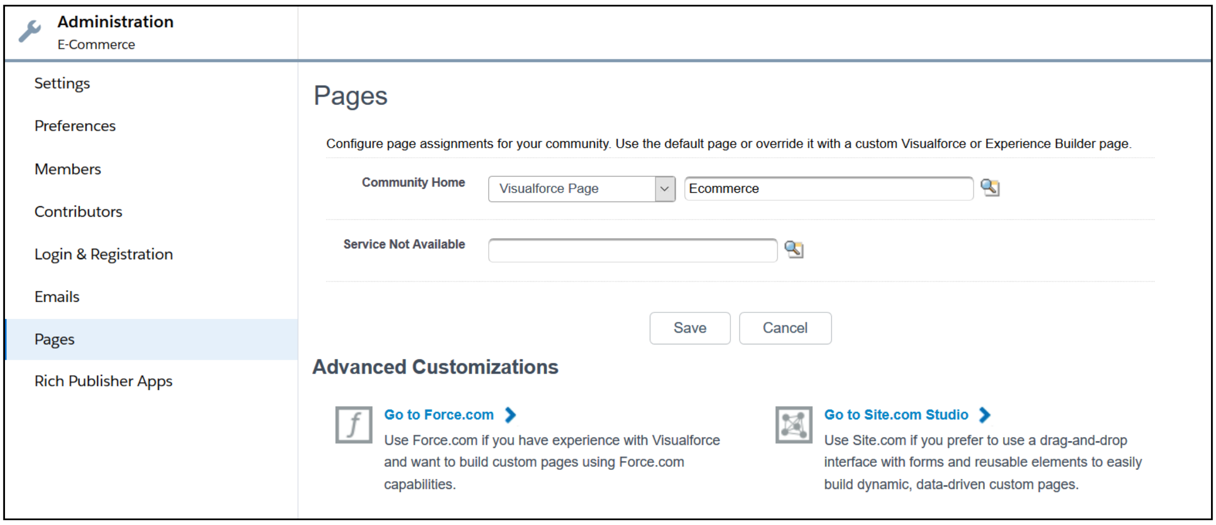

Storefront
ECommerce
Customize Page | Edit Layout | P
Open Activities [0] | Files [0] | Related Content [0] | Notes & Attachments [0] | Storefront Banners [0] | Activity History [0]
Edit
Delete
Clone
Submit for Approval
New Task
New Event

Storefront Detail

Storefront Name	ECommerce	Owner	Usha Nandihallimath [Change]
Default Locale	en-US	Enable ABO	✓
Default Price List	Tier 1 Hardware & Software	Asset Actions	Buy More; Change Configuration; Re
Default Account for Guest Users	ABC Corporation	Enable Request Quote	✓
Translation URL		Enable Tax Calculations	✓
Payment Callback URL		Channel Type	E-Commerce
Guest Username	amvuggu@apttus.com.cu.dc1	Enable Promotions	✓
Guest Password2		Logo	https://2wyzqu1t50vr2esar426fpzd-w content/uploads/2015/12/APTTUS_N
Created By	Usha Nandihallimath, 3/31/2020 6:13 AM	Last Modified By	Usha Nandihallimath, 3/31/2020 6:13

Open Activities

2. Set up a Community to host the storefront you created. After the community is set up, assign the corresponding visualforce page as the home page of the community.
3. Within the community workspace, go to the Administration section, and from the left menu, click **Pages**.



Pages

Configure page assignments for your community. Use the default page or override it with a custom Visualforce or Experience Builder page.

Community Home: Visualforce Page | Ecommerce

Service Not Available:

Advanced Customizations

Go to Force.com: Use Force.com if you have experience with Visualforce and want to build custom pages using Force.com capabilities.

Go to Site.com Studio: Use Site.com if you prefer to use a drag-and-drop interface with forms and reusable elements to easily build dynamic, data-driven custom pages.

4. From the Community Home drop-down, select **Visualforce Page**. Search and select **ECommerce** for your ecommerce template and click **Save**.

Your standard template is successfully deployed. From All Communities, click the URL to see how your application looks in your Salesforce org.

Deploying Custom Template

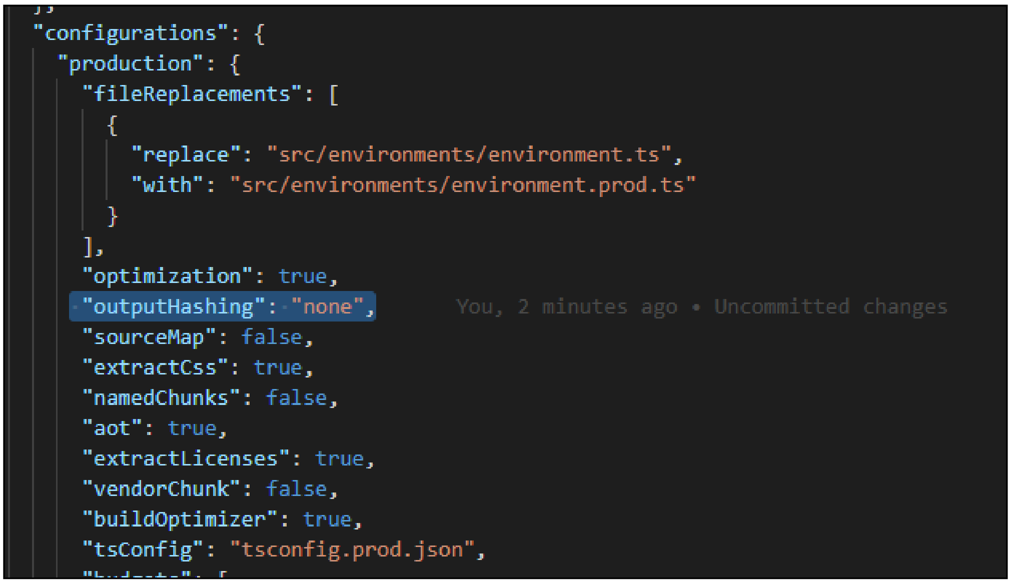
If you have access the Digital Commerce SDK and have customizations of your own. You can deploy your customizations to Salesforce and host them on your community.

To deploy custom template

1. You must set up the **Main.ts** file. The Digital Commerce SDK breaks out the individual pages into *chunks* that make load times quicker for those pages. In order to support this within Salesforce, you must point the webpack script to the chunks. You can do this by modifying the main.ts file as below:

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

declare var __webpack_public_path__: string;
const sv = (<any>window).sv;

if (environment.production) {
  enableProdMode();
  if (sv && sv.resource) {
    __webpack_public_path__ = sv.resource + '/';
  }
}


platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

2. You must change the output hashing in Angular.json. By default, when the application is built, unique hash keys are assigned to all the scripts preventing certain caching mechanisms in the browser. This is not required in Salesforce and may cause issues. To prevent such issues, you must set the outputHashing parameter in the angular.json file to *none* for your build.

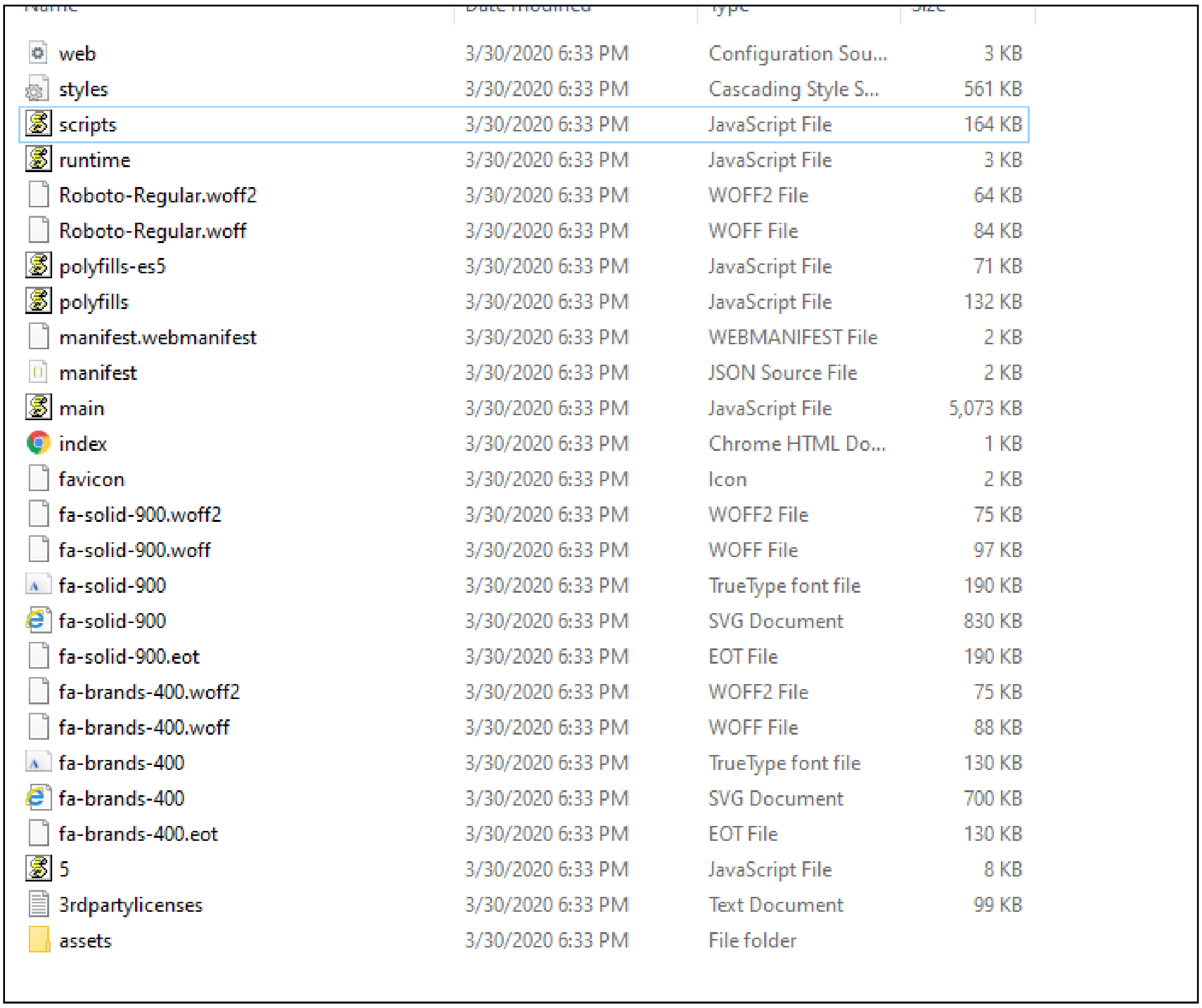


```
  "configurations": {
    "production": {
      "fileReplacements": [
        {
          "replace": "src/environments/environment.ts",
          "with": "src/environments/environment.prod.ts"
        }
      ],
      "optimization": true,
      "outputHashing": "none",      You, 2 minutes ago • Uncommitted changes
      "sourceMap": false,
      "extractCss": true,
      "namedChunks": false,
      "aot": true,
      "extractLicenses": true,
      "vendorChunk": false,
      "buildOptimizer": true,
      "tsConfig": "tsconfig.prod.json",
      "budgets": [
```

