APTTUS®

# Digital Commerce on Salesforce Summer 2019 Implementation and Deployment Guide

REV B

02/07/2020

# 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.<br>• 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.

| Release | Topic | Description |
|---|---|---|
| Summer 2019 Rev B | Managing Permissions for Secure Guest User Access to Storefront | New topic. New enhancement introduced in Summer 2019 Patch 13. |
| 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<br>• Tax Integration with Avalara Tax Engine<br>• Setting Up Multi-language Using Translations<br>• 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

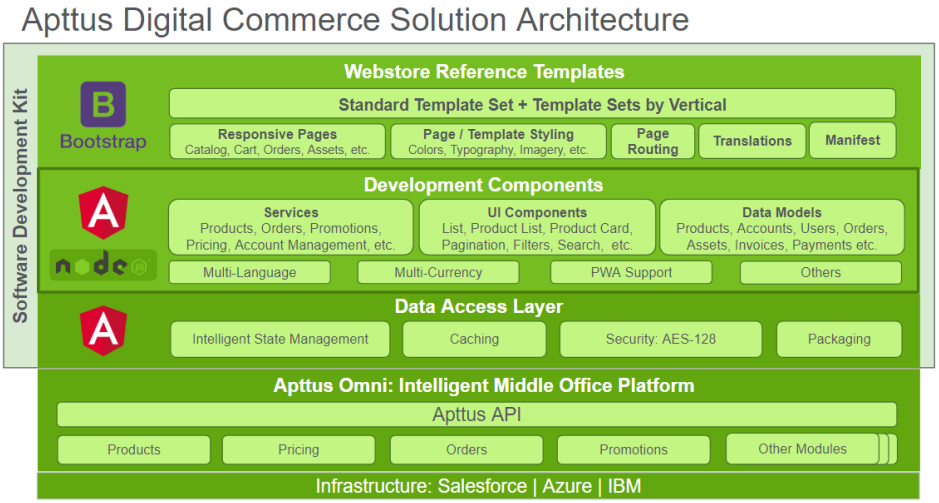
Apttus Digital Commerce empowers enterprises to quickly evolve and scale global Omni-Channel selling strategies. Apttus Digital Commerce provides enterprises with the agility, speed and scale to create unique, valuable customer experiences that drive greater brand consistency and revenue. From streamlining and simplifying complex selling scenarios involving Configure Price Quote to Partner Commerce, Apttus Digital Commerce scales across diverse selling strategies quickly.

**Quickly build and launch new sites**: Create professional, high-performance, mobile-enabled websites that deliver up-to-the-minute dynamic content in multiple languages and currencies - no additional IT resources needed.

**Scale with demand**: Scale sophisticated product catalogs, pricing and promotional programs across regions and channels, with automation and administrative capabilities that ensure consistency and precise execution.

**Update with clicks, not code**: With click to configure technology, your CRM administrator can make strategic updates in minutes. Enterprises can define catalogs, products, service and associated options and promotions once then replicate them across multiple Digital Commerce sites.

## Apttus Digital Commerce Solution Architecture




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

- Install Apttus E-Commerce Package and dependent packages
- Set up and activate communities
- Create custom field on account for the price list
- 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
- Set up single and multiple store
- Post deployment community setup
- Assign Apttus E-Commerce permission set
- Manage permissions for secure guest user access
- Payment integration
- Tax integration with Avalara tax engine
- Set up multi-language using translations
- Set email notification template for checkout
- Enable state and country picklists

## 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. |
| 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. |

| Term | Description |
| --- | --- |
| 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. |

# 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 Configuration & Pricing | CPQ | Y |
| 5 | Apttus Quote/Proposal-Configuration Integration | Integrations | Y |
| 6 | Apttus CPQ Admin | CPQ | Y |
| 7 | Apttus CPQ API | CPQ | Y |
| 8 | Apttus Order Management | CPQ | Y |

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

# E-Commerce Package Objects

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

| Apttus E-Commerce Object | Purpose of the Object |
| --- | --- |
| Storefront | Represents the storefront details. |
| Storefront Banner | Represents the storefront banner details. |

# Logging in to Apttus Digital E-Commerce

Log in to your Salesforce.com org.

> **Note**
> Do not use the Back button on your browser.

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.

# Setting Up Communities

In order to deploy the Digital Commerce code, you must set up and enable a 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 (i.e home, login, forgot password, change password etc). Being that its a single page application, it is designed to handle all incoming requests.

## To set up a community

1. Go to **Setup > Customize > Communities > Communities Settings** and select **Enable communities**.
2. Under the select a domain name section, type a domain name for your community and click **Check Availability** to see if the domain name is available.
3. If you see a success message, click **Save**.
4. The Communities page is refreshed and displays All Communities section. Click **New Community**.
5. Select from one of the Standard Community templates. For example, Salesforce Tabs + Visualforce.
6. Click **Get Started**.
7. In **Name**, type a name for your community. **URL** displays the domain name of your community, in **Optional**, type a suffix for your community. Click **Create**. For example, Name = E-Commerce, Optional = ecomm.

Your community is created.

## 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.

# 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.

# 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

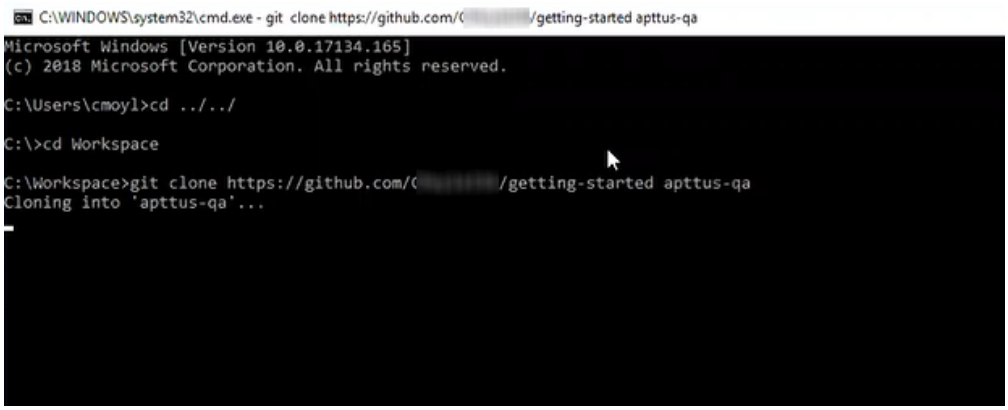
1. In the Storefront record, from the Note & Attachments related list, click **New Note**.
2. For **Title**, enter a title of the Banner image.
3. Click **Save**.
4. Click **Attach File** to attach a banner image and follow the instructions.

# 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.




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

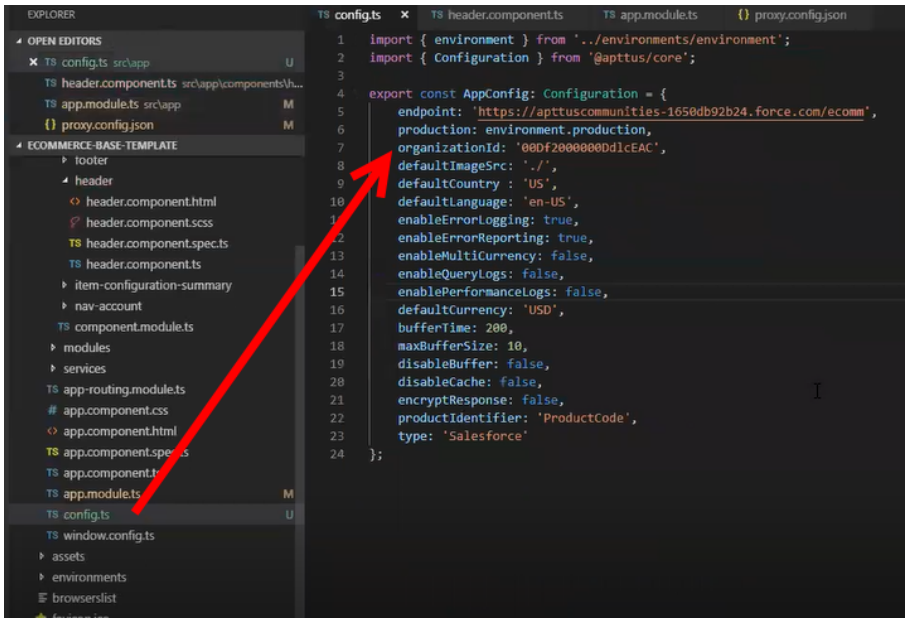
As part of the setup, sf.json file is created that contains the credentials for deploying the application. If you want to deploy your Storefront in a different org, you can update the credentials in this sf.json file.

