APTTUS®

Digital Commerce on Salesforce Winter 2018 Implementation and Deployment Guide

12/06/2018

Table of Contents

Overview 4
Key Capabilities 4
About This Guide 4
What's New 5
Apttus Digital Commerce Solution Architecture 6
Logging in to Apttus Digital E-Commerce 7
To log into the application 8
Installing Apttus E-Commerce Package 9
Setting Up Communities 10
To set up a community 10
Activating the Community 10
To activate a community 10
Creating Custom Field on Account for the Price List 11
Adding a Storefront Record 12
To add a storefront tab 12
To add a storefront record 12
Adding Storefront Promotional Banners 13
To create storefront banners 13
Cloning the Reference Templates 14
To clone a template from Apttus repository 14
Installing the Reference Template 15
To install the cloned template 15
Configuring Templates 15
Salesforce Credentials 16
Configuration Parameters 16
Setting Up Proxy for Local Development 18
Importing the app modules in the root module 18
Setting Up Subset of Categories 20

Turning Off Sentry for a Customer 20
Local Development Setup 21
Bootstrap Theme Changes 21
Server Deployment 22
To deploy your application on salesforce org 22
Customizing Your Application 23
Customizing HTML Content and Standard Components 23
To customize the HTML content 23
Adding Custom Fields on Object Models 24
To create a custom model 24
Adding Custom Attributes to a Product 26
Customizing Logic in the Services 27
Customizing the Template Page with Custom Field 27
To customize the template page 27
Setting Up Single and Multiple Store 29
Post Deployment Community Setup 30
Setting Up the Default Page 30
To set up a default page 30
Granting User Access to Community via Profiles 30
To enable users to access a community 30
Enabling Self Registration 30
To enable self registration 31
Setting Up Guest Users 31
To set up a guest user 31
Apttus E-Commerce Permission Set 31
Apttus Contact Support 33
Apttus Copyright Disclaimer 36

Overview

Apttus E-Commerce empowers enterprises to quickly evolve and scale global Omni-Channel selling strategies. Apttus E-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 E-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 E-Commerce sites.

Key Capabilities

- Accessible from any device, anywhere
- Multi-channel support for direct sales, partners, and end users
- Product configuration based on constraint and business rules
- Upsell and cross-sell recommendations
- Ability to differentiate products and pricing by channel and market segment
- Search products, services, features and pricing
- Shopping cart conveys shipping, tax, payment terms
- Integration with PCI-compliant systems

About This Guide

Apttus E-Commerce Guide is designed to provide administrators with information on setting up data to be consumed within Apttus E-Commerce. This guide covers the most common use cases for administration and assumes a level of familiarity with basic Salesforce.

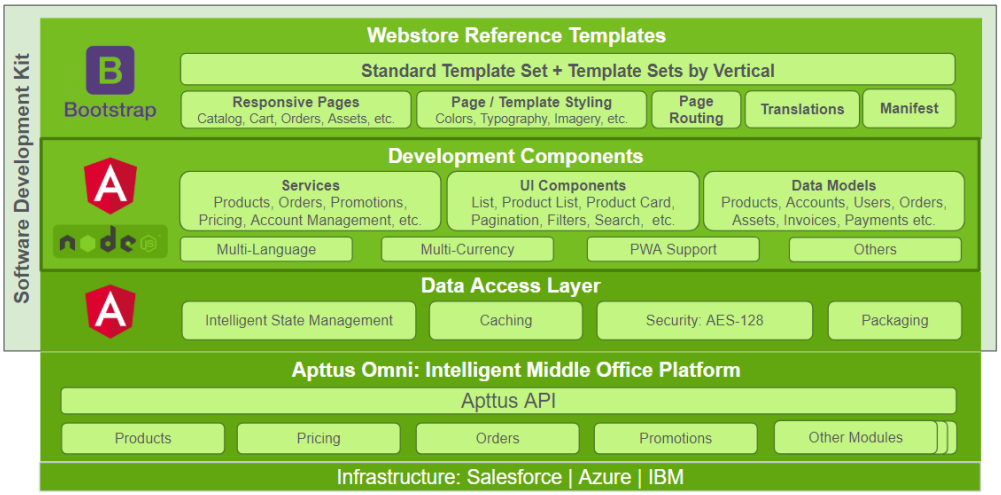
What's New

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

Release	Topic	Description
Winter 2018	Adding Custom Attributes to a Product	New topic. A new feature for this release.