3. Build your application using angular cli. The angular cli provides an easy script to build your application. Simply run the following replacing production with whatever configuration you are using.

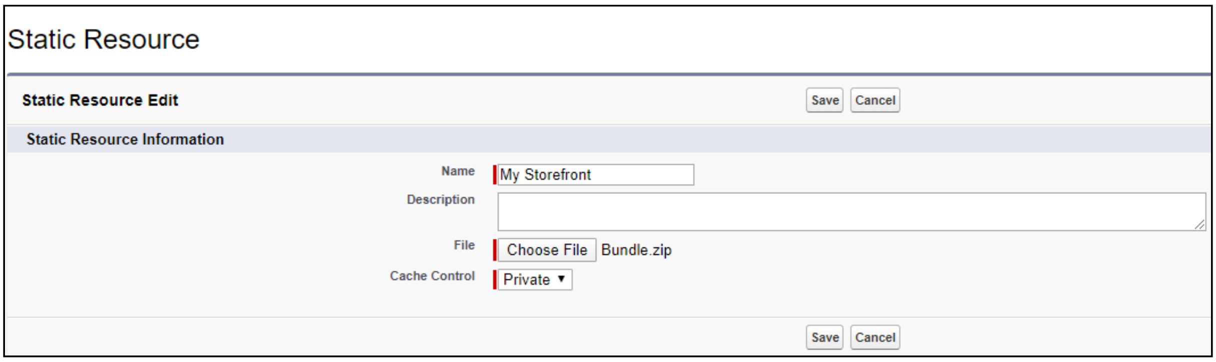
```
ng build --configuration="production"
```

4. After the application is built, an output folder called ***dist*** with the entire contents of your application in various resource files is created.



Name	Date modified	Type	Size
web	3/30/2020 6:33 PM	Configuration Sou...	3 KB
styles	3/30/2020 6:33 PM	Cascading Style S...	561 KB
scripts	3/30/2020 6:33 PM	JavaScript File	164 KB
runtime	3/30/2020 6:33 PM	JavaScript File	3 KB
Roboto-Regular.woff2	3/30/2020 6:33 PM	WOFF2 File	64 KB
Roboto-Regular.woff	3/30/2020 6:33 PM	WOFF File	84 KB
polyfills-es5	3/30/2020 6:33 PM	JavaScript File	71 KB
polyfills	3/30/2020 6:33 PM	JavaScript File	132 KB
manifest.webmanifest	3/30/2020 6:33 PM	WEBMANIFEST File	2 KB
manifest	3/30/2020 6:33 PM	JSON Source File	2 KB
main	3/30/2020 6:33 PM	JavaScript File	5,073 KB
index	3/30/2020 6:33 PM	Chrome HTML Do...	1 KB
favicon	3/30/2020 6:33 PM	Icon	2 KB
fa-solid-900.woff2	3/30/2020 6:33 PM	WOFF2 File	75 KB
fa-solid-900.woff	3/30/2020 6:33 PM	WOFF File	97 KB
fa-solid-900	3/30/2020 6:33 PM	TrueType font file	190 KB
fa-solid-900	3/30/2020 6:33 PM	SVG Document	830 KB
fa-solid-900.eot	3/30/2020 6:33 PM	EOT File	190 KB
fa-brands-400.woff2	3/30/2020 6:33 PM	WOFF2 File	75 KB
fa-brands-400.woff	3/30/2020 6:33 PM	WOFF File	88 KB
fa-brands-400	3/30/2020 6:33 PM	TrueType font file	130 KB
fa-brands-400	3/30/2020 6:33 PM	SVG Document	700 KB
fa-brands-400.eot	3/30/2020 6:33 PM	EOT File	130 KB
5	3/30/2020 6:33 PM	JavaScript File	8 KB
3rdpartylicenses	3/30/2020 6:33 PM	Text Document	99 KB
assets	3/30/2020 6:33 PM	File folder	

5. Create the static resource. After the contents of the application is built, you must bundle it and deploy it to a Salesforce static resource. Select all of the build files and put them into a zip file with a name of your choice.
6. Go to **Setup** > **Develop** > **Static Resources** and click **New** to create a new static resource with the zip file as its contents.



Name: My Storefront

Description:

File: Choose File Bundle.zip

Cache Control: Private

7. Create the Visualforce Page. After the static resource has been created, you must create a visualforce page to display the contents of the resource file. See the example visualforce page below you can use to host your application. You can make any changes you want, but there are a couple considerations:
   - You must add the script to assign the *window.sv.resource* variable to the URL of the static resource.
   - Remove the base tag or assign it to the relative URL of the community.
   - Replace *Resource.ECommerce* in the below template with the name of the static resource file you uploaded.

```
<apex:page standardStylesheets="false" showHeader="false" sidebar="false"
docType="html-5.0" applyHtmlTag="false" applyBodyTag="false">
  <head>
    <meta charset="utf-8" />
    <title>ECommerce</title>

    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" type="image/x-icon" href="favicon.ico" />
    <link rel="manifest" href="manifest.webmanifest" />
    <meta name="theme-color" content="#1976d2" />
    <link rel="stylesheet" href="{!URLFOR($Resource.ECommerce, 'styles.css')}" />
    <script>
        window.sv = {
            resource: '{!URLFOR($Resource.ECommerce)}/'
        };
    </script>
  </head>

  <body>
    <app-root></app-root>
    <noscript
      >Please enable JavaScript to continue using this application.</noscript
    >
    <script src="{!URLFOR($Resource.ECommerce, 'runtime.js')}"></script>
    <script src="{!URLFOR($Resource.ECommerce, 'polyfills-es5.js')}"></script>
    <script src="{!URLFOR($Resource.ECommerce, 'polyfills.js')}"></script>
    <script src="{!URLFOR($Resource.ECommerce, 'scripts.js')}"></script>
    <script src="{!URLFOR($Resource.ECommerce, 'main.js')}"></script>
  </body>
</apex:page>
```

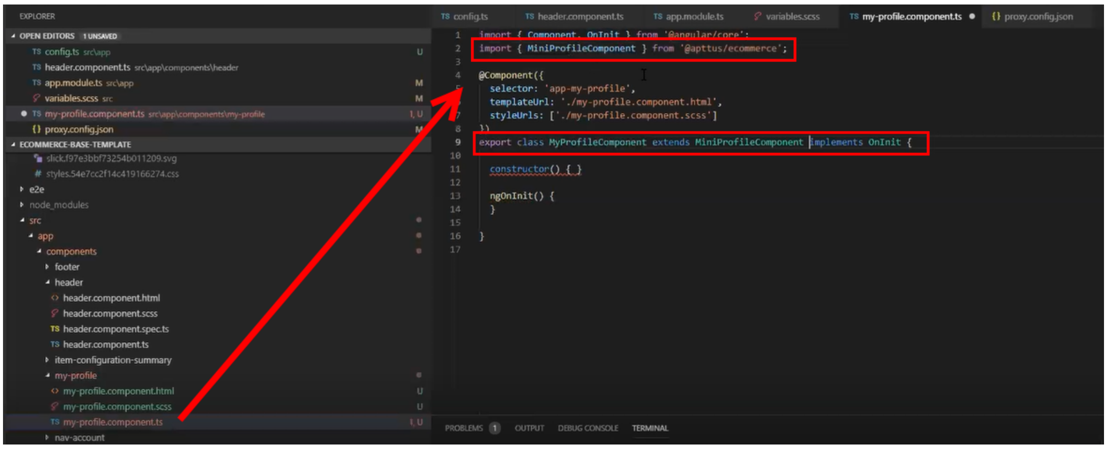
Customizing Your Application

Customizing HTML Content and Standard Components

You can override the HTML for any component that comes with your angular application. You can do so by writing an extension component. Similarly, you can override the methods within it as well. For example, there is a registration method and you want to retain it but want to change the look and feel of the template, you must write an extension component. You can override extension classes, extension services or extension components that are present within the angular library.