## 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:

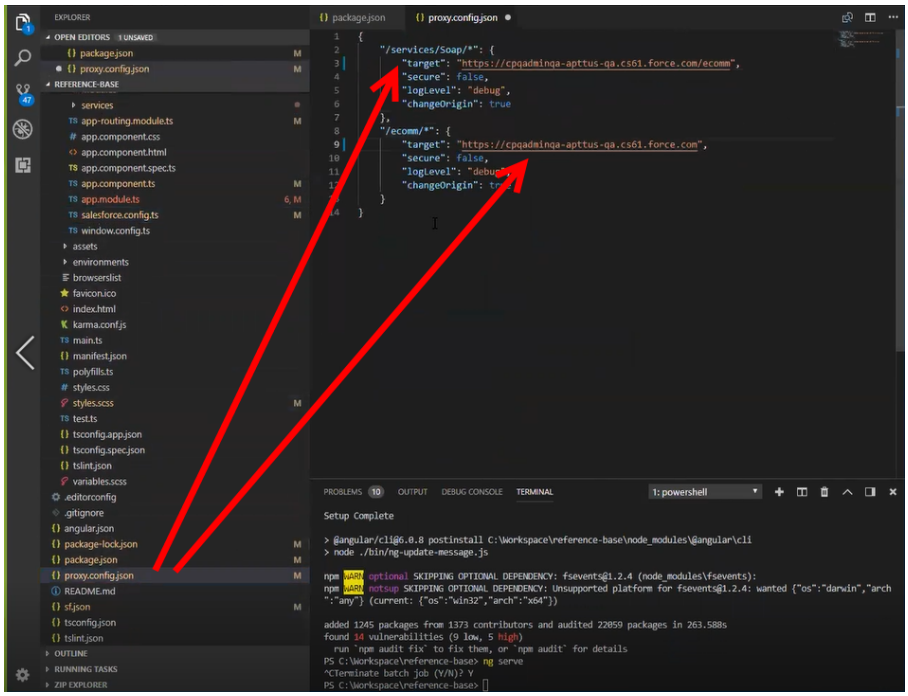
| Parameter | Type | Description |
|---|---|---|
| endpoint | String | Type the community URL to point the application to. |
| production | True / False | Specify the environment where you want the application to run. |
| organizationId | String | Specify the org Id of the application. For details, see What is your Salesforce.com Organization Id? section. |
| defaultImageSrc | String | Specify the URL of the default image to use when no image is found. |
| defaultCountry | String | This is optional. The default country code for guest users. By default, it is "US". |
| defaultLanguage | String | This is optional. The default locale of the guest users. By default, it is "en-US". |
| enableErrorLogging | True / False | This is optional. Set this to True, in non-production mode, to send error logs to Apttus. |
| enableErrorReporting | True / False | This is optional. When set to True, in non-production mode, shows a modal window to provide user feedback to Apttus. |
| enableMultiCurrency | True / False | If using a multi-currency enabled org, set to true to enable currency fields on models. |
| enableQueryLogs | True / False | Set to true to print query requests and results in the browser console. |
| enablePerformanceLogs | True / False | Set to true to print performance metrics of requests in the browser console. |
| defaultCurrency | String | The default currency to use for guest users. Defaults to USD. |
| bufferTime | number | 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 | The maximum number of requests to batch into a single callout. |
| disableBuffer | True / False | When set to true, will disable buffered requests entirely. 1 request = 1 network callout. |
| disableCache | True / False | When set to true, data returned from requests will not be cached. |

| Parameter | Type | Description |
|---|---|---|
| encryptResponse | True / False | When set to true, responses from the server will be encrypted.<br>⚠ Encryption payload is limited to 1 MB. |
| productIdentifier | String | The API name of the field on the product to use as the unique identifier in the application. |




## 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.



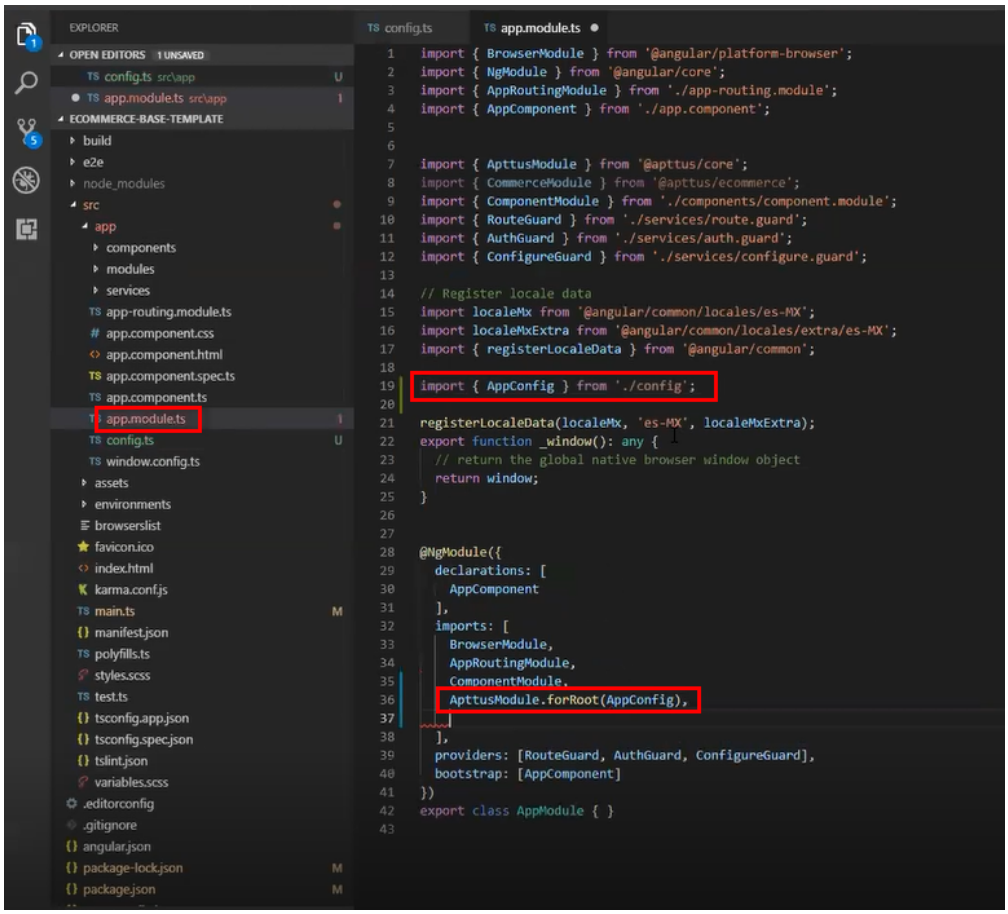

All the code for the templates is open sourced into the library we just cloned and installed. It consists of all the modules such as cart page, home page, account page and more. We can go into any template, for example, Home page layout where you can see some of the components of the Apttus underlying 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

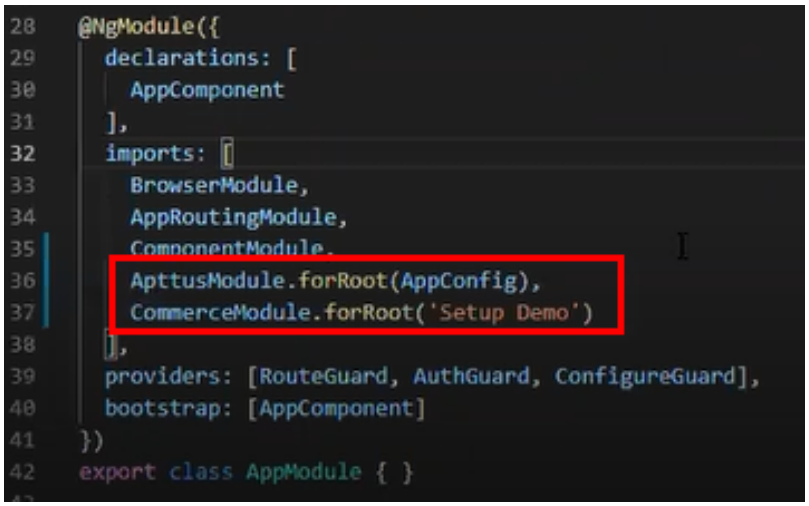
Add two lines for the following:

- 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.



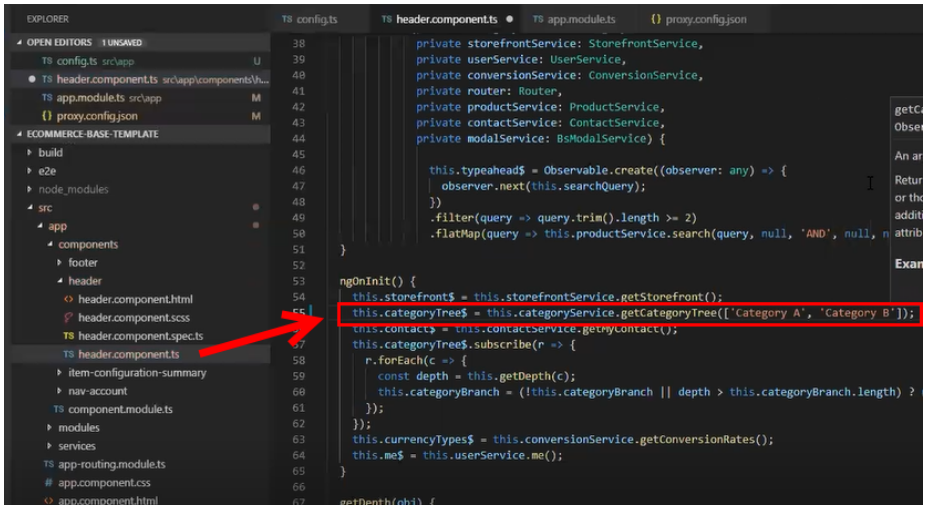

- CommerceModule.forRoot - In the forRoot method of the CommerceModule declare the storefront that you want to use. For example, CommerceModule.forRoot('Tier 1')

This defines the Storefront for your application.



## Setting Up Subset of Categories

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




## 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.

In order to deploy on your Salesforce org, go to your code and run the npm run deploy command. This command retrieves everything you built and run it through a web pack and compress it down, get it production ready and then deploy it to a Visualforce and static resource page in your org.