Apttus Digital Commerce Solution Architecture

Apttus Digital Commerce Solution Architecture



Logging in to Apttus Digital E-Commerce

Log in to your Salesforce.com org.

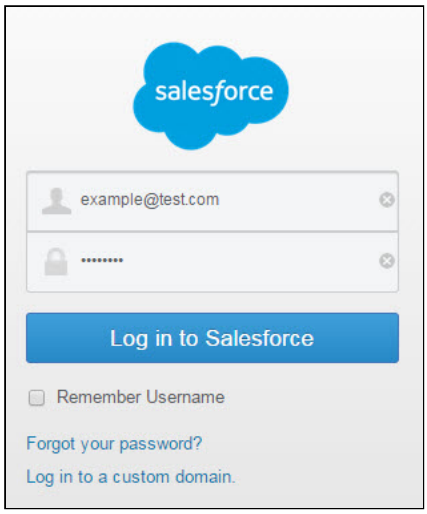
> **Note**
> Do not use the Back button on your browser.

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

- 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.

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		Y
7	Apttus CPQ API	CPQ	Y
8	Apttus Order Management		Y

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

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 name for your storefront.
4. For **Default Price List**, select a price list.
5. 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.

6. 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. Enter the following information:

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

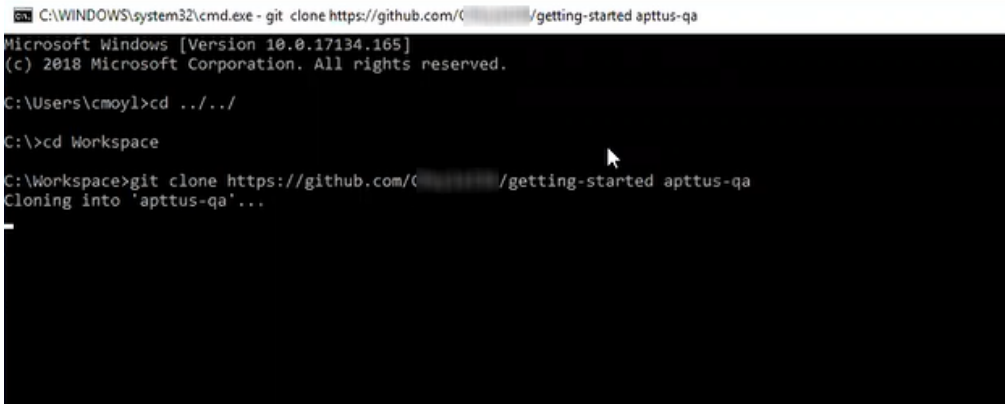
3. Click **Save**. The Storefront Banner details page appears.
4. Using **Note and Attachments**, attach a banner image to the Storefront Banner record.

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

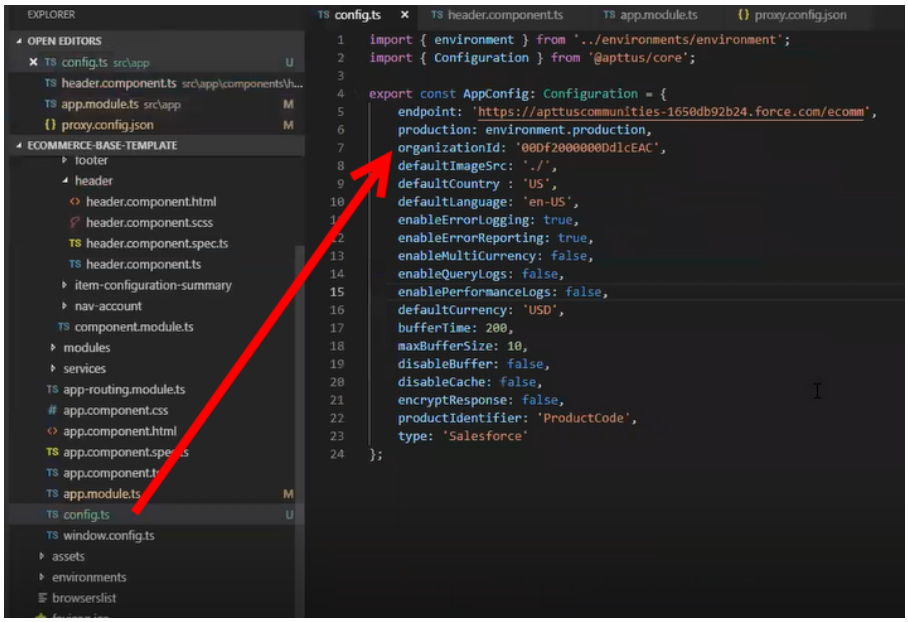
As part of the setup, sf.json file is created that contains the credentials for deploying the application. If you want to deploy your Storefront in a different org, you can update the credentials in this sf.json file.