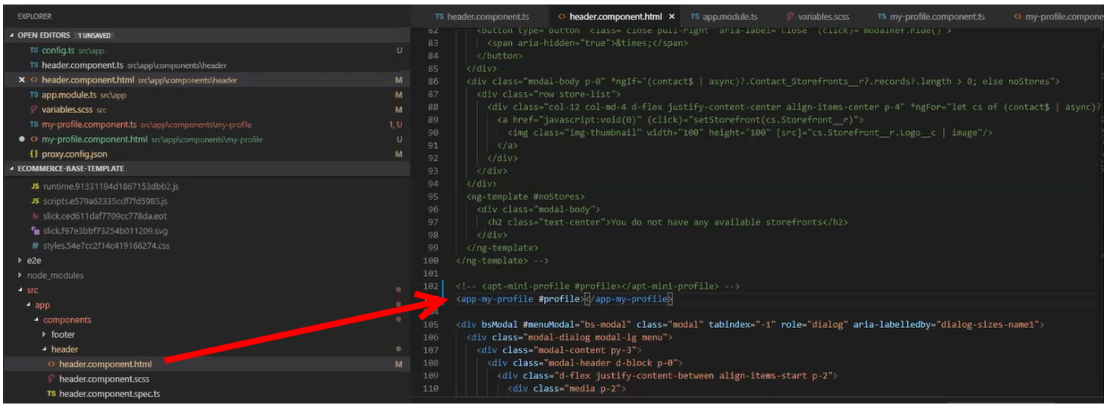
To customize the HTML content

1. Go to header.component.html file and browse for apt-mini-profile component. The HTML content are all bundled in this component.
2. Go to your application and use basic angular syntax to generate a component. For example: ng g c components/my-profile --module-components/component.module.ts --spec-false
3. A new my-profile.component.ts is created.
4. Import the miniprofile component from @apttus/ecommerce and extend the miniprofile component. By doing this, all the controller code is inherited from the miniprofile component.



```
import { Component, OnInit } from '@angular/core';
import { MiniProfileComponent } from '@apttus/ecommerce';

@Component({
  selector: 'app-my-profile',
  templateUrl: './my-profile.component.html',
  styleUrls: ['./my-profile.component.scss']
})
export class MyProfileComponent extends MiniProfileComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

5. Refer the Digital Commerce on Salesforce SDK and search for miniprofilecomponent. Click on the Template tab for HTML template for all of the components in it.
6. Copy all the HTML content and paste it in the my-profile.component.html you just created and save it.
7. From the new component you just created, copy the selector and go to the header where it is referenced and paste the selector.


EXPLORER
OPEN EDITORS 1 UNSAVED
config.ts src\app U
header.component.ts src\app\components\header
header.component.html src\app\components\header M
app.module.ts src\app M
variables.scss src M
my-profile.component.ts src\app\components\my-profile 1, U
my-profile.component.html src\app\components\my-profile U
proxy.config.json M
ECOMMERCE-BASE-TEMPLATE
runtime.91331194d1867153dbb2.js
scripts.e579a62335cdf7fd5985.js
slick.ced611daf7709cc778da.eot
slick.f97e3bbf73254b011209.svg
styles.54e7cc2f14c419166274.css
e2e
node_modules
src
app
components
footer
header
header.component.html M
header.component.scss
header.component.spec.ts
header.component.ts
header.component.html
app.module.ts
variables.scss
my-profile.component.ts
my-profile.compone

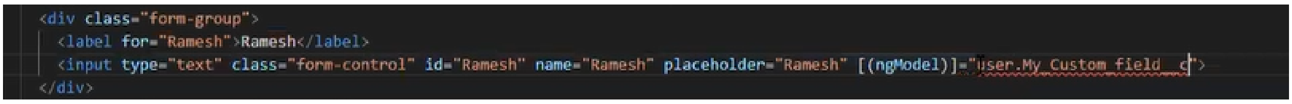
```
      <button type="button" class="close pull-right" aria-label="Close" (click)="modalRef.hide()">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    <div class="modal-body p-0" *ngIf="(contact$ | async)?.Contact_Storefronts__r?.records?.length > 0; else noStores">
      <div class="row store-list">
        <div class="col-12 col-md-4 d-flex justify-content-center align-items-center p-4" *ngFor="let cs of (contact$ | async)?
          <a href="javascript:void(0)" (click)="setStorefront(cs.Storefront__r)">
            <img class="img-thumbnail" width="100" height="100" [src]="cs.Storefront__r.Logo__c | image"/>
          </a>
        </div>
      </div>
    </div>
    <ng-template #noStores>
      <div class="modal-body">
        <h2 class="text-center">You do not have any available storefronts</h2>
      </div>
    </ng-template>
  </ng-template> -->

  <!-- <apt-mini-profile #profile></apt-mini-profile> -->
  <app-my-profile #profile></app-my-profile>