## To deploy your application on salesforce org

1. Go to your code, type **npm run deploy** and press **Enter**.
2. For **Which angular project are you using?**, type the name of your project. For example, reference-base.
3. For **What is the name of the visualforce page you would like to deploy to?**, type the name of your visualforce page. For example, gettingstarted.
4. For **What is the name of the static resource you would like to deploy to?**, type the name of your visualforce page. For example, gettingstarted.

After successful completion, the system displays a message that your application is ready and provides a link. Click the link to see how your application looks in your Salesforce org.

The application deployment has a 4 minute time limit after which it times out. You can go to your Salesforce org and see your deployment in progress from **Setup > Deployment Status**.

You can also verify your deployment from the package and dist folder.

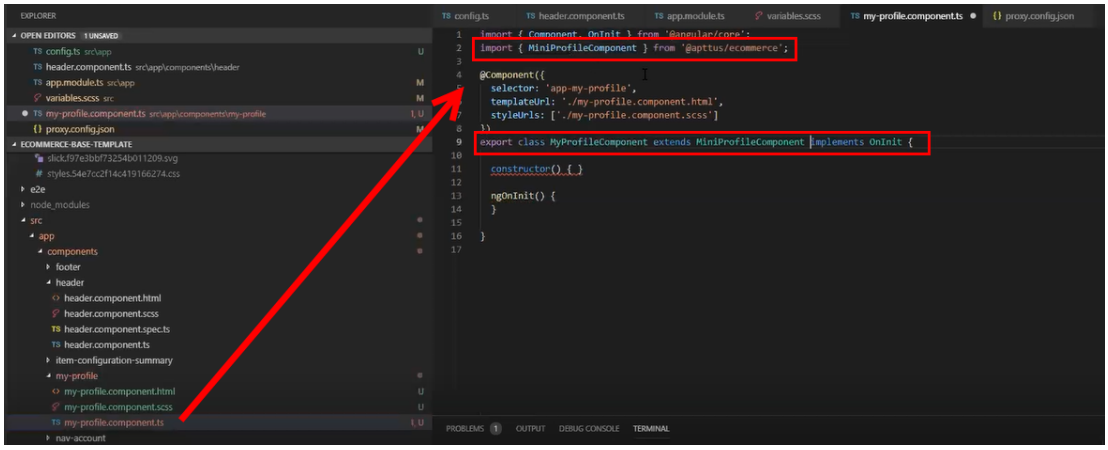
# Customizing Your Application

## Customizing HTML Content and Standard Components

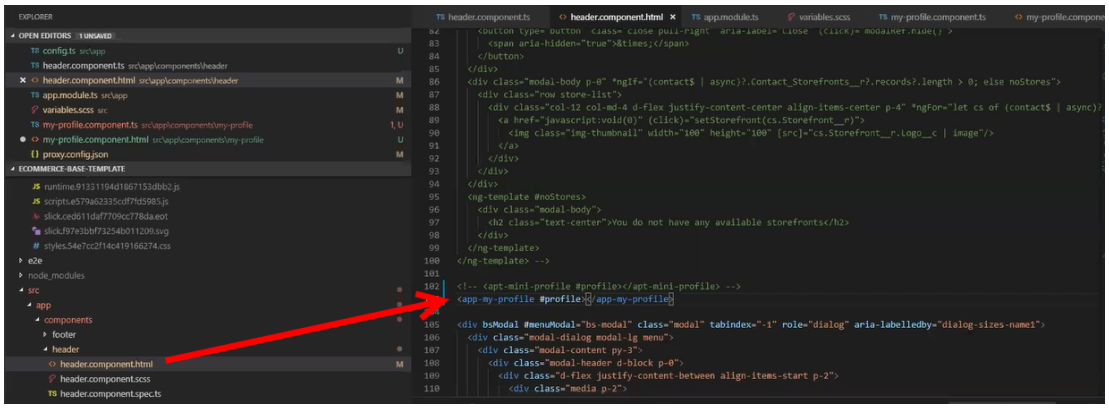
You can override the HTML for any component that comes with your angular application. You can do so by writing an extension component. Similarly, you can override the methods within it as well. For example, there is a registration method and you want to retain it but want to change the look and feel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




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.



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>
```

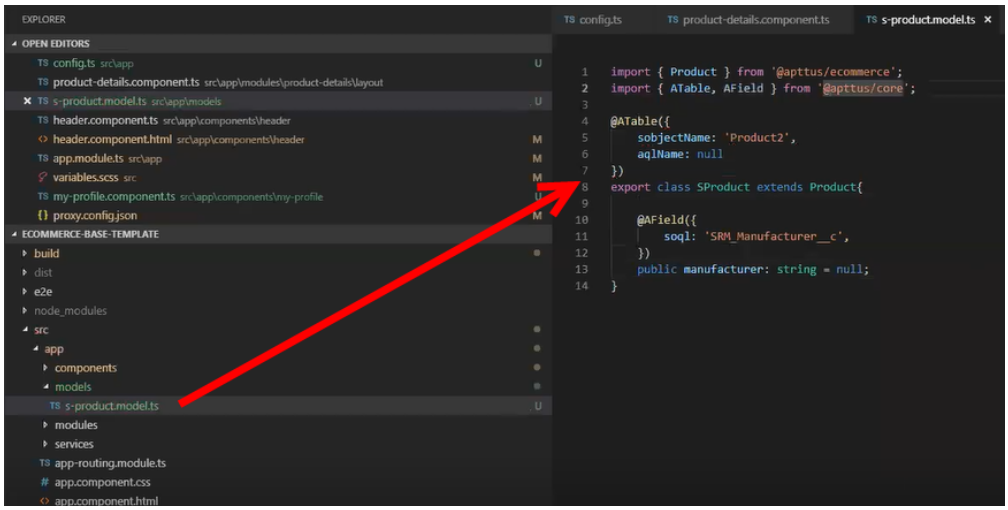
# Adding Custom Fields on Object Models

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.

Lets see how to add a custom field to the Product object.

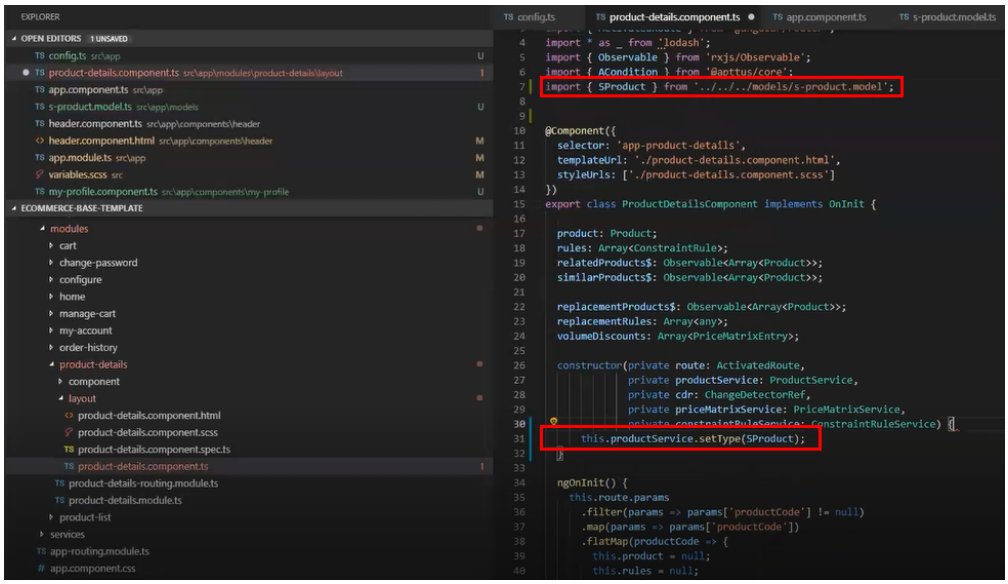
## To create a custom model

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.




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 type script 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 field sobjectName whereas for AIC it can be added to aqlName, as shown in the image above. This will make a unified system between Salesforce and AIC. If you dont have an aqlName, you can specify null.
    - 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.

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.




# 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 product attribute value service, do the following:
   a. Set the context in onInit ( ) where type=ProductAttributeValue.
   b. Call this.generateModel ( ) method that is present in the Apttus core library. This takes care of all the custom attributes.