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:

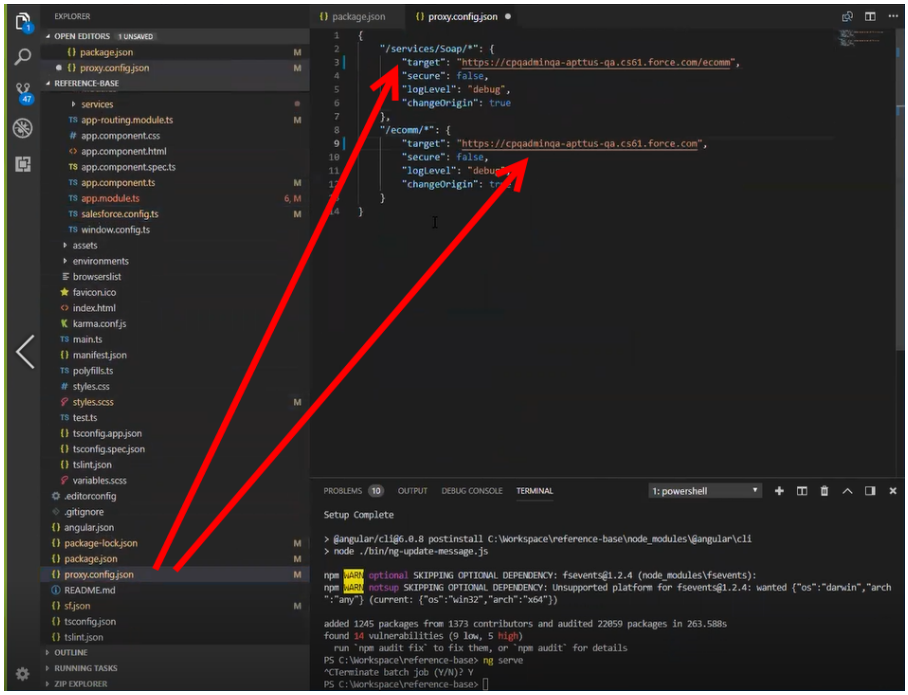
Parameter	Type	Description
endpoint	String	Type the community URL to point the application to.
production	True / False	Specify the environment where you want the application to run.
organizationId	String	Specify the org Id of the application. For details, see What is your Salesforce.com Organization 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.

Parameter	Type	Description
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.
encryptResponse	True / False	When set to true, responses from the server will be encrypted. ⚠ 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.