  <div bsModal #menuModal="bs-modal" class="modal" tabindex="-1" role="dialog" aria-labelledby="dialog-sizes-name1">
    <div class="modal-dialog modal-lg menu">
      <div class="modal-content py-3">
        <div class="modal-header d-block p-0">
          <div class="d-flex justify-content-between align-items-start p-2">
            <div class="media p-2">
```

8. Run ng serve command.

The profile component is overridden.

Go to my-profile.component.html and within the HTML content add a custom field and save.

Refresh your application on the local machine and you will see the newly added custom field. If you want to assign the custom field to a user that is associated with the component, you can do so by modifying the ngModel. For example:
[(ngModel)]="user.My_Custom_field__c"



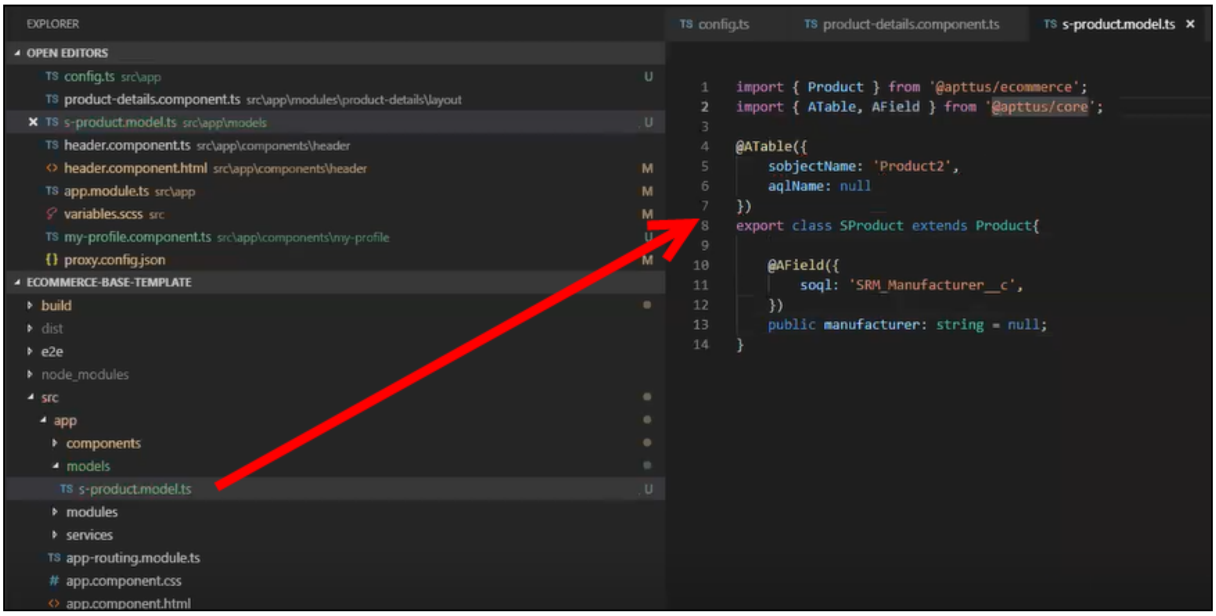
```
<div class="form-group">
  <label for="Ramesh">Ramesh</label>
  <input type="text" class="form-control" id="Ramesh" name="Ramesh" placeholder="Ramesh" [(ngModel)]="user.My_Custom_field__c">
</div>
```

Adding Custom Fields on Object Models

You can add custom fields to a modal by using the extended model in components. The strategy to add customization's to the application is to use the object-oriented nature of type script to extend and override the out of the box content. Let us see how to add a custom field to the Product object.

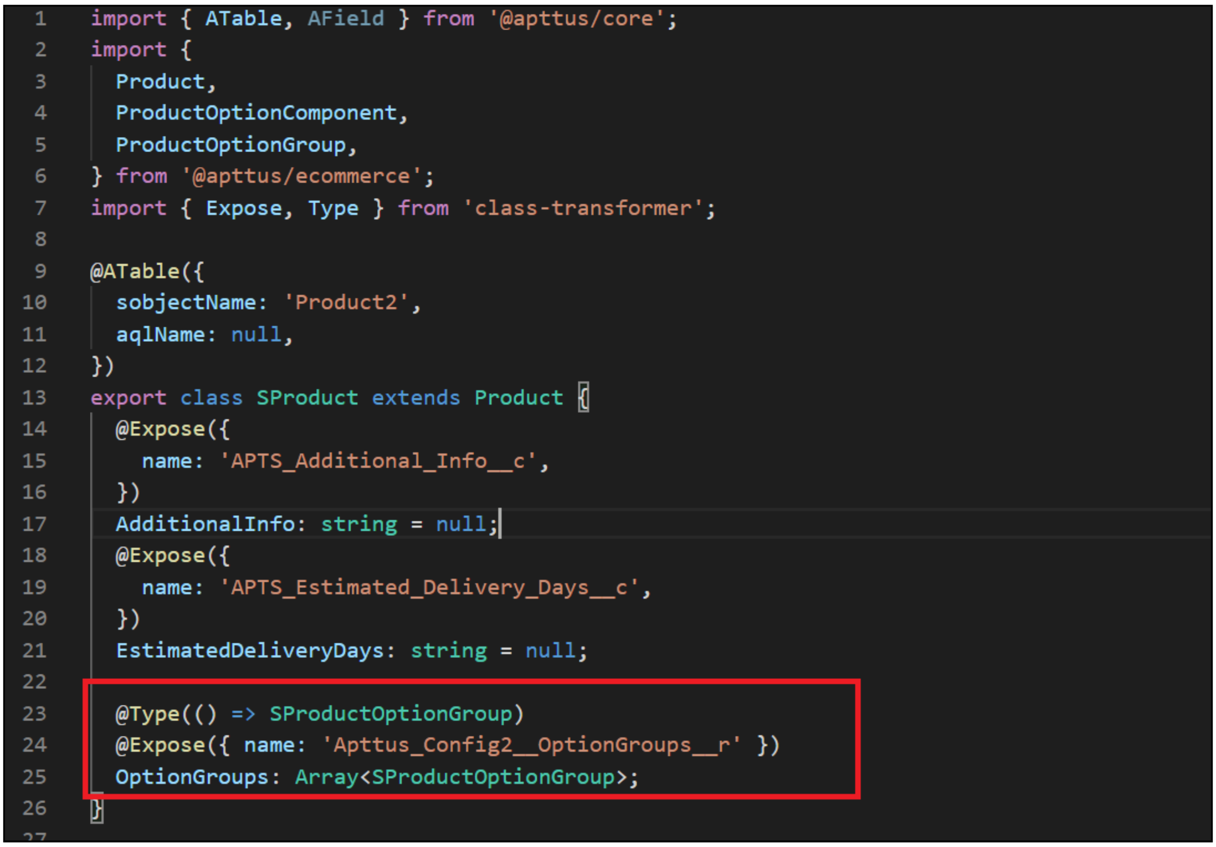
To create a custom model

1. Right-click on the App folder and click **Create Folder** to create a new folder for models and name it. For example: models.
2. Right-click on the models folder and click **Create File** to create a product model and name it. For example: s-product.model.ts
3. In the s-product.model.ts, import Product from @apttus/ecommerce. For example: import { Product } from @apttus/ecommerce
4. Create the extension of the product for your custom model.
   For example: export class SProduct extends Product {} The SProduct class has all the properties of the original product.

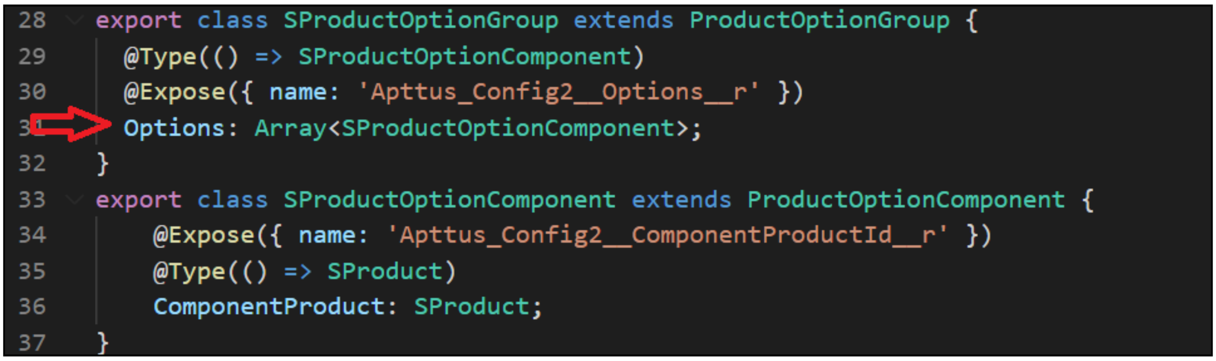


```
import { Product } from '@apttus/ecommerce';
import { ATable, AField } from '@apttus/core';

@ATable({
    sobjectName: 'Product2',
    aqlName: null
})
export class SProduct extends Product{

    @AField({
        soql: 'SRM_Manufacturer__c',
    })
    public manufacturer: string = null;
}
```

Example for option group lookups to fetch the option products:



```
import { ATable, AField } from '@apttus/core';
import {
  Product,
  ProductOptionComponent,
  ProductOptionGroup,
} from '@apttus/ecommerce';
import { Expose, Type } from 'class-transformer';

@ATable({
  sobjectName: 'Product2',
  aqlName: null,
})
export class SProduct extends Product {
  @Expose({
    name: 'APTS_Additional_Info__c',
  })
  AdditionalInfo: string = null;
  @Expose({
    name: 'APTS_Estimated_Delivery_Days__c',
  })
  EstimatedDeliveryDays: string = null;

  @Type(() => SProductOptionGroup)
  @Expose({ name: 'Apttus_Config2__OptionGroups__r' })
  OptionGroups: Array<SProductOptionGroup>;
}
```

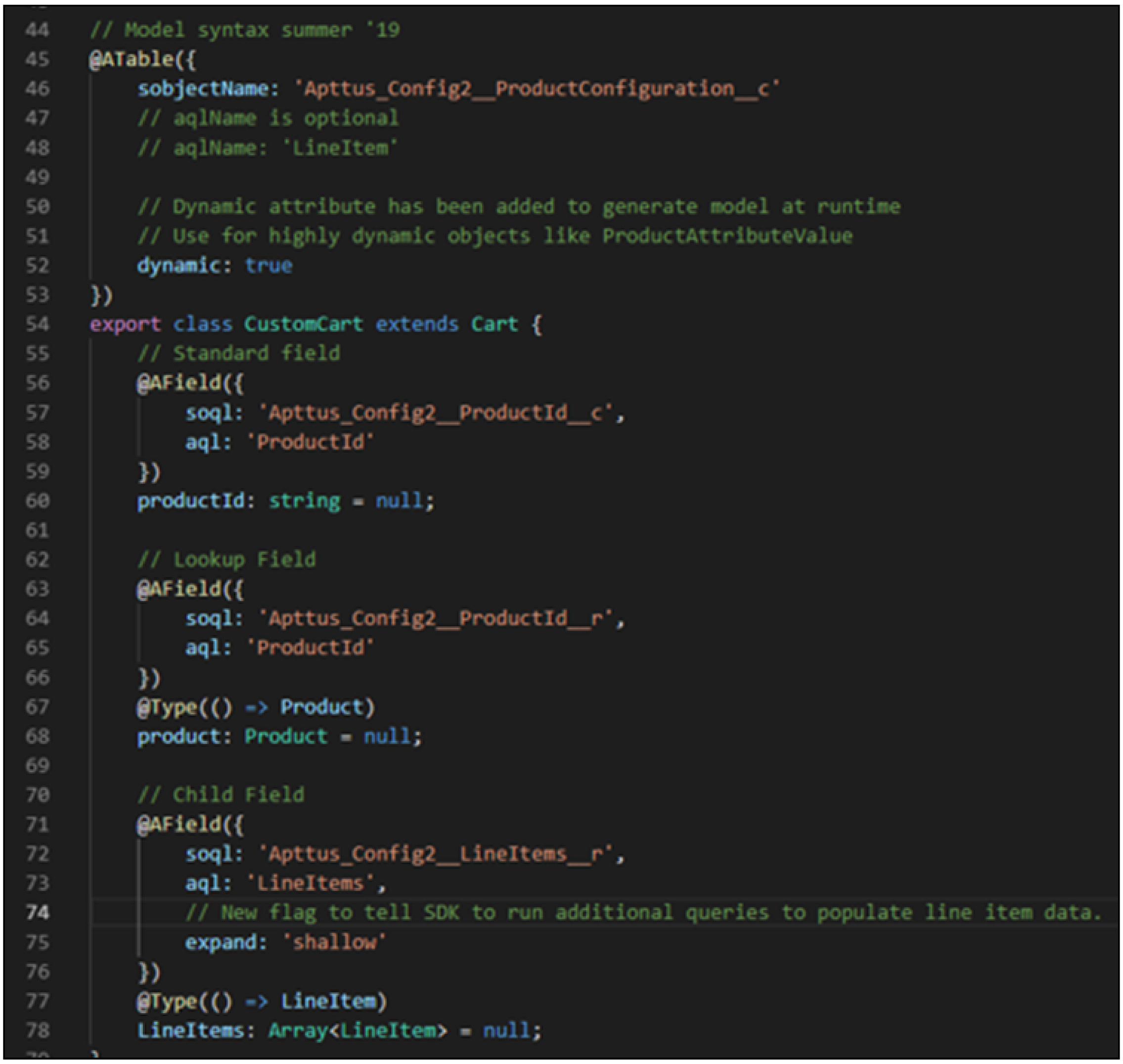