```
export class ProductAttributeValueService extends AObjectService{
    type = ProductAttributeValue;

    onInit(){
        this.type = ProductAttributeValue;
        this.generateModel();
    }
}
```

3. Finally, inject this service in the constructor method in the product details component to call this.generateModel ( ) method on the service.

# 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:

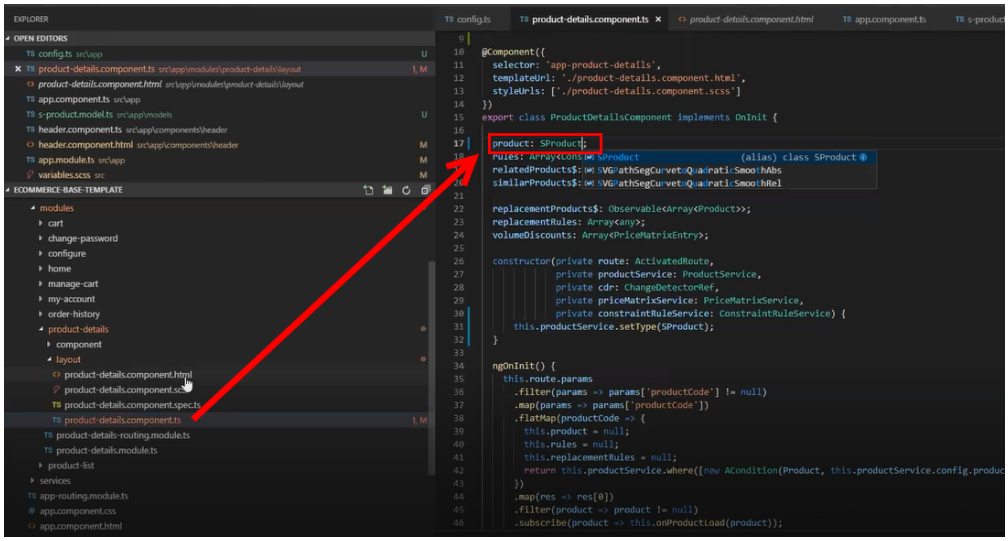
- Change the modal that goes with some business logic and service
- Change the business logic or service for a particular modal
- Create completely new modals and new services

# 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.




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