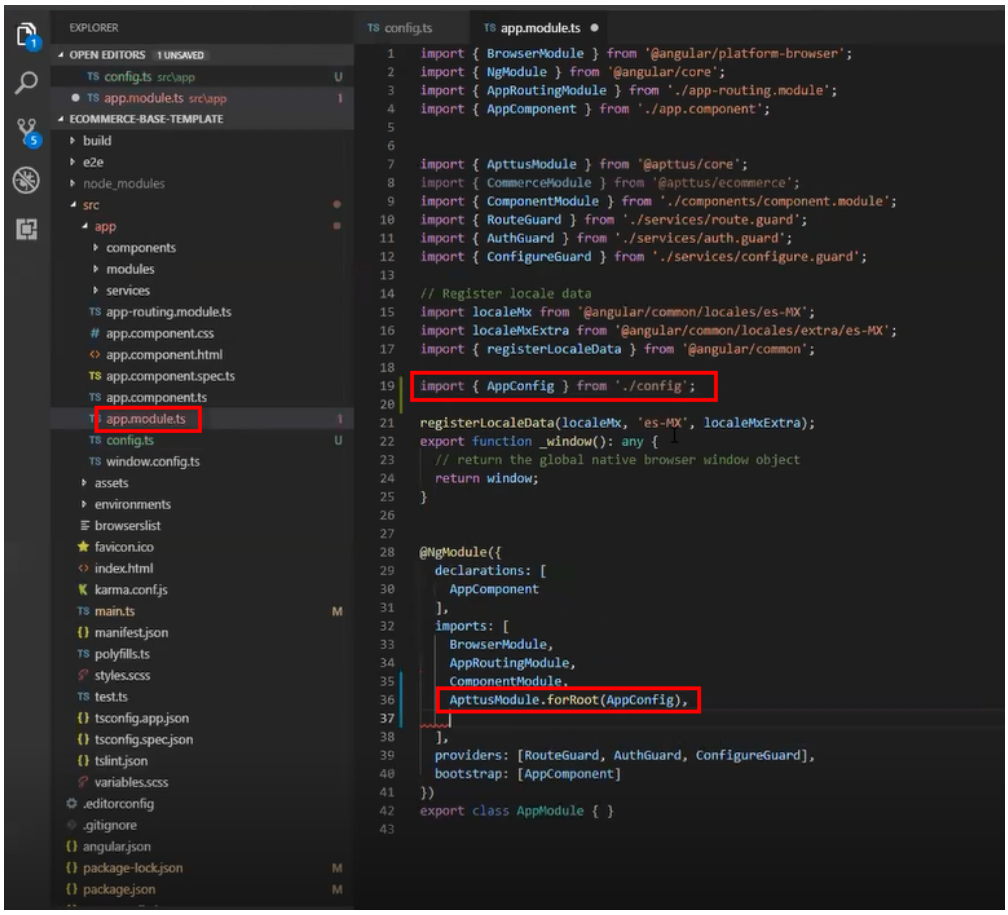
All the code for the templates is open sourced into the library we just cloned and installed. It consists of all the modules such as cart page, home page, account page and more. We can go into any template, for example, Home page layout where you can see some of the components of the Apttus underlying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

```
@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();
}
```

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.