```
export class SProductOptionGroup extends ProductOptionGroup {
  @Type(() => SProductOptionComponent)
  @Expose({ name: 'Apttus_Config2__Options__r' })
  Options: Array<SProductOptionComponent>;
}
export class SProductOptionComponent extends ProductOptionComponent {
    @Expose({ name: 'Apttus_Config2__ComponentProductId__r' })
    @Type(() => SProduct)
    ComponentProduct: SProduct;
}
```

5. When adding custom properties to your model, you must associate the following decorators from the @apttus/core library. These are used to map your classes to the underlying Salesforce objects. Basically, this mechanism is to map typescript classes with Salesforce objects using decorators.
    - ATable - This is used to map product class to any object on the backend by specifying what object you want to map it up to. For Salesforce, the object name can be added to the field sobjectName.
    - AField - This is used to specify the Salesforce name of the custom field. For example: SRM_Manufacturer__c. It is not necessary to use the Salesforce syntax for the field name. You can provide a generic name. For example manufacturer.

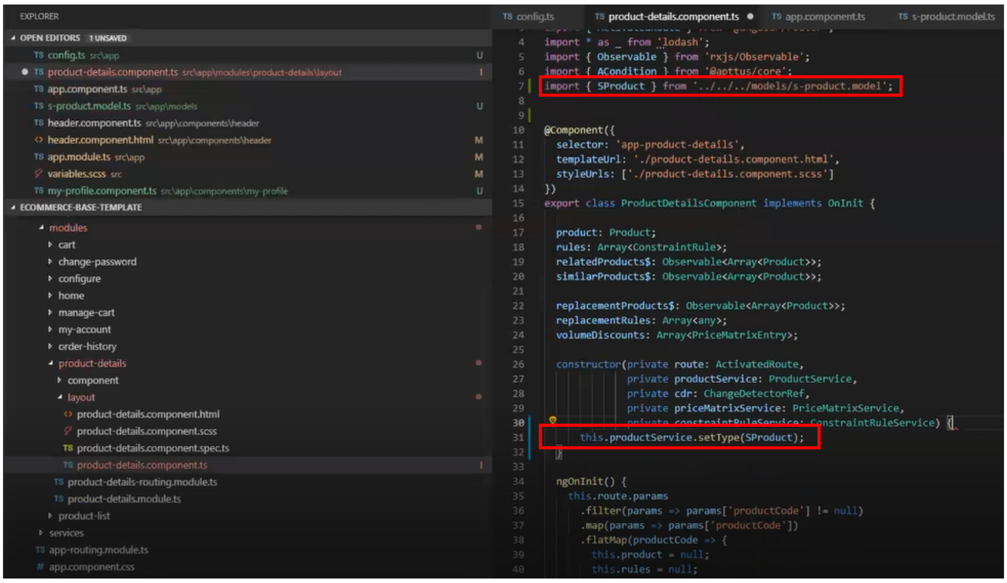
You must specify a default value for any field created. Do not leave a blank value.

- Expose - This is used to specify the Salesforce name of the custom field. For example: Apttus_Config2__ComponentProductId__r. This is applicable only for a few models when trying to extend instead of using AField. You must specify a default value for any field created. Do not leave a blank value.
    - Account
    - Adjustmentlineitem
    - AppliedRuleActionInfo
    - AppliedRuleInfo
    - AssetLineitem
    - Cart Lineitem
    - Cart
    - Category
    - Classification
    - Incentive
    - Order
    - Order line item
    - PriceList
    - PriceListCategory
    - PriceListItem
    - Product
    - ProductAttribute
    - ProductAttributeGroup
    - ProductAttributeGroupMember
    - ProductFeature
    - ProductGroup
    - ProductOptionGroup
    - ProductTranslation
    - Quote
    - Quote lineitem
    - SummaryGroup
    - TaxBreakup
    - TaxCode
- Type - This is used to determine relationship types (child or lookups). It is recommended to also set the default value to 'null' on all attributes unless a default value is explicitly needed.



```
// Model syntax summer '19
@ATable({
    sobjectName: 'Apttus_Config2__ProductConfiguration__c'
    // aqlName is optional
    // aqlName: 'LineItem'

    // Dynamic attribute has been added to generate model at runtime
    // Use for highly dynamic objects like ProductAttributeValue
    dynamic: true
})
export class CustomCart extends Cart {
    // Standard field
    @AField({
        soql: 'Apttus_Config2__ProductId__c',
        aql: 'ProductId'
    })
    productId: string = null;

    // Lookup Field
    @AField({
        soql: 'Apttus_Config2__ProductId__r',
        aql: 'ProductId'
    })
    @Type(() => Product)
    product: Product = null;

    // Child Field
    @AField({
        soql: 'Apttus_Config2__LineItems__r',
        aql: 'LineItems',
        // New flag to tell SDK to run additional queries to populate line item data.
        expand: 'shallow'
    })
    @Type(() => LineItem)
    LineItems: Array<LineItem> = null;
```

Now that you have created your custom product model, mapped type script class to the Salesforce object Product2, and provided custom fields, you must map it back to the service. The product service looks up at the product class. You can override that using a setType method that is available on every single service. This method should be created in the constructor of the component. This changes the mapping of that service to the model that you want to use. Once this is done within your application or within your module, the method gets applied to every other component within your module. Now pass it in the class reference of the class you just created. For details, refer the image below.


EXPLORER
OPEN EDITORS 1 UNSAVED
config.ts src\app U
product-details.component.ts src\app\modules\product-details\layout 1
app.component.ts src\app
s-product.model.ts src\app\models U
header.component.ts src\app\components\header
header.component.html src\app\components\header M
app.module.ts src\app M
variables.scss src M
my-profile.component.ts src\app\components\my-profile U
ECOMMERCE-BASE-TEMPLATE
modules
cart
change-password
configure
home
manage-cart
my-account
order-history
product-details
component
layout
product-details.component.html
product-details.component.scss
product-details.component.spec.ts
product-details.component.ts 1
product-details-routing.module.ts
product-details.module.ts
product-list
services
app-routing.module.ts
app.component.css
config.ts
product-details.component.ts
app.component.ts
s-product.model.ts

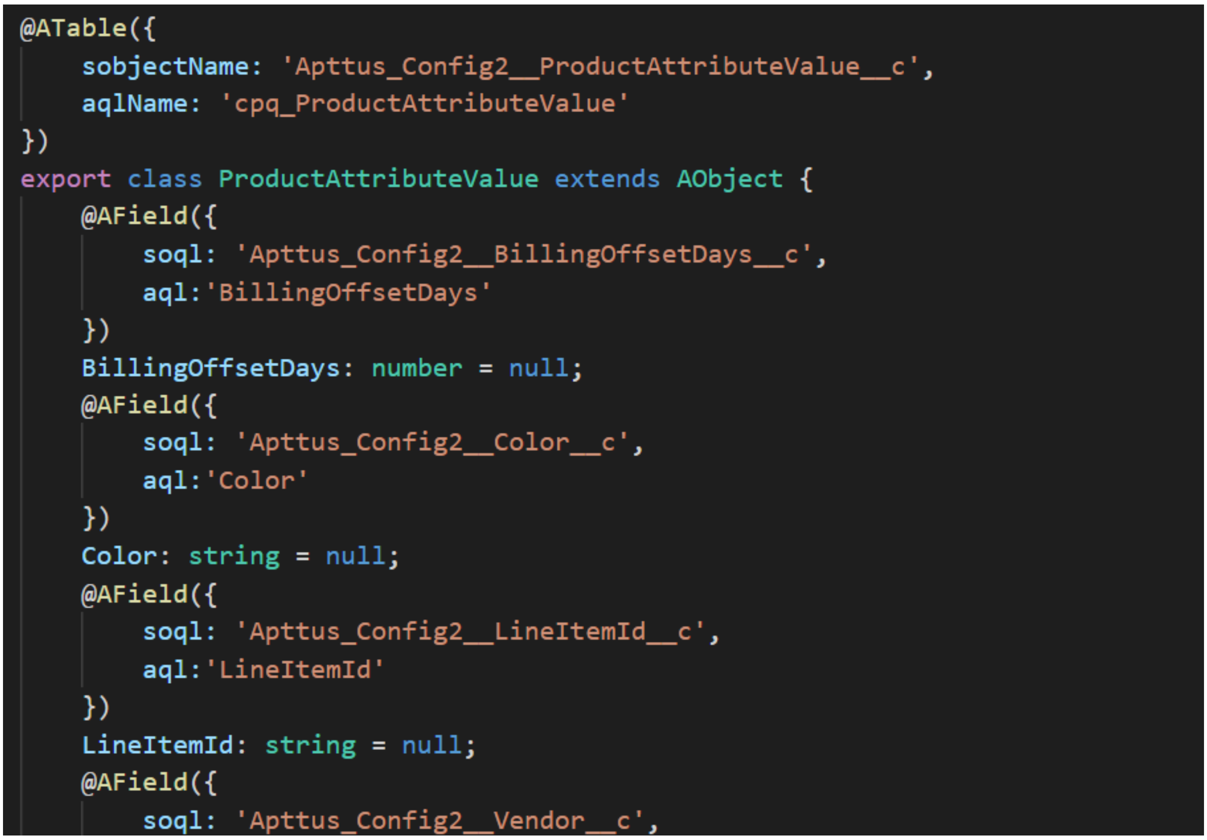
```
import * as _ from 'lodash';
import { Observable } from 'rxjs/Observable';
import { ACondition } from '@apttus/core';
import { SProduct } from '../../../models/s-product.model';


