

Subscribe Pro

Version 17.1.0



Table of Contents

1.	Summary	1-3
2.	Component Overview	2-4
2.1	Functional Overview	2-4
2.2	Use Cases	2-4
2.3	Limitations, Constraints	2-4
2.4	Compatibility	2-4
2.5	Privacy, Payment	2-4
3.	Implementation Guide	3-5
3.1	Setup	3-5
3.2	Configuration	3-5
3.3	Custom Code	3-5
3.4	External Interfaces	3-5
3.5	Testing	3-5
4.	Operations, Maintenance	4-6
4.1	Data Storage	4-6
4.2	Availability	4-6
4.3	Support	4-6
5.	User Guide	5-7
5.1	Roles, Responsibilities	5-7
5.2	Business Manager	5-7
5.3	Storefront Functionality	5-7
6.	Known Issues	6-8
7.	Release History	7-9

1. Summary

This document provides technical instructions for installing the Commerce Cloud Integration LINK integration that lets you integrate Subscribe Pro product subscriptions with a Commerce Cloud storefront.

This cartridge integrates Salesforce Commerce Cloud with the Subscribe Pro Subscription Commerce Platform.

This integration requires deployment of a new cartridge and modification of Storefront code.

A subscription to the Subscribe Pro solution is required to use this cartridge.
Sandbox (testing) environments are available.

Provided “int_subscribe_pro” cartridge is implemented using controllers architecture.

This integration is based on the Site Genesis demo store provided by Commerce Cloud.

2. Component Overview

2.1 Functional Overview

Subscribe Pro enables brands and retailers to add subscription commerce, auto-ship, continuity, clubs, boxes and recurring billing features to their existing eCommerce site, fully integrated with their existing eCommerce technology. Subscribe Pro supports recurring payments via credit card, ApplePay and eCheck, with pre-built integrations for all common payment gateways. Contact us today to find out how our solution can help you implement a product subscription program.

2.2 Use Cases

Scenario: A user visits a site that has subscription products available to purchase and would like to purchase those Products one time in order to try the product out before signing up for a recurring subscription.

Result: User will be presented with various options to subscribe to the subscription product with one of the options being a one time purchase. The user should be able to add this product to cart and checkout as if the product is a normal product in the catalog. Messaging should be present throughout checkout to inform the user that this is a one time purchase.

Scenario: A user visits a site that has subscription products available to purchase and would like to purchase a subscription product on a recurring interval automatically.

Result: User will be presented with various options to subscribe to the subscription product with one of the options being a subscription. The user should be able to add this product to cart and checkout as if the product is a normal product in the catalog. Messaging should be present throughout checkout to inform the user that this is a subscription purchase and display the interval they have chosen for repeat orders.

Scenario: A user has added some subscription products to their Basket and would like to adjust some of the values.

Result: User will be presented with options to change their subscription settings on the Cart Page. The user will also be able to click the “edit” link, on the subscription product, and change the subscription options in the quick view dialog.

Scenario: User has purchased a subscription product. The next time the ProcessSubproSubscriptions is run the necessary details about the customer and the products purchased will be transmitted to Subscribe Pro and the order will be marked as having been processed.

2.3 Limitations, Constraints

- Merchants must be enrolled at Subscribe Pro
- Subscription products must be configured at Subscribe Pro and Sales Force Commerce Cloud
- Apply Pay or a Credit Card will be required for subscription products
- Merchant must use Subscribe Pro as their Apple Pay Provider, if they want to use Apple Pay for Subscription Products
- Customers must be logged into the storefront in order to purchase subscription products
- Customer must save their Credit Card to their account in order to purchase subscription products

2.4 Compatibility

Compatible Demandware API version 16.2

Compatible Site Genesis version 17.4

2.5 Privacy, Payment

Customer profile data and payment information will be transferred and stored at Subscribe Pro.

This information includes:

- Name
- Email
- Address Details
- Saved Payment Instrument ID

Subscribe Pro will be handling Apple Pay Payment transactions. Subscribe Pro will store the Customers saved Payment Instrument ID, in order to process recurring orders.

3. Implementation Guide

3.1 Setup

Subscribe Pro LINK integration uses the following cartridges:

- *int_subscribe_pro* – this cartridge contains all controllers, templates and scripts to create and display Subscribe Pro subscriptions. Also LINK-subscribe-pro contains the Business Manager XML imports such as the System Object Types customizations, services configurations and configuration of job for processing orders with pending subscription request.
- *app_storefront_controllers*,
- *app_storefront_core* – standard storefront cartridges with Subscribe Pro integrated. They can be used as an example for the integration and testing the cartridge.