```
TS product-details.component.ts    <> product-details.component.html ●    TS app.component.ts
1   <div class="py-4 container-fluid">
2     <apt-breadcrumb [sobject]="product"></apt-breadcrumb>
3
4     <div class="row" *ngIf="product && rules; else loading">
5       <div class="col-12 col-md-4">
6         <product-images [product]="product"></product-images>
7       </div>
8       <div class="col-12 col-md-8">
9         <div class="card animated fadeIn" *ngIf="product">
10          <div class="card-body">
11            <div class="d-flex justify-content-between flex-wrap">
12              <h4>{{product?.Name}}</h4>
13              <span *ngIf="replacementRules?.length !== 0">
14                <span class="oi oi-warning mr-2"></span> This item has been repl
15              </span>
16            </div>
17
18
19            <div class="d-flex justify-content-begin align-items-center">
20              <!-- <star-rating [starType]="'icon'" [rating]="product.Rating_Sco
21              <small>{{product?.ProductCode}}</small>
22
23              <small>{{product?.manufacturer}}</small>
24            </div>
25
26
27            <div class="mt-4 d-flex justify-content-between flex-wrap align-item
28              <h3 class="d-flex justify-content-start align-items-center">
```

3. Your template displays the new custom field.

# 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.

# 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.

### 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 permissoin 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.

# Payment Integration

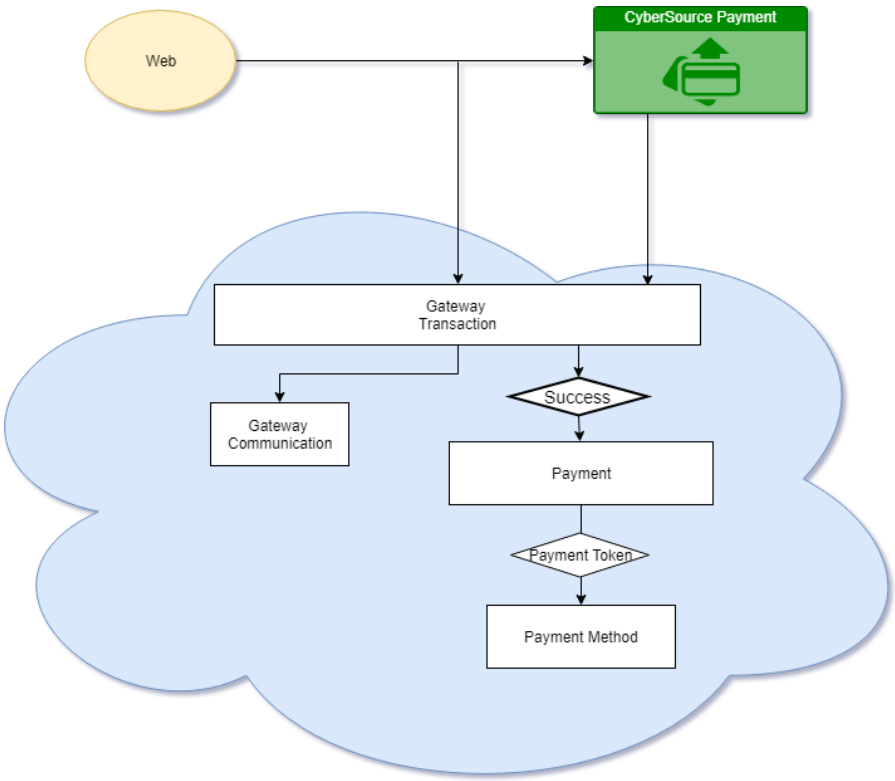
**Pre-requisite**
You must have already installed Apttus CPQ and Apttus Billing Management packages.

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 order. |

| Object Name | Purpose |
|---|---|
| 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 via 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 Secure Acceptance Method

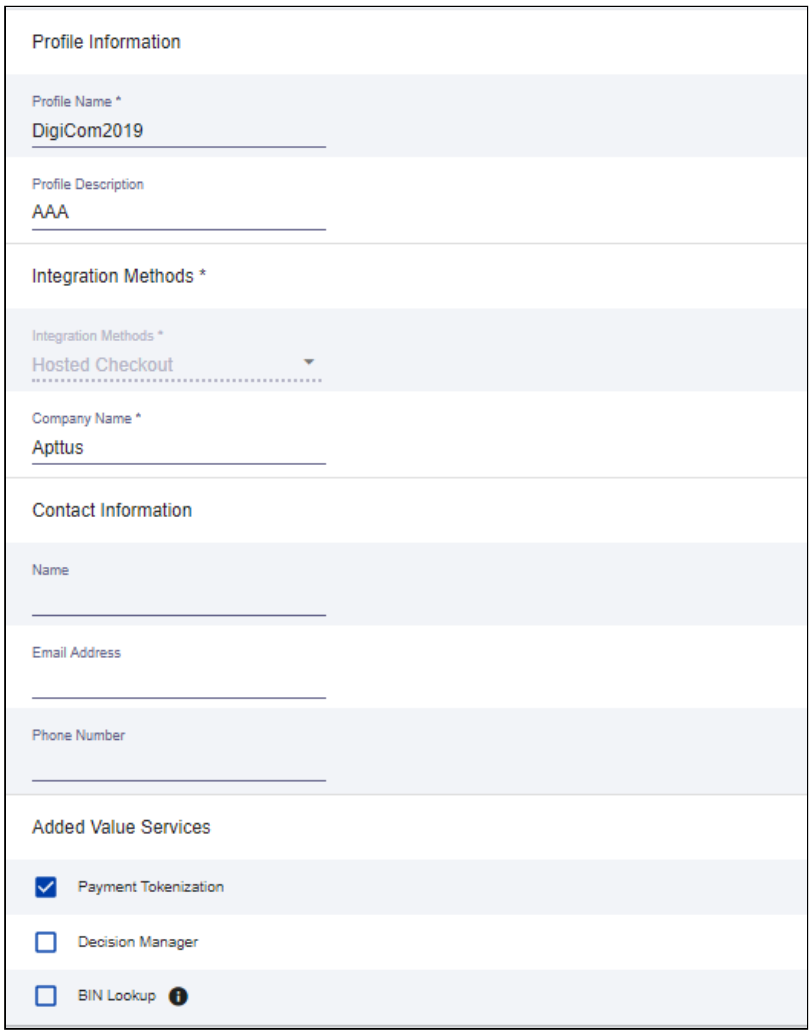
> **Pre-requisite**
> You must subscribe for a Cybersource subscription account.

In order 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.




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 in order 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 option is 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 i.e 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

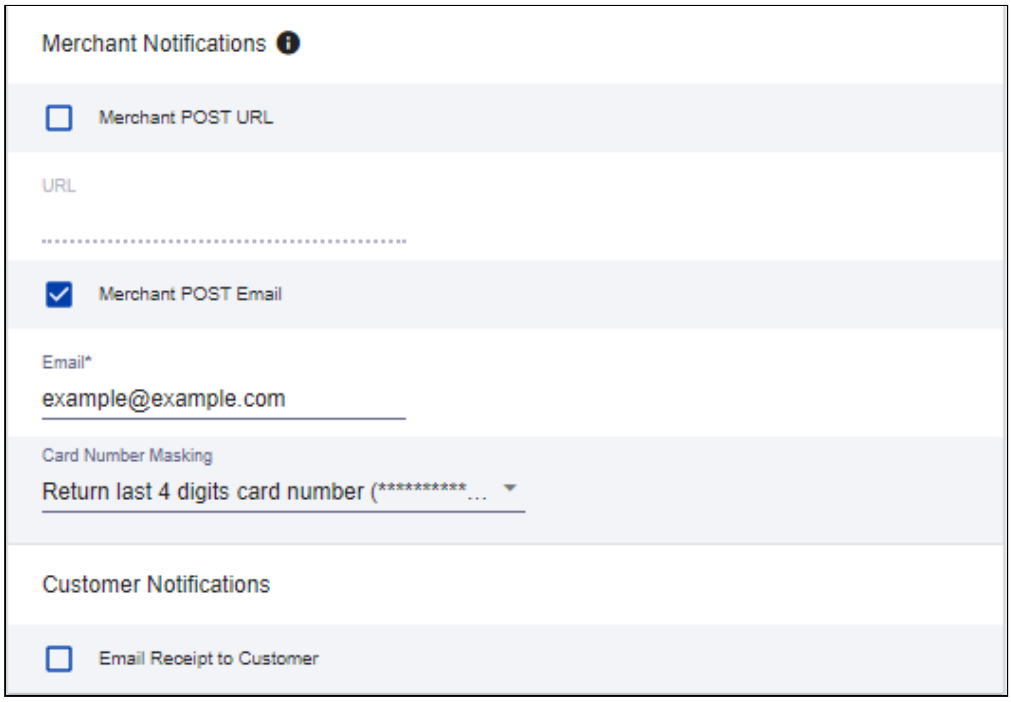
You must create your access key and security key that is needed for transactions.

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 via email or a post on API. Currently Email notifications is supported.