@Component({
  selector: 'app-product-details',
  templateUrl: './product-details.component.html',
  styleUrls: ['./product-details.component.scss']
})
export class ProductDetailsComponent implements OnInit {

  product: Product;
  rules: Array<ConstraintRule>;
  relatedProducts$: Observable<Array<Product>>;
  similarProducts$: Observable<Array<Product>>;

  replacementProducts$: Observable<Array<Product>>;
  replacementRules: Array<any>;
  volumeDiscounts: Array<PriceMatrixEntry>;

  constructor(private route: ActivatedRoute,
              private productService: ProductService,
              private cdr: ChangeDetectorRef,
              private priceMatrixService: PriceMatrixService,
              private constraintRuleService: ConstraintRuleService) {
      this.productService.setType(SProduct);
  }

  ngOnInit() {
    this.route.params
      .filter(params => params['productCode'] != null)
      .map(params => params['productCode'])
      .flatMap(productCode => {
        this.product = null;
        this.rules = null;
```

Adding Custom Attributes to a Product

You can add custom attribute on any product in the product details page through the product attribute model that contains all the product attributes.

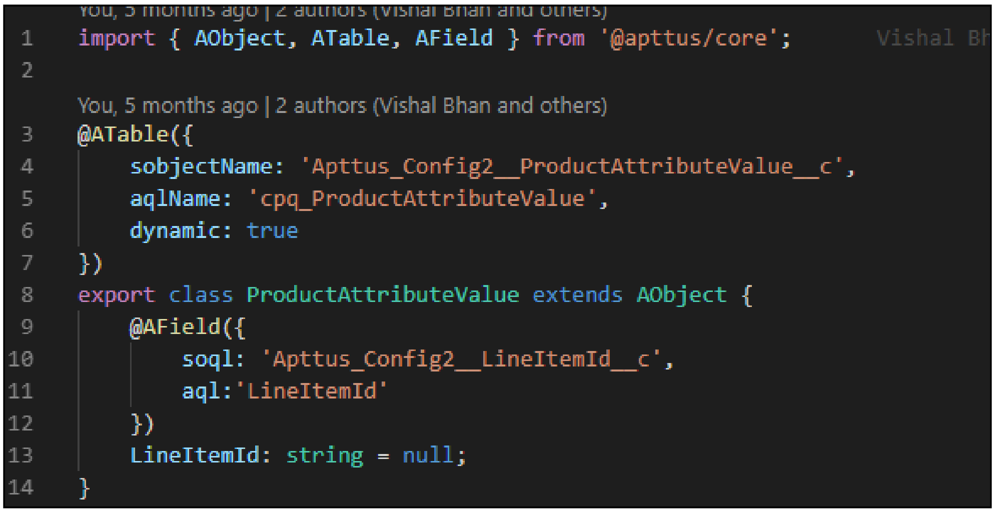
To add a custom attribute to a product

1. You must import the product attribute model from the Apttus E-Commerce library into the product attribute value service.



```
@ATable({
    sobjectName: 'Apttus_Config2__ProductAttributeValue__c',
    aqlName: 'cpq_ProductAttributeValue'
})
export class ProductAttributeValue extends AObject {
    @AField({
        soql: 'Apttus_Config2__BillingOffsetDays__c',
        aql:'BillingOffsetDays'
    })
    BillingOffsetDays: number = null;
    @AField({
        soql: 'Apttus_Config2__Color__c',
        aql:'Color'
    })
    Color: string = null;
    @AField({
        soql: 'Apttus_Config2__LineItemId__c',
        aql:'LineItemId'
    })
    LineItemId: string = null;
    @AField({
        soql: 'Apttus_Config2__Vendor__c',
```

2. In the @Table decorator, you must set the Dynamic flag to *True*.



```
You, 5 months ago | 2 authors (Vishal Bhan and others)
import { AObject, ATable, AField } from '@apttus/core';      Vishal Bh

You, 5 months ago | 2 authors (Vishal Bhan and others)
@ATable({
    sobjectName: 'Apttus_Config2__ProductAttributeValue__c',
    aqlName: 'cpq_ProductAttributeValue',
    dynamic: true
})
export class ProductAttributeValue extends AObject {
    @AField({
        soql: 'Apttus_Config2__LineItemId__c',
        aql:'LineItemId'
    })
    LineItemId: string = null;
}
```

The following objects have the dynamic flag set to true, by default.

a. ProductAttributeValue
b. OrderAttributeValue
c. QuoteAttributeValue
d. AssetAttributeValue

Customizing Logic in the Services

Your application is built around the concept of Modals and Services and pairing the two to work together. You can do one or more of the following:

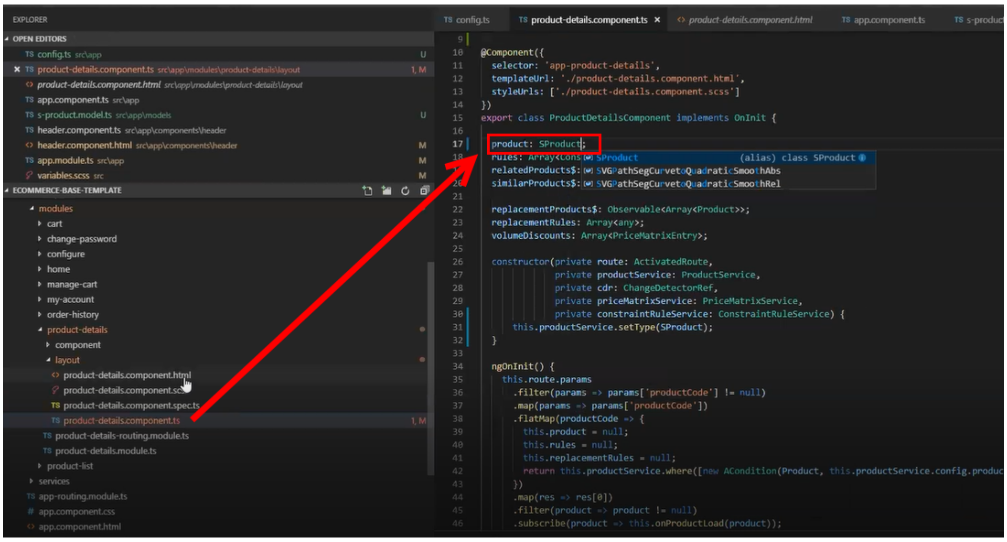
- Change the modal that goes with some business logic and service
- Change the business logic or service for a particular modal
- Create completely new modals and new services

Customizing the Template Page with Custom Field

You can use the custom field you just created in your template.

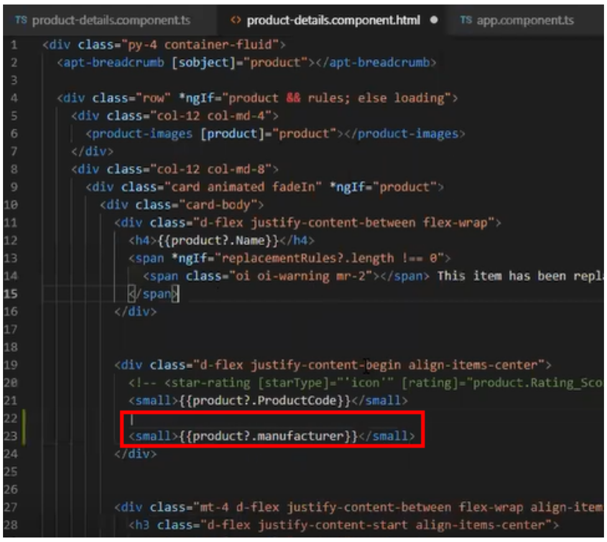
To customize the template page

1. Go to product-details.component.ts and modify the product name with your custom field name.


EXPLORER
OPEN EDITORS
config.ts src\app
product-details.component.ts src\app\modules\product-details\layout
product-details.component.html src\app\modules\product-details\layout
app.component.ts src\app
s-product.model.ts src\app\models
header.component.ts src\app\components\header
header.component.html src\app\components\header
app.module.ts src\app
variables.scss src
ECOMMERCE-BASE-TEMPLATE
modules
cart
change-password
configure
home
manage-cart
my-account
order-history
product-details
component
layout
product-details.component.html
product-details.component.scss
product-details.component.spec.ts
product-details.component.ts
product-details-routing.module.ts
product-details.module.ts
product-list
services
app-routing.module.ts
app.component.css
app.component.html
config.ts
product-details.component.ts
product-details.component.html
app.component.ts
s-product