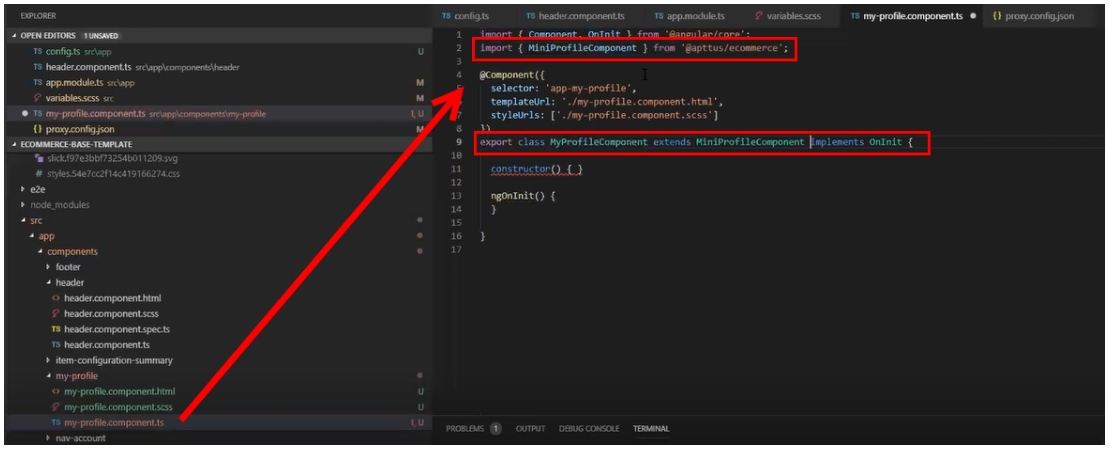
Customizing Your Application

Customizing HTML Content and Standard Components

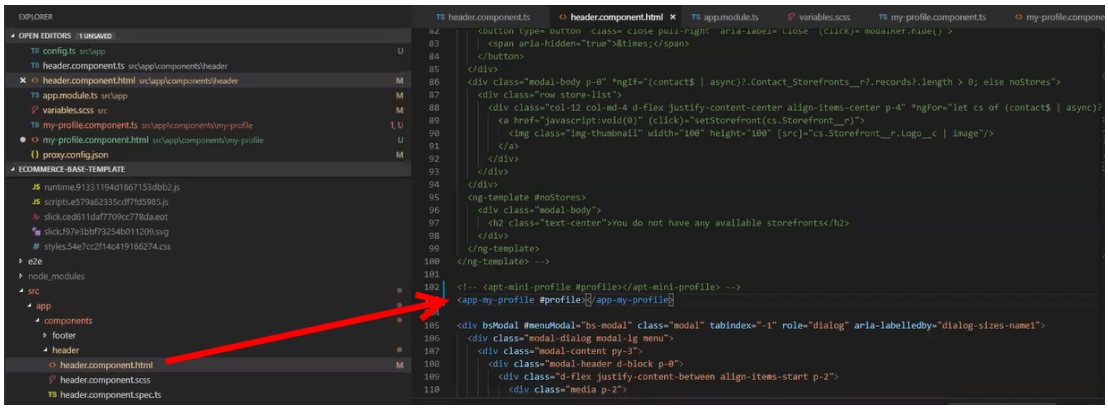
You can override the HTML for any component that comes with your angular application. You can do so by writing an extension component. Similarly, you can override the methods within it as well. For example, there is a registration method and you want to retain it but want to change the look and feel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

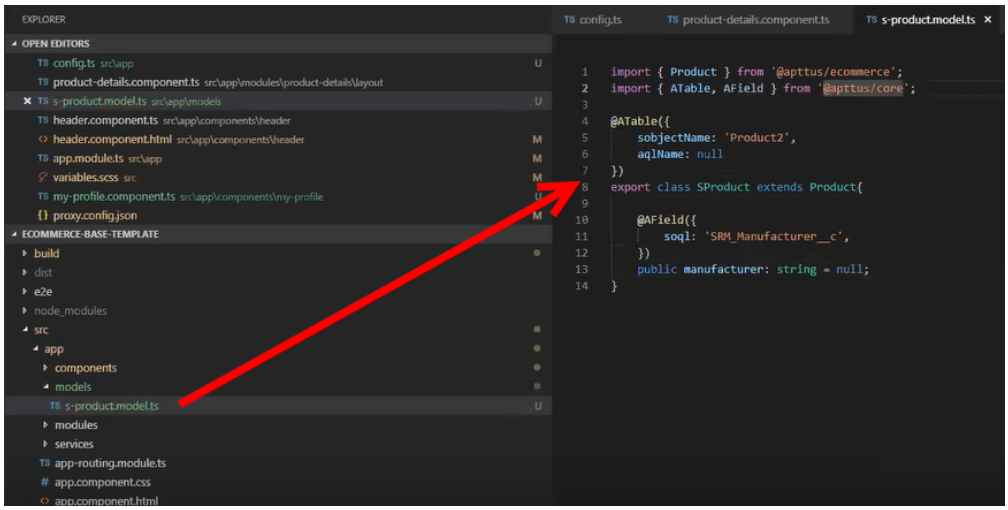
Adding Custom Fields on Object Models

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.