## Configuring Customer Responses

Customer response setting allows you to configure the redirect page once 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

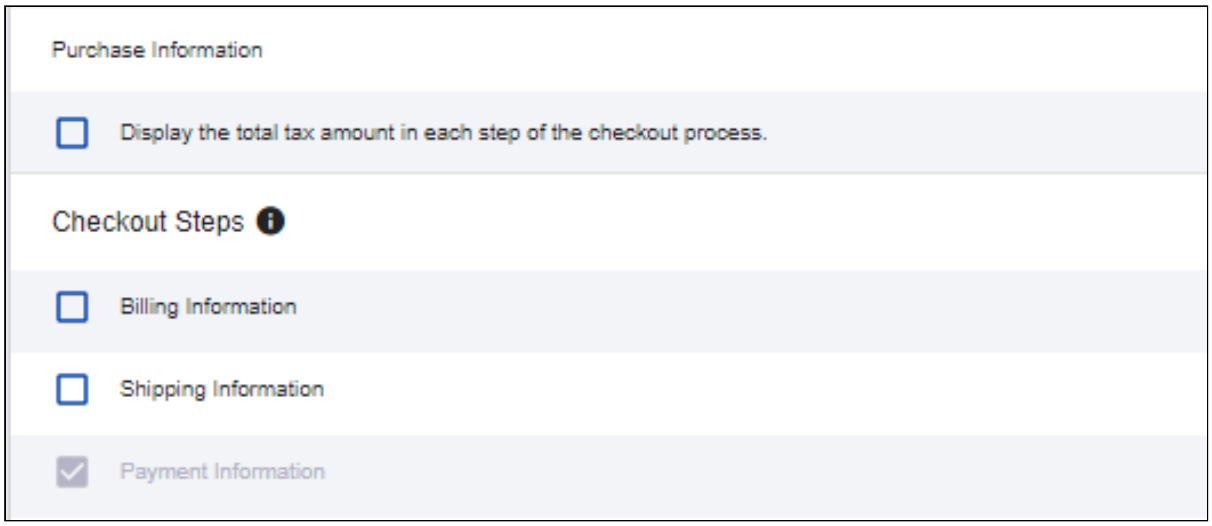
You must configure your cybersource secure acceptance page from the Payment Form tab.

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

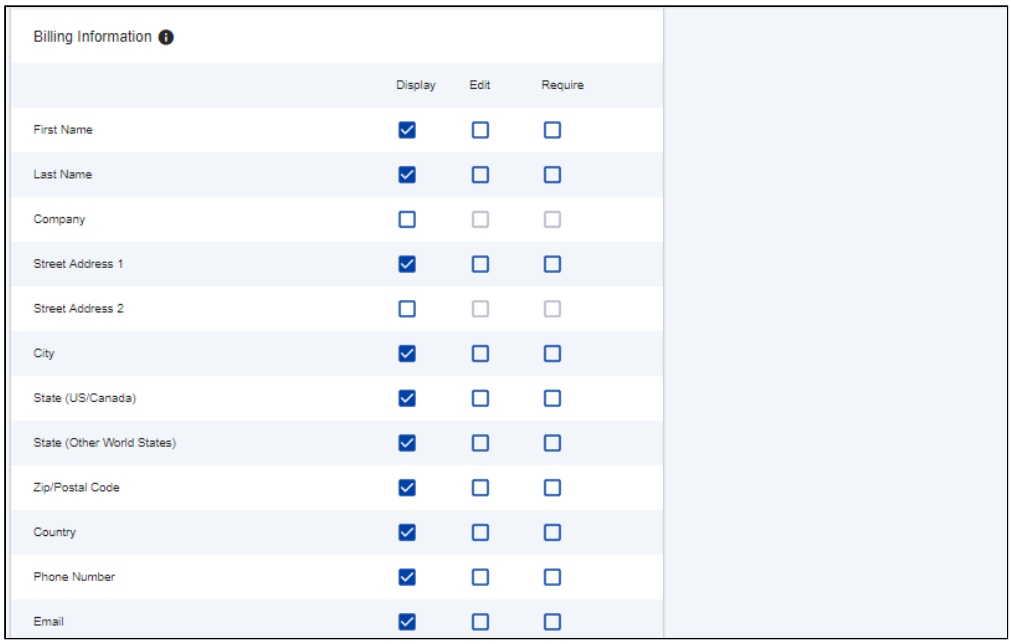
Payment Form Flow

- ○ 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.

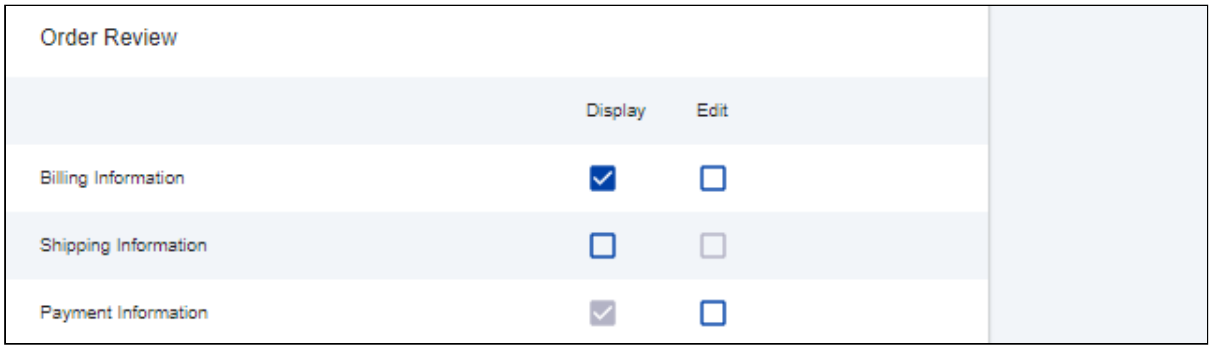



4. Select relevant information to display on billing page.




5. Select relevant information to display on echeck page.


6. Configure the settings for review page.




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.

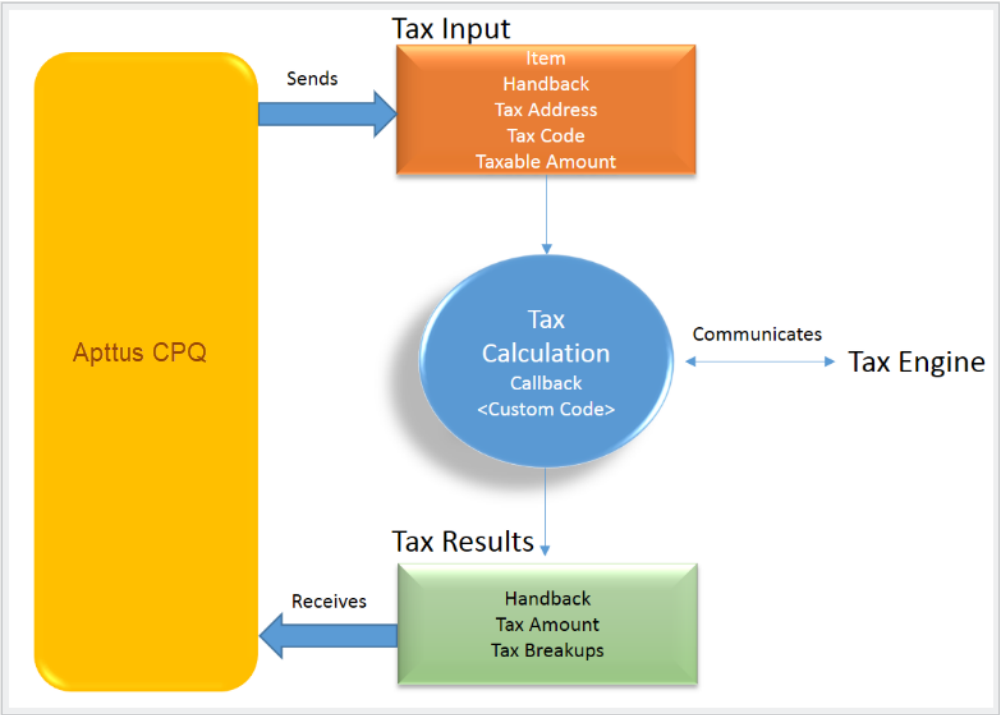
### To define custom labels

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 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 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 will contain 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 *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

> **Pre-requisite**
> You must have already installed Apttus CPQ packages.

In order 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. Click 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 you 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.

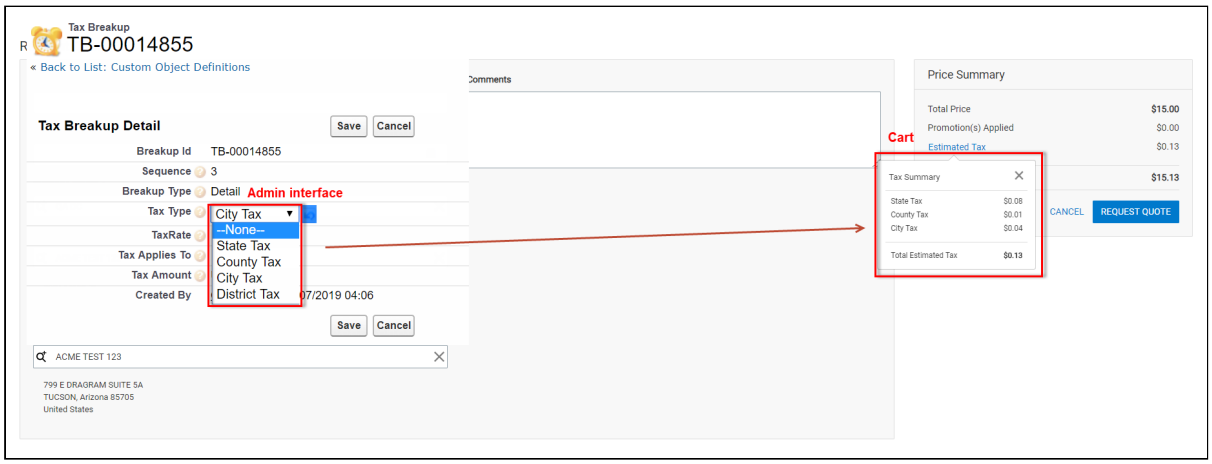
9. Select the Effective Date and Expiration Date for the tax certificate.
10. Click Save. You can define the newly created tax certificate in your Account object.

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.

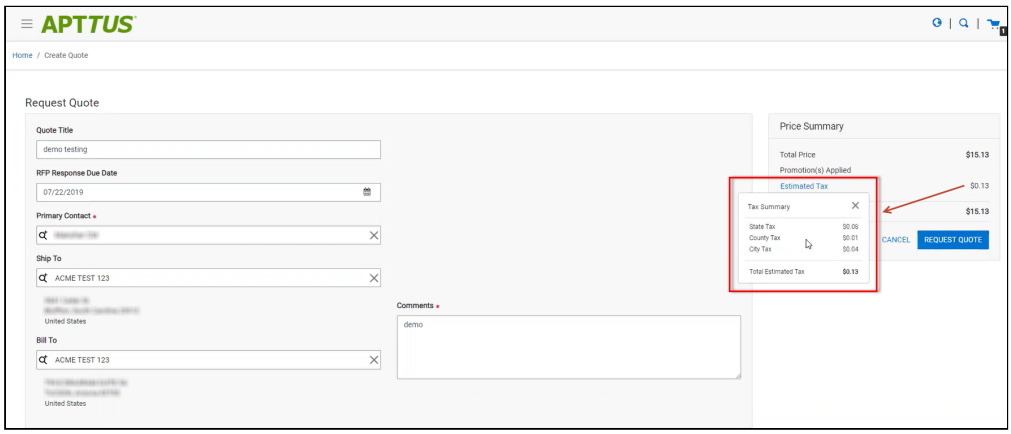



Now when the order is generated, you get order tax breakup for orders and proposal tax break up for proposals. These tax breakups are attached to respective line items.

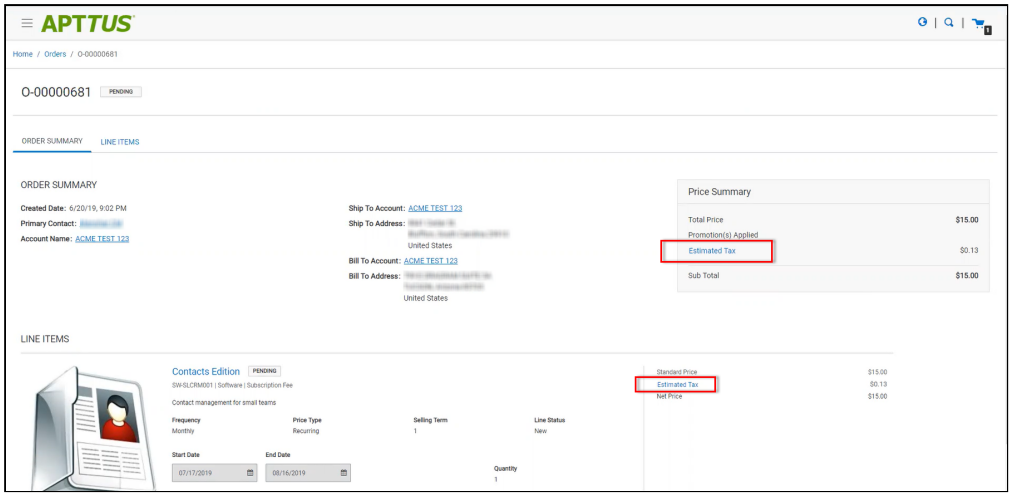
You must ensure that the Account object has a valid Billing and Shipping Address and the Tax certificate is defined.

**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.




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, the tax summary is displayed in My Accounts > Orders.




# Setting Up Multi-language Using Translations

You can set up multi-language support for your Dgitial 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 that is 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 picks up 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: tra
      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.

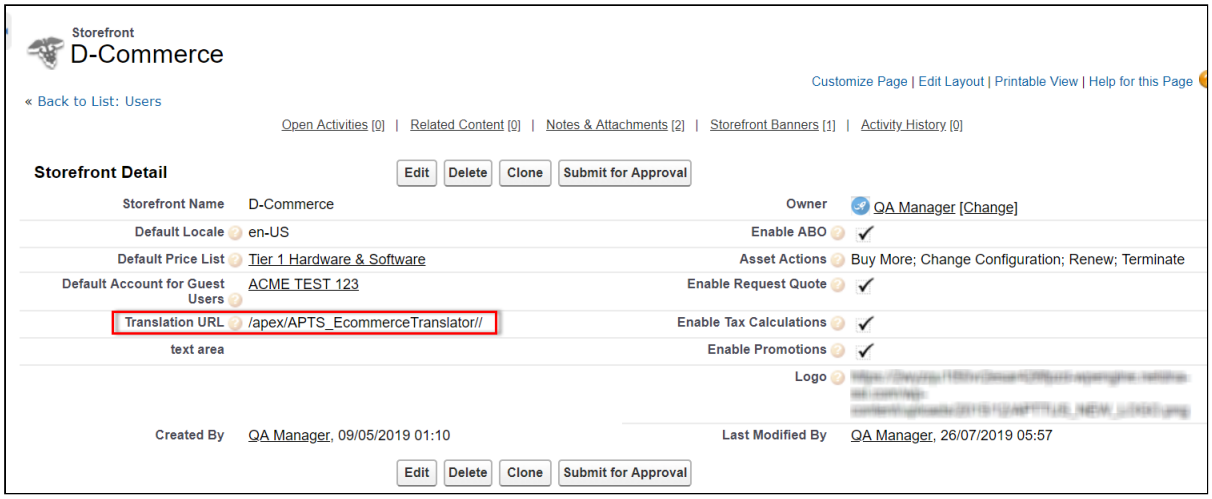
**Example Code Snippet for E-Commerce Translation**