```
@Component({
  selector: 'app-product-details',
  templateUrl: './product-details.component.html',
  styleUrls: ['./product-details.component.scss']
})
export class ProductDetailsComponent implements OnInit {

  product: SProduct;
  rules: Array<Cons  SProduct                      (alias) class SProduct
  relatedProducts$:  SVGPathSegCurvetoQuadraticSmoothAbs
  similarProducts$:  SVGPathSegCurvetoQuadraticSmoothRel

  replacementProducts$: Observable<Array<Product>>;
  replacementRules: Array<any>;
  volumeDiscounts: Array<PriceMatrixEntry>;

  constructor(private route: ActivatedRoute,
              private productService: ProductService,
              private cdr: ChangeDetectorRef,
              private priceMatrixService: PriceMatrixService,
              private constraintRuleService: ConstraintRuleService) {
    this.productService.setType(SProduct);
  }

  ngOnInit() {
    this.route.params
      .filter(params => params['productCode'] != null)
      .map(params => params['productCode'])
      .flatMap(productCode => {
        this.product = null;
        this.rules = null;
        this.replacementRules = null;
        return this.productService.where([new ACondition(Product, this.productService.config.produc
      })
      .map(res => res[0])
      .filter(product => product != null)
      .subscribe(product => this.onProductLoad(product));
```

2. Go to the product-details.component.html and add an entry for your custom field to be displayed on the template.



```
<div class="py-4 container-fluid">
  <apt-breadcrumb [sobject]="product"></apt-breadcrumb>

  <div class="row" *ngIf="product && rules; else loading">
    <div class="col-12 col-md-4">
      <product-images [product]="product"></product-images>
    </div>
    <div class="col-12 col-md-8">
      <div class="card animated fadeIn" *ngIf="product">
        <div class="card-body">
          <div class="d-flex justify-content-between flex-wrap">
            <h4>{{product?.Name}}</h4>
            <span *ngIf="replacementRules?.length !== 0">
              <span class="oi oi-warning mr-2"></span> This item has been repl
            </span>
          </div>


          <div class="d-flex justify-content-begin align-items-center">
            <!-- <star-rating [starType]="'icon'" [rating]="product.Rating_Sco
            <small>{{product?.ProductCode}}</small>

            <small>{{product?.manufacturer}}</small>
          </div>


          <div class="mt-4 d-flex justify-content-between flex-wrap align-item
            <h3 class="d-flex justify-content-start align-items-center">
```

3. Your template displays the new custom field.

Search Engine Optimization

All B2C customers need the capability to publish their product list and product detail pages to Google search engine so that they can be searchable and linkable. Apttus UI template loads data asynchronously and Angular renders the data retrieved from Salesforce.com.

Follow the steps below to enable Google search to crawl a Digital Commerce website by creating a sitemap index, dynamic sitemap's and register them with Google.

Step 1: Create a robots.txt file and set it in the Community page

```
<apex:page contentType="text/plain" showHeader="false">
User-agent: *
Disallow: /<store name>/cart?
Disallow: /<store name>/manage-cart?
Disallow: /<store name>/Orders?
Disallow: /<store name>/installed-products?
Disallow: /<store name>/my-account?
Sitemap: http://<domain>/<store name>/APTSSiteIndex
</apex:page>
```

Step 2: Create a APTSSiteIndex Visualforce page

Create a APTSSiteIndex Visualforce page that renders the XML as below.

Pre-requisite: There must be 2 existing sitemaps, one for all products and one for all categories.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://<domain>/<store name>/APTSSiteMapProducts</loc>
    <lastmod>2019-10-06</lastmod>
  </url>
  <url>
    <loc>http://<domain>/<store name>/APTSSiteMapCategories</loc>
    <lastmod>2019-10-06</lastmod>
  </url>
</urlset>
```

Step 3: Create a APTSSiteMapProducts Visualforce page

Create a APTSSiteMapProducts VF page that will generate all the Active Products. Each Sitemap can only have 10K urls.

Sample Output

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://<domain>/<store name>/APTSProduct?id=<productcode></loc>
    <lastmod>2019-10-06</lastmod>
  </url>
</urlset>
```

The APTSProduct renders the following html data.

HTML Data