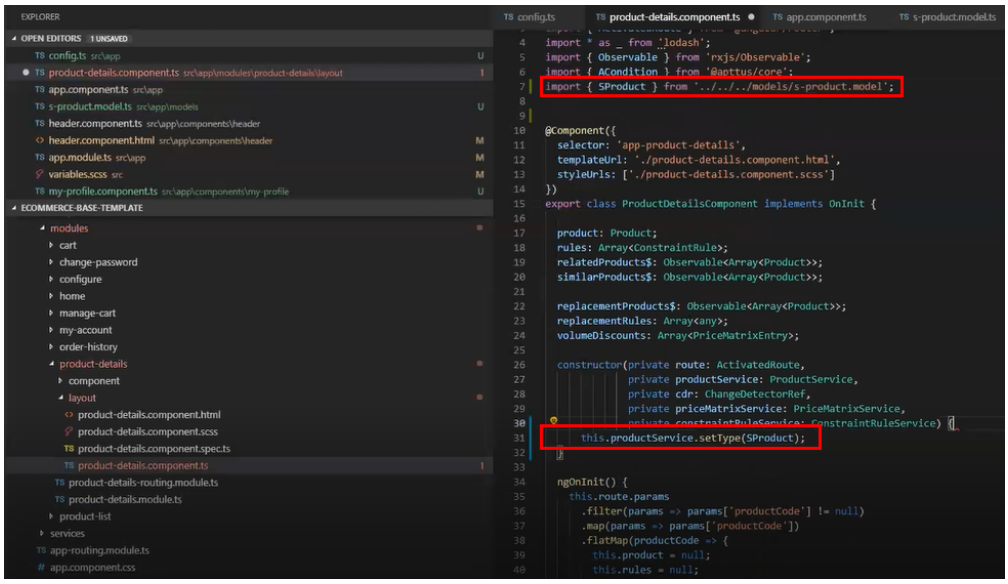
To create a custom model

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.

To add a custom attribute to a product

1. Create a typescript model with a custom class extending the ProductAttributeValue model from apttus/ecommerce library.
2. Add a custom product attribute with ATable and AField decorators to have a generic name for this attribute. Refer Adding Custom Fields on Object Models.

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

@ATable({
    sobjectName: 'Apttus_Config2__ProductAttributeValue__c',
    aqlName: 'clm_AgreementProductAttributeValue'
})
export class CustomProductAttributeValue extends ProductAttributeValue{
    @AField({
        soql: 'Rack_Units__c',
        aql: ''
    })
    RackUnits: string = null;
```

3. Change the attribute-accordion component to bind it with the template.

a. Import the newly created custom class to the attribute-accordion component.

```
import { Component, OnChanges, ChangeDetectionStrategy, Input, Output, EventEmitter } from
import { ProductAttribute, Product, ProductAttributeMap, ProductAttributeValue, InputFieldC
import { MetadataService } from '@apttus/core';
import { CustomProductAttributeValue } from '../../../../custom-product-attribute.model';
```

b. Change the return type of **setAttributeDefaults** method and add an instance of **CustomProductAttributeValue** class to **attributeValue**.