```
<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>
```




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. It looks at the URL and read the file and get all the key values and based on that it gives the response and this is displayed on the template.

# Defining translations on Salesforce using Custom Labels

In order 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.

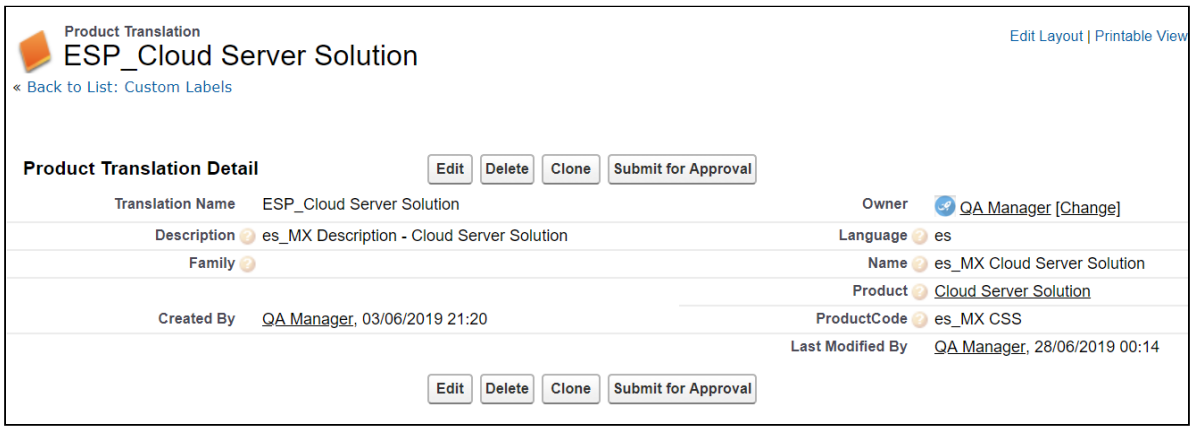
# Defining translations for custom objects

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

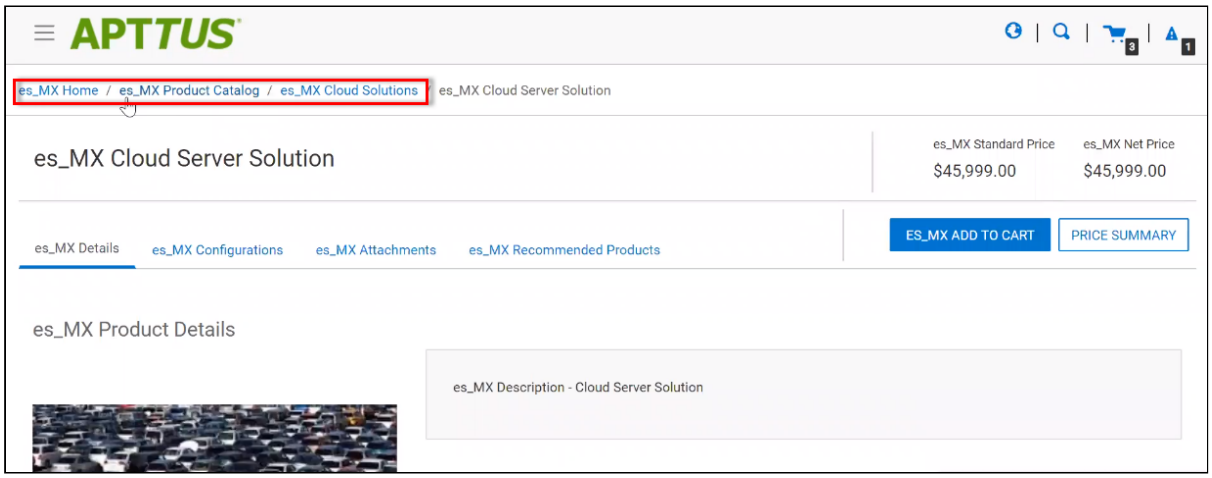
## To define translations for Products

1. Go to the Product Translations tab and click New.
   - OR -
2. 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.
3. Enter a Translation Name.
4. Enter translated text for the following fields:
   - Description
   - Language
   - Name
   - Product
   - ProductCode

   > ⚠ Currently, Family is not supported.




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




> For illustration purpose, the translated categories are prefixed with es_ in the above image. In 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 the 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) |

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

Help for this Page

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.

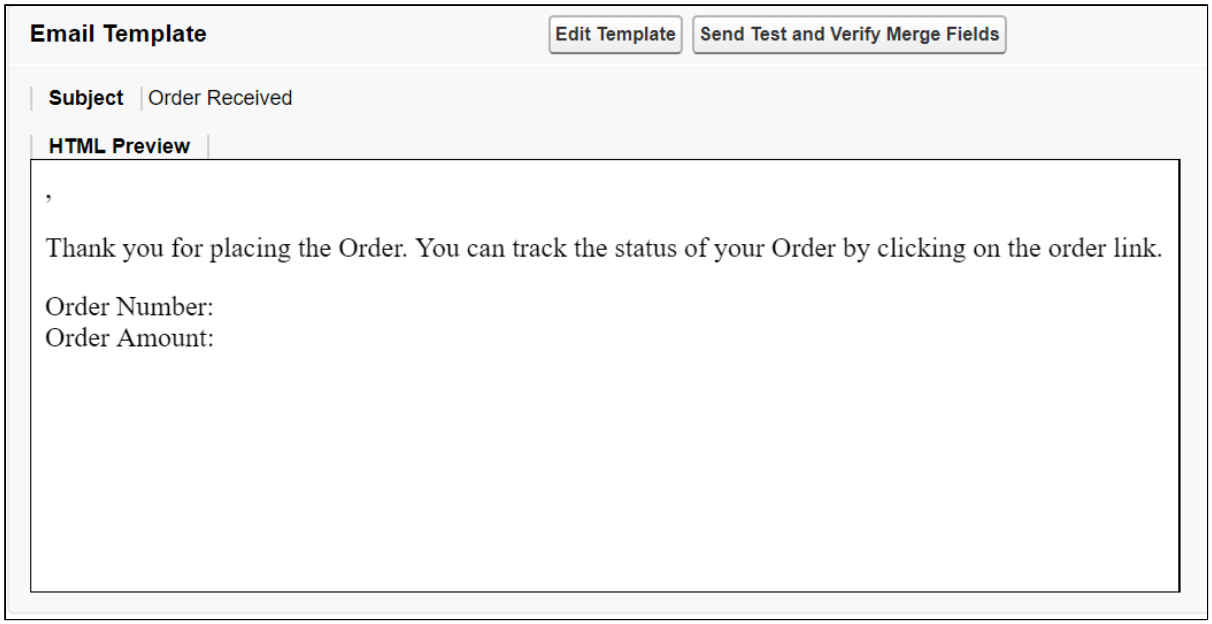
Folder ECommerceEmailTemplates Edit | Create New Folder

New Template

| 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. Th is email template is being used to notify users when order gets created from E-Commerce.

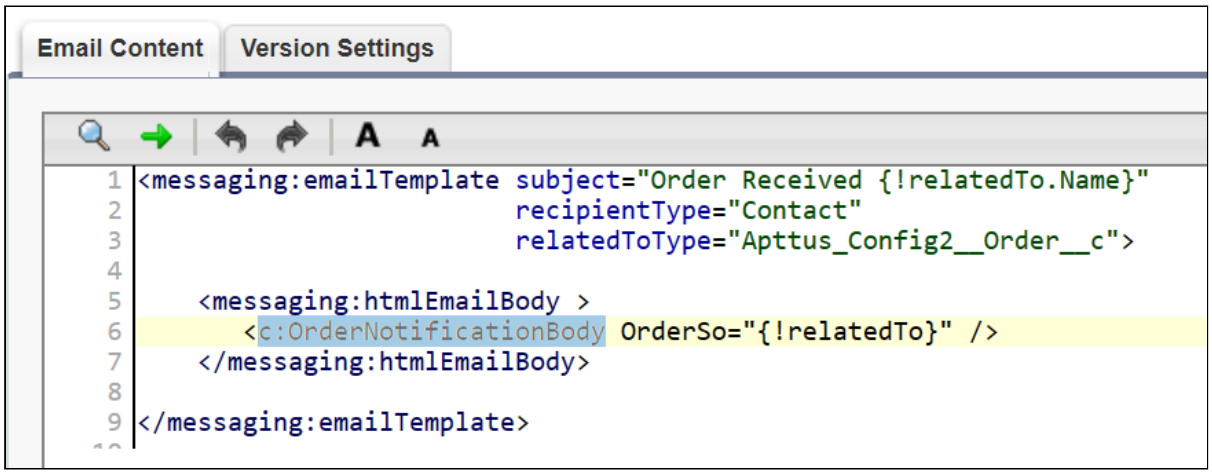
5. Under the Email Template related list, you can view the default email template.




6. Click Save.

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

1. Click Edit Template.




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. In order to change the body of the template, you can define your own component and replace the value here.

See example email below:

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

[redacted],

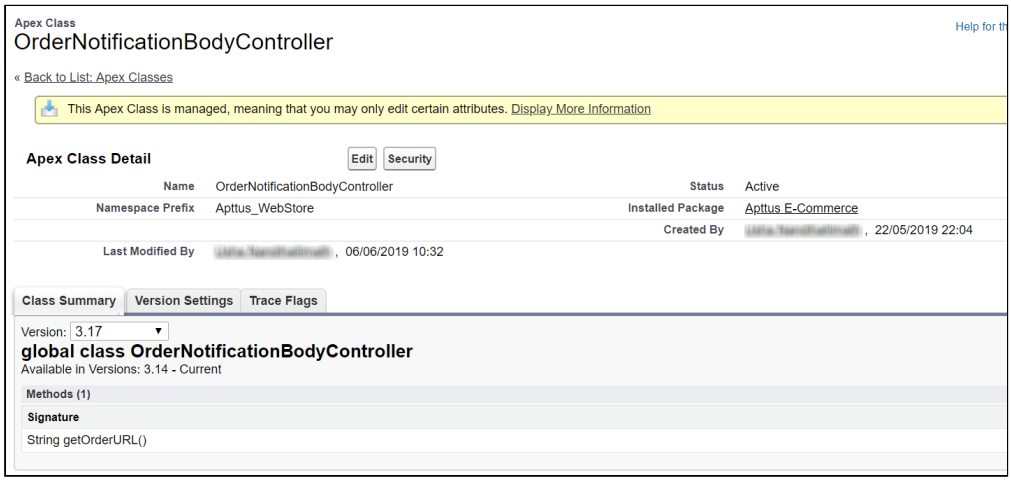
Thank you for placing the Order. You can track the status of your Order by clicking on the order link.

Order Number: O-00001842
Order Amount: $10.00

## Adding additional fields and changing the format of the field values in the email template

If you plan to use the out of the box template but want to change the format of the order number and amount. You can do so by changing the controller in the apex class.

1. Click Setup > App Setup > Develop and click Apex Classes.
2. Browse to and click to open OrderNotificationBodyController.




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

# 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.

# 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.

- A new 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.
- A new optional Communities setting, **Reassign new records created by guest users to the default owner**, is available that restricts a guest user from owning any sObject 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 new cart.
- A new 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.

## Resolution

The security alerts are *mandatory* at the end of February 2020. The security alerts are opt-in until that date.

If you allow guest users to do *any* of the following:

- Self-register
- Browse your storefront

- Add products to a cart
- Complete checkout

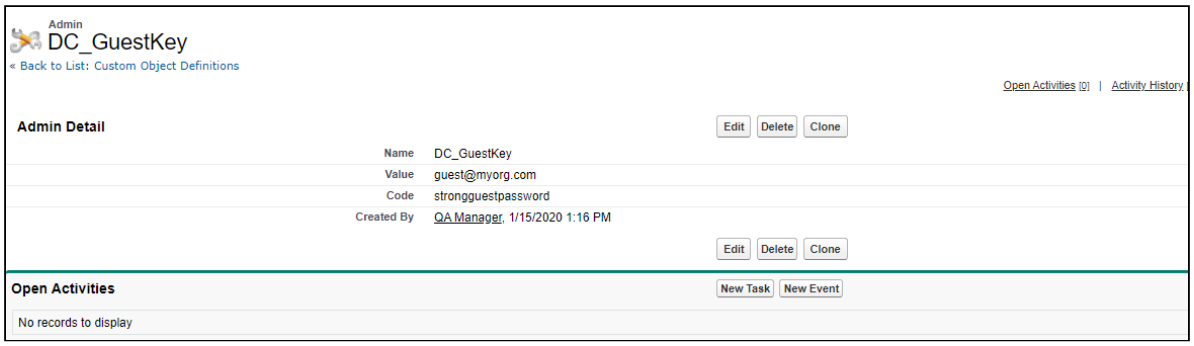
We recommend that you install the appropriate patch for your version of Apttus Digital Commerce and immediately complete the required workaround steps below to maintain guest user access on your storefront.