Complete the following steps to install the cartridge.

Initial Install

See the Commerce Cloud documentation for adding a cartridge to your storefront:

https://documentation.demandware.com/DOC1/index.jsp?topic=%2Fcom.demandware.dochelp%2FSiteDevelopment%2FAddinganexistingcartridgetoyourstorefront.html&cp=0_4_0_0_3

Note. You must also register the cartridge with the Business Manager site in order to run the Subscribe Pro jobs. Navigate to *Administration > Sites > Manage Sites > Business Manager > Settings tab*.

Update Site's cartridge path

1. Navigate to *Administration > Sites > Manage Sites > [SiteName] > Settings tab*.
2. Add “:int_subscribe_pro” to the effective cartridge path.
3. Click “Apply”.

Install Services

1. Navigate to *Administration > Operations > Import & Export > Services*
 - a. Import the profile and credentials found in the following XML
 - i. *LINK-subscribe-pro/Metadata/services/services-shared.xml*
 - b. Import the service definitions found in the following XML
 - i. *LINK-subscribe-pro/Metadata/services/services.xml*

Import Meta Data

1. Navigate to *Administration > Site Development > Import & Export > Meta data.*
2. Import the System Object custom attributes found in the following XML
 - a. *LINK-subscribe-pro/Metadata/meta/system-objecttype-extensions.xml*

After import, following system definition attributes should appear in the system:

System object definitions

CustomerAddress			
ID	Name	Type	Description
subproAddressID	Subscribe Pro Address ID	String	This attribute will contain the address id that is returned from Subscription Pro upon creating this address in the Subscribe Pro system.
CustomerPaymentInstrument			
ID	Name	Type	Description
subproPaymentProfileID	Subscribe Pro Payment Profile ID	String	This attribute will contain the payment profile id that is returned from Subscription Pro upon creating payment profile record in the Subscribe Pro system.
subproCCPrefix	Credit Card Prefix	String	This attribute will contain the first 6 digits of the customers Credit Card, to be sent to Subscribe Pro.
Order / Basket			
Basket duplicates the order attributes so that they will automatically be passed from one to another.			
ID	Name	Type	Description
subproContainsSubscriptions	Contains Subscriptions	Boolean	This attribute will be used to quickly assess if an order has subscriptions contained within it
subproSubscriptionsToBeProcessed	Subscriptions to be processed	Boolean	This attribute will note whether or not this order has subscriptions that need to be processed. This value will be set to true if any products in the basket have subscriptions and will be set to false after those subscriptions have been processed
OrderAddress			

ID	Name	Type	Description
subproAddressID	Subscribe Pro Address ID	String	This attribute will contain the address id that is returned from Subscription Pro upon creating this address in the Subscribe Pro system.
customerAddressID	SFCC Customer Address ID	String	This attribute will contain the SFCC customer address id. This information may be useful when a customer is updating subscription options after an order has been placed.
Product			
ID	Name	Type	Description
subproSubscriptionEnabled	Subscription Enabled	Boolean	This will decide whether or not the PDP should attempt to show subscription information
ProductLineItem			
ID	Name	Type	Description
subproSubscriptionOptionMode	Subscription Option Mode	String	This attribute will contain a value for whether or not if this product is subscription or a one time purchase. Possible values are: onetime_purchase subscription
subproSubscriptionSelectedOptionMode	Subscription Selected Option Mode	String	This attribute will contain subscription mode which was chosen on this ProductLineItem
subproSubscriptionInterval	Subscription Refill Interval	String	This attribute will contain the user selected time interval to refresh a subscription product
subproSubscriptionAvailableIntervals	Subscription Available Intervals	String	This attribute will contain a string with names of available subscription refill intervals, separated by comma
subproSubscriptionCreated	Subscription Created	Boolean	
subproSubscriptionDateCreated	Subscription Date Created	Date	
subproSubscriptionID	Subscription ID	String	This will be populated when the order processing occurs and the actual subscription is created.

subproSubscriptionDiscount	Subscription Discount	Double	This attribute will contain subscription discount offered on this product, as either a percentage or a fixed price, depending on the value of the subproSubscriptionIsDiscountPercentage property
subproSubscriptionIsDiscountPercentage	Subscription Is Discount Percentage	Boolean	This attribute will contain true if the discount on the product specified as a percentage or false if it is a fixed price amount.