```
setAttributeDefaults(productAttributeList: Array<ProductAttribute>): CustomProductAttributeValue {
  const attributeValue: CustomProductAttributeValue = new CustomProductAttributeValue();
  productAttributeList.forEach(attr =>{
    attr.Field = this.metadataService.getKeyName(CustomProductAttributeValue,attr['_describe'].name);
    attributeValue[attr['_describe'].name] = InputFieldComponent.getDefaultValue(attr['_describe'])
  });
  return attributeValue;
}
```

c. You can now use **getKeyName** method of **metadata service** to get the generic name 'RackUnits' of the custom attribute field by passing in the field name and a reference to the custom class.

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:

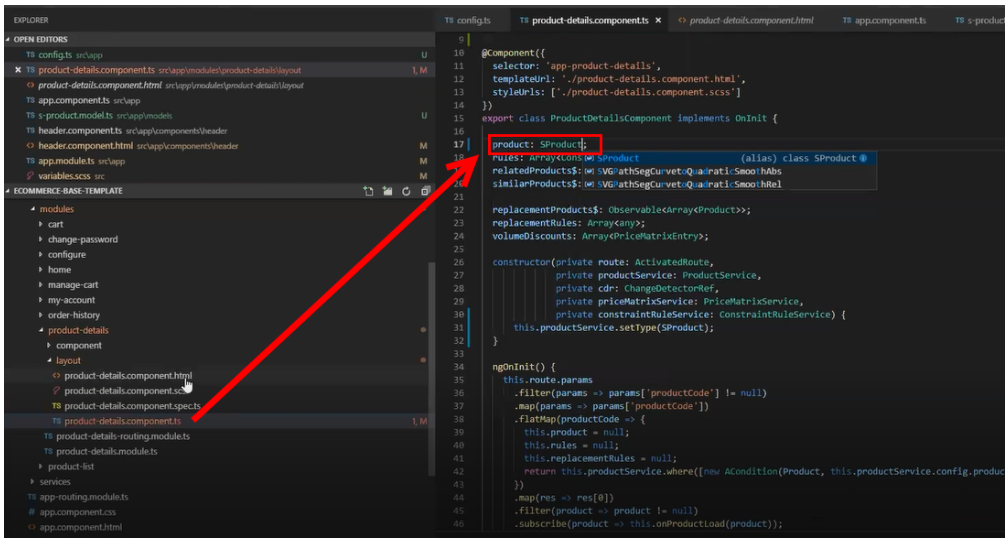
- Change the modal that goes with some business logic and service
- Change the business logic or service for a particular modal
- Create completely new modals and new services

Customizing the Template Page with Custom Field

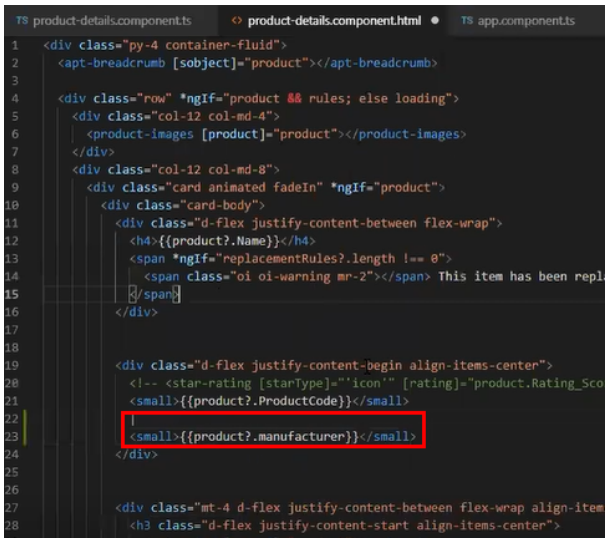
You can use the custom field you just created in your template.

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.



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.