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 will not be able to perform the actions mentioned above.

## To maintain guest user access on your storefront

1. Create a remote site setting matching your community URL.
2. The application looks at a record with the Admin tab to act as the guest user account for the storefront.
3. Create a new user in the system with access to the community where the storefront is hosted.
4. Provide the Apttus Ecommerce permission set to the user you created in Step 3. This grants access to the various objects and fields necessary to perform the actions within the storefront.

5. Populate the credentials for the user in the Apttus__APTS_Admin__c object with the following attribute values:
   a. **Name**: APTS_DCGuestKey
   b. **Apttus__Value__c**: Guest Username (Formatted as an Email)
   c. **Apttus__Code__c:** Guest Password




6. Create sharing rules to allow the guest user to read the Admin key built above and the user record for the guest user:

**Product Information Sharing Rules** New Recalculate Product Information Sharing Rules Help

| Action | Criteria | Shared With | Access Level |
|---|---|---|---|
| Edit \| Del | Product Information: File NOT EQUAL TO null | E-Commerce Site Guest User | Read Only |

**User Sharing Rules** New Recalculate User Sharing Rules Help

| Action | Criteria | Shared With | User |
|---|---|---|---|
| Edit \| Del | User: Alias EQUALS guest | E-Commerce Site Guest User | Read Only |

# 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

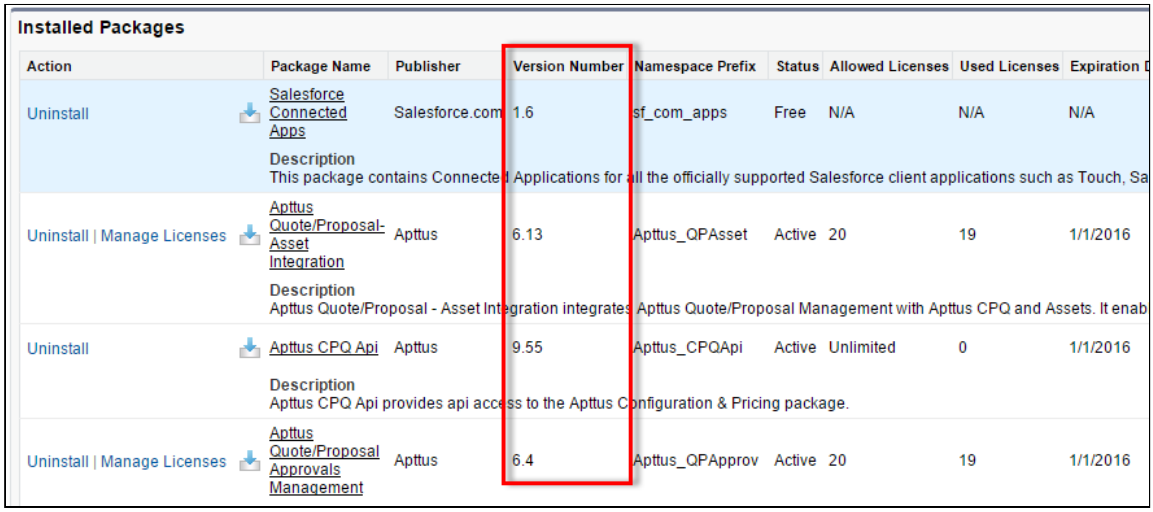
If you experience an issue with an Apttus product and need help, you can contact Apttus Support. Before you contact Apttus support, prepare a brief description of the problem you are experiencing. Additionally, to enable us to resolve your problem at the earliest, provide the following important information:

- 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.



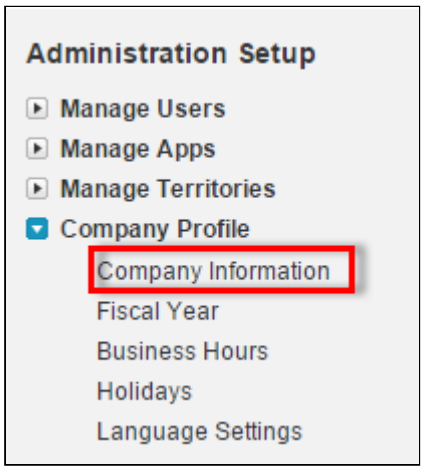
**Installed Packages**

| 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<br>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<br>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<br>Apttus CPQ Api provides api access to the Apttus Configuration & Pricing package. | | | | | | | |
| Uninstall \| Manage Licenses | Apttus Quote/Proposal Approvals Management | Apttus | 6.4 | Apttus_QPApprov | Active | 20 | 19 | 1/1/2016 |

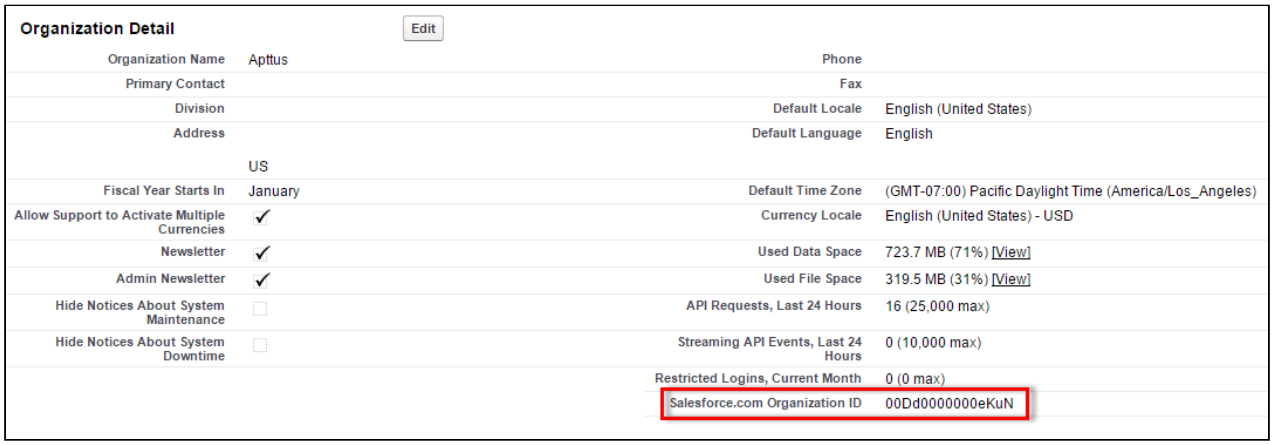
### What is your Salesforce.com Organization ID?

To determine the Salesforce.com organization ID:

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




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




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

To find the merge server end point:

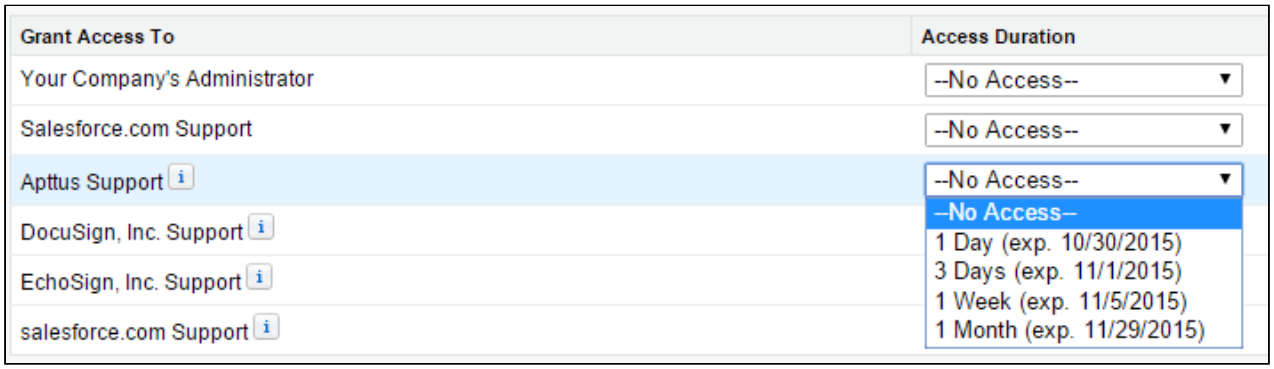
1. Go to **Setup > App Setup > Develop > Custom Settings**.
2. Click **Manage** for Comply System Properties.
3. Click **System Properties**.
4. The Merge Webservice Endpoint field displays the setting. The https://mergeserver.apttus.net:9876 portion of the setting is what will be helpful to customer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

### Grant Login Access of the affected user and an administrator.

To grant login access:

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


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




3. Click **Save**.

# Apttus Copyright Disclaimer



For additional resources and support, please visit https://community.apttus.com.

DOC ID: DCOMSFSUM19IDGREVB20200207