```
<html>
<head>
<title>Executive Anvil</title>
<script type="application/ld+json">
{
  "@context": "https://schema.org/",
  "@type": "Product",
  "name": "Executive Anvil",
  "url": "https://example.com/anvil",
  "image": [
    "https://example.com/photos/1x1/photo.jpg",
    "https://example.com/photos/4x3/photo.jpg",
    "https://example.com/photos/16x9/photo.jpg"
   ],
  "description": "Sleeker than ACME's Classic Anvil, the Executive Anvil is perfect for the business traveler looking for something to drop from a height.",
  "sku": "0446310786",
  "offers": {
    "@type": "Offer",
    "url": "https://example.com/anvil",
    "priceCurrency": "USD",
    "price": "119.99",
    "priceValidUntil": "2020-11-05"
  }
}
</script>
</head>
<body>
</body>
</html>
```

Useful Tips for Turbo Enabled Catalog

Digital Commerce catalog leverages Turbo enabled catalog APIs for better performance. For optimal results, you must keep the following in mind:

- While implementing the Digital Commerce application, you can choose the product fields based on which the catalog search works. For better performance, you must limit the number of searchable fields to 3.

- The performance may degrade if long text fields like Product Description are made searchable.
- It is recommended to keep the number of custom fields on the product model less. Too many custom fields on the product model may degrade performance.

Frequently Asked Questions (FAQs)

How secure is the data stored in JavaScript cache from vulnerability? What kind of security standards are used in handling cache?

All caching happens in JavaScript memory. Hashing and compression is used with garbage collection. JavaScript memory is cleared out as soon as user leave browser session. We do not store any data in the local storage in browser such as cookies etc.

How is the data transmission secured between client API and Server Rest API?

Apttus Digital Commerce SDK uses encrypted communication between client and server using AES 128 encryption. Also have timestamped endpoint and are valid only for 30 seconds. If it takes more time then request is not valid.

How is data protected from Cross site Scripting (XSS)?

To systematically block XSS bugs, Apttus Digital Commerce treat all values as untrusted by default. When a value is inserted into the DOM whether its from a template, property, attribute, binding or interpolation, angular sanitizes and escapes untrusted values.

How is data protected from SQL Injection attack?

Digital Commerce sanitizes user input before it is converted to query and send to server such as escape special characters etc.

How secure is the user login from Angular to Salesforce Server? How is the session's secured from any intruder attack?

Use standard salesforce standard authentication using access token. Use standard OAuth protocol to generate access token. Also go through communities security for data access.

How secure are the queries triggered from Angular Client API to Salesforce for Search, Product details etc..?

All user input is sanitized and custom parser in SDK is used to generate queries. Salesforce E-Commerce interface has further validation on SQL queries, on server side before data is accessed from Salesforce.

What best practices should be followed, when SDK components & services are extended or customized to avoid exposing any security vulnerability?

Setup strong security model as the salesforce org level using standard salesforce security framework (roles, profiles, permission sets etc.). SDK does not circumvent Salesforce security in any way.

Use standard SDK mechanisms for extending services, models, interface for any customization without circumventing the SDK architecture to integrate and interface with Salesforce.

Apttus Contact Support

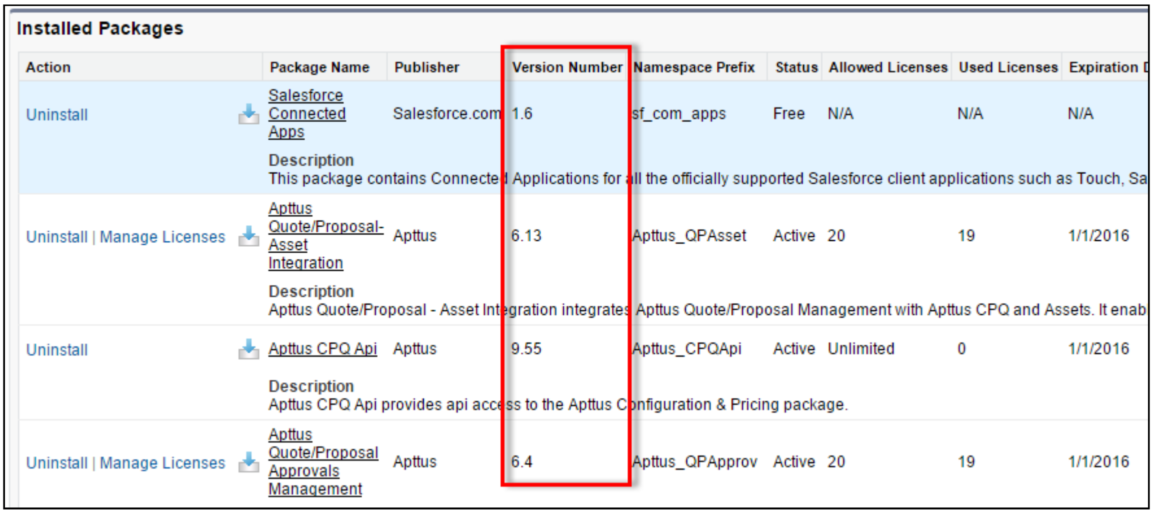
If you experience an issue with an Apttus product and need help, you can contact Apttus Support. Before you contact Apttus support, prepare a brief description of the problem you are experiencing. Additionally, to enable us to resolve your problem at the earliest, provide the following important information:

- What is the environment in which you are experiencing the problem: Sandbox or Production?
- How many users are affected?

Which product versions are installed?

To determine version numbers:

1. Go to **Setup** > **App Setup** > **Installed Packages**.
2. In the Installed Packages section, all the installed packages are displayed. You can find the version numbers in the Version Number column.



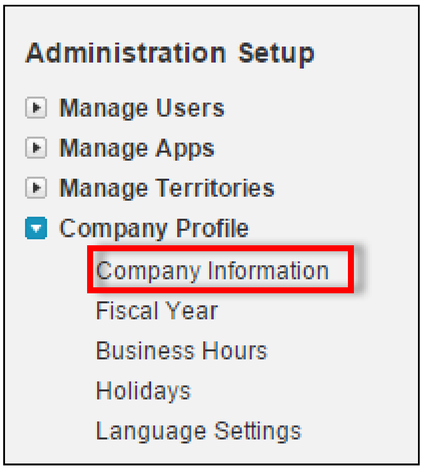
Installed Packages

Action	Package Name	Publisher	Version Number	Namespace Prefix	Status	Allowed Licenses	Used Licenses	Expiration D
Uninstall	Salesforce Connected Apps	Salesforce.com	1.6	sf_com_apps	Free	N/A	N/A	N/A
	Description This package contains Connected Applications for all the officially supported Salesforce client applications such as Touch, Sa							
Uninstall | Manage Licenses	Apttus Quote/Proposal-Asset Integration	Apttus	6.13	Apttus_QPAsset	Active	20	19	1/1/2016
	Description Apttus Quote/Proposal - Asset Integration integrates Apttus Quote/Proposal Management with Apttus CPQ and Assets. It enab							
Uninstall	Apttus CPQ Api	Apttus	9.55	Apttus_CPQApi	Active	Unlimited	0	1/1/2016
	Description Apttus CPQ Api provides api access to the Apttus Configuration & Pricing package.							
Uninstall | Manage Licenses	Apttus Quote/Proposal Approvals Management	Apttus	6.4	Apttus_QPApprov	Active	20	19	1/1/2016

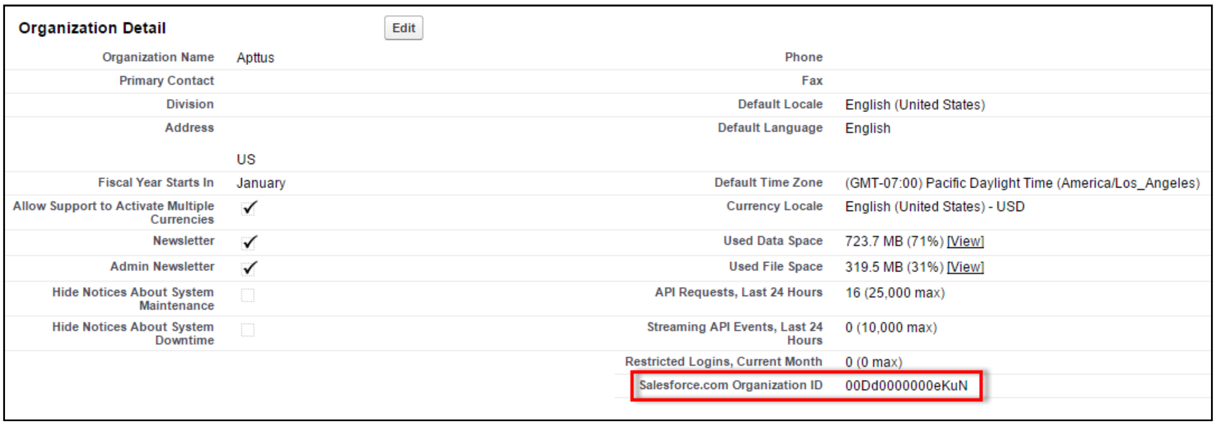
What is your Salesforce.com Organization ID?

To determine the Salesforce.com organization ID:

1. Go to **Setup** > **Administration Setup** > **Company Profile** > **Company Information**.



2. From the Organization Detail pane, provide the Salesforce.com Organization ID.



Organization Name	Apttus	Phone	
Primary Contact		Fax	
Division		Default Locale	English (United States)
Address	US	Default Language	English
Fiscal Year Starts In	January	Default Time Zone	(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)
Allow Support to Activate Multiple Currencies	✓	Currency Locale	English (United States) - USD
Newsletter	✓	Used Data Space	723.7 MB (71%) [View]
Admin Newsletter	✓	Used File Space	319.5 MB (31%) [View]
Hide Notices About System Maintenance	□	API Requests, Last 24 Hours	16 (25,000 max)
Hide Notices About System Downtime	□	Streaming API Events, Last 24 Hours	0 (10,000 max)
		Restricted Logins, Current Month	0 (0 max)
		Salesforce.com Organization ID	00Dd0000000eKuN

If you are having issues generating documents, what is your merge server end point?

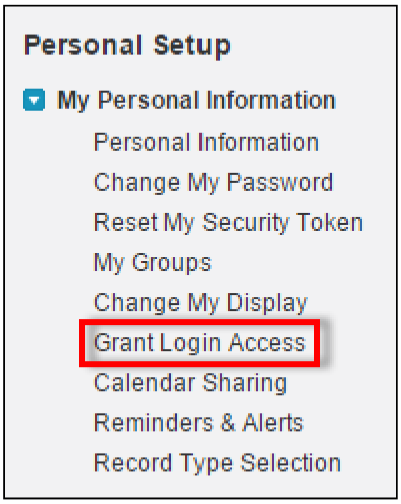
To find the merge server end point:

1. Go to **Setup** > **App Setup** > **Develop** > **Custom Setting**s.
2. Click **Manage** for Comply System Properties.
3. Click **System Properties**.
4. The Merge Webservice Endpoint field displays the setting. The https://mergeserver.apttus.net:9876 portion of the setting is what will be helpful to customer support.

Grant Login Access of the affected user and an administrator.

To grant login access:

1. Go to **Setup** > **Personal Setup** > **My Personal Information** > **Grant Login Access**.

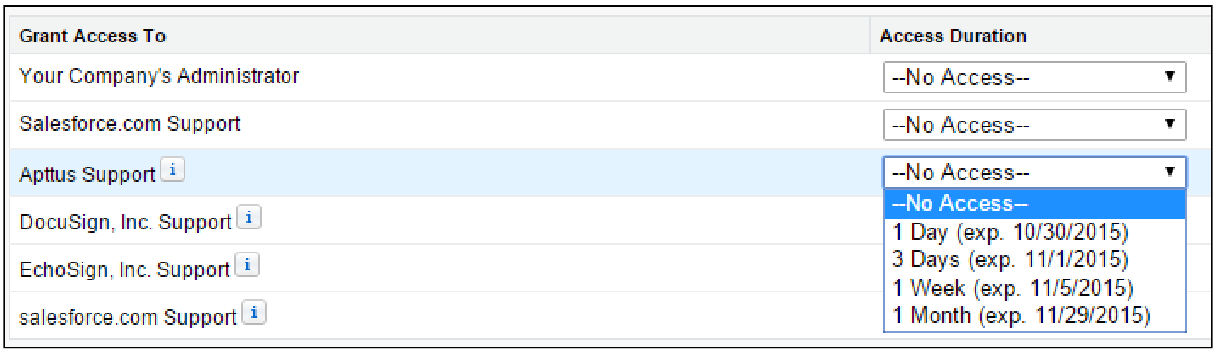


Personal Setup

My Personal Information

- Personal Information
- Change My Password
- Reset My Security Token
- My Groups
- Change My Display
- Grant Login Access
- Calendar Sharing
- Reminders & Alerts
- Record Type Selection

2. From the Apptus Support picklist, select an option for access duration.



Grant Access To	Access Duration
Your Company's Administrator	--No Access--
Salesforce.com Support	--No Access--
Apttus Support	--No Access-- --No Access-- 1 Day (exp. 10/30/2015) 3 Days (exp. 11/1/2015) 1 Week (exp. 11/5/2015) 1 Month (exp. 11/29/2015)
DocuSign, Inc. Support	
EchoSign, Inc. Support	
salesforce.com Support	

3. Click **Save**.

Apttus Copyright Disclaimer

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: DCOMSFSUM20P1IDGREVA20201021