Apttus Contact Support

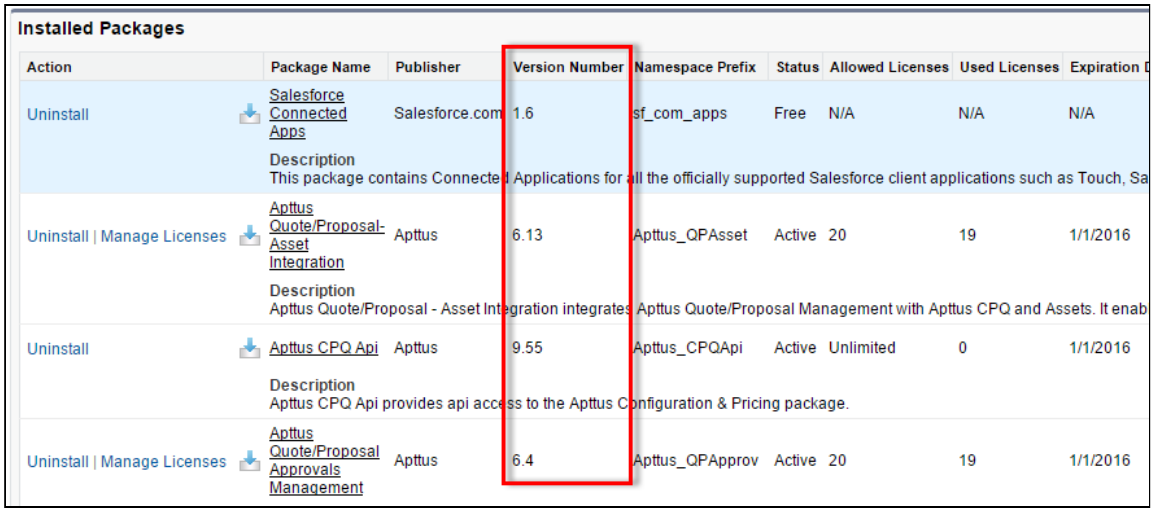
If you experience an issue with an Apttus product and need help, you can contact Apttus Support. Before you contact Apttus support, prepare a brief description of the problem you are experiencing. Additionally, to enable us to resolve your problem at the earliest, provide the following important information:

- 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.



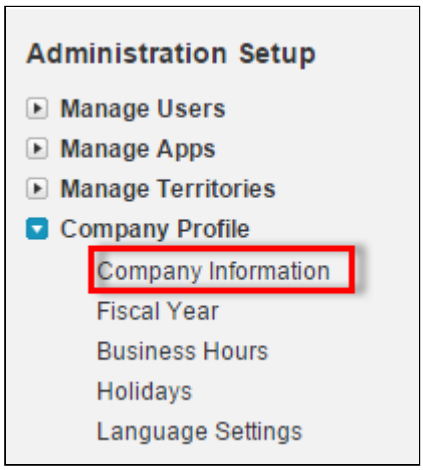
Installed Packages

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

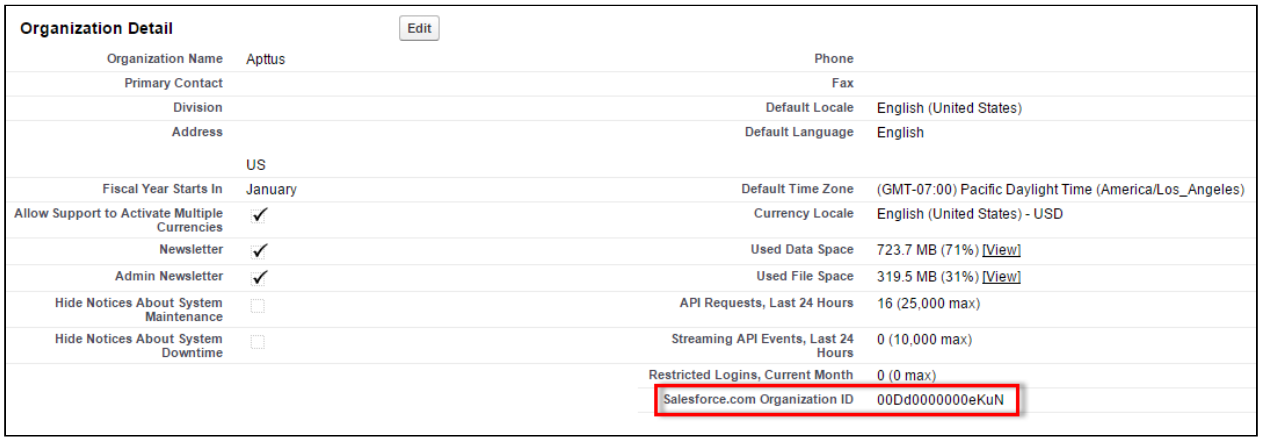
What is your Salesforce.com Organization ID?

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.



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

To find the merge server end point:

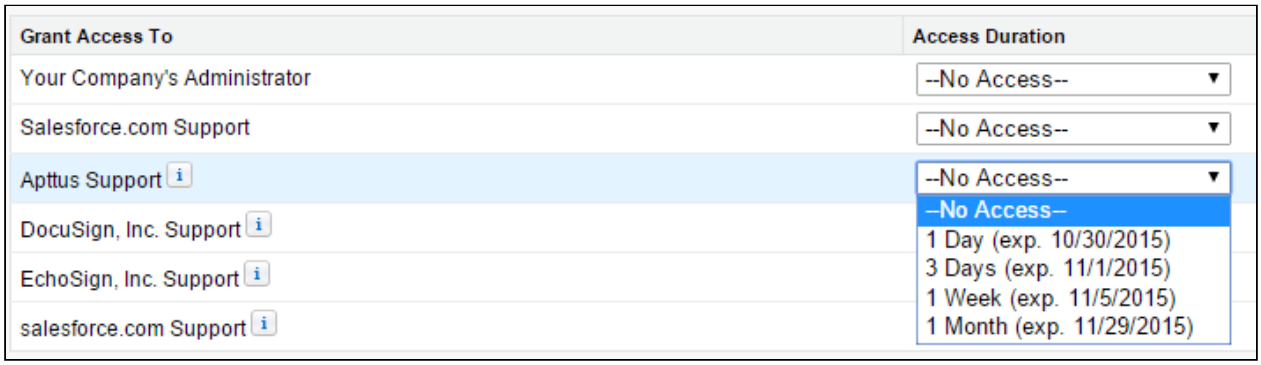
1. Go to **Setup > App Setup > Develop > Custom Settings**.
2. Click **Manage** for Comply System Properties.
3. Click **System Properties**.
4. The Merge Webservice Endpoint field displays the setting. The https://mergeserver.apttus.net:9876 portion of the setting is what will be helpful to customer 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: DCOMSFWIN18IDG20181206