Profile

ID	Name	Type	Description
subproCustomerID	Subscribe Pro Customer ID	String	This attribute will contain the customer id that is returned from Subscription Pro upon creating this customers records in the Subscribe Pro system.

Site preferences

Please configure the following site preferences:

ID	Name	Type	Description
subproApplePayLoginMsg	Subpro Apple Pay Login Message	String	Message informing customer that he/she needs to be logged in as a registered customer to use Apple Pay with SubPro Subscription products
subproCheckoutLoginMsg	Subpro Checkout Login Message	String	This attribute will contain a message that informs customer that he/she needs to be logged in as a registered customer to proceed checkout with SubPro Subscription products
subproCreditCardRequirementMsg	Subpro Credit Card Requirement Message	String	This attribute will contain an error message noting that a credit card is required for subscriptions
subproOrderProcessingErrorMail	Subscribe Pro Order Processing Error Mail	String	An email address that order subscriptions processing failures will be sent to
subproWidgetScriptUrl	Subscribe Pro Widget Script Url	String	URL to Subscribe Pro Widget Script
subproApiBaseUrl	Subscribe Pro API Base Url	String	Subscribe Pro API Base URL
subproAPICredSuffix	Suffix to append to the Subscribe Pro Web Service Credentials	String	Suffix to append to the Subscribe Pro Web Service Credentials

Payment Card

ID	Name	Type	Description
subproCardType	SubPro Card Type	String	Credit Card Type that matches the Credit Card Types at Subscribe Pro

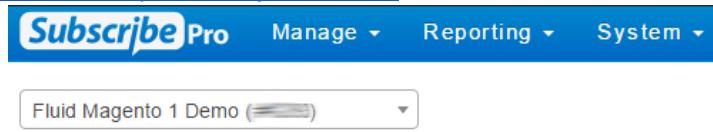
Import ProcessSubproSubscriptions job

1. Navigate to *Administration > Operations > Import & Export > Job Schedules*.
 2. Import the System Object custom attributes found in the following XML
 - a. *LINK-subscribe-pro/Metadata/jobs/process-orders-job.xml*
-

3.2 Configuration

Update Web Service Credentials

1. Get your Client ID and Client Secret in Subscribe Pro BM:
<https://platform.subscribepro.com/system/client>



Edit API Client #1999

A screenshot of the "Edit API Client #1999" form. It has a "Client Name" field containing "Magento Store". Below it are two password input fields: "Client ID (Public)" and "Client Secret", both of which have red boxes around them and arrows pointing to them from the text above. The "Client ID (Public)" field contains a long string of characters, and the "Client Secret" field also contains a long string of characters.

2. Navigate to *Administration > Operations > Services > Service Credentials > subpro.http.cred - Details*
3. Use Client ID as User and Client Secret as Password:

[Administration > Operations > Services > Service Credentials > subpro.http.cred - Details](#)

subpro.http.cred

Fields with a red asterisk (*) are mandatory. Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

These credentials are used by 12 services.

A screenshot of the "subpro.http.cred" service credential configuration form. It includes fields for "Name:" (set to "subpro.http.cred"), "URL:" (set to "https://api.subscribepro.com/services/v2/{ENDPOINT}?{PAR}"), "User:" (containing "XXXXXXX" with a red box and arrow), and "Password:" (containing "*****" with a red box and arrow). At the bottom are "Apply" and "Reset" buttons.

4. Navigate to *Administration > Operations > Services > Service Credentials > subpro.http.cred.oauth - Details*
5. Use Client ID as User and Client Secret as Password (the same as above).

Setup Schedule Job to process orders with pending subscriptions

The job will process all orders that have any subscriptions that need to be processed.

Configure job parameters:

1. Navigate to *Administration > Operations > Job Schedules > ProcessSubproSubscriptions* job.
2. On tab *Schedule and History* set run interval (default is every 15 minutes).
3. Open *Step Configurator* tab, choose *ProcessSubproSubscriptions-Process* workflow.
4. Set (**in hours**) custom parameter *ordersProcessInterval* to define what the start time for the search placed orders should be.
5. Click “Assign” button.

Configure Subscription Products

To configure products which will have a subscription you'll need to make changes to SFCC Business Manager as well as to Subscribe Pro Business Manager.

There are different type of products that can be configured:

- Standard Product
- Variation Product
- Variation Masters
- Product Bundles
- Product Sets
- Option Product

Below described action you need to perform to configure each type of product.

Standard Product

Standard Product is a product that is sold and displayed alone and does not have variations, such as different sizes or colors. If you include a standard product in a product set or product bundle, it is considered a set product or bundled product, and not a standard product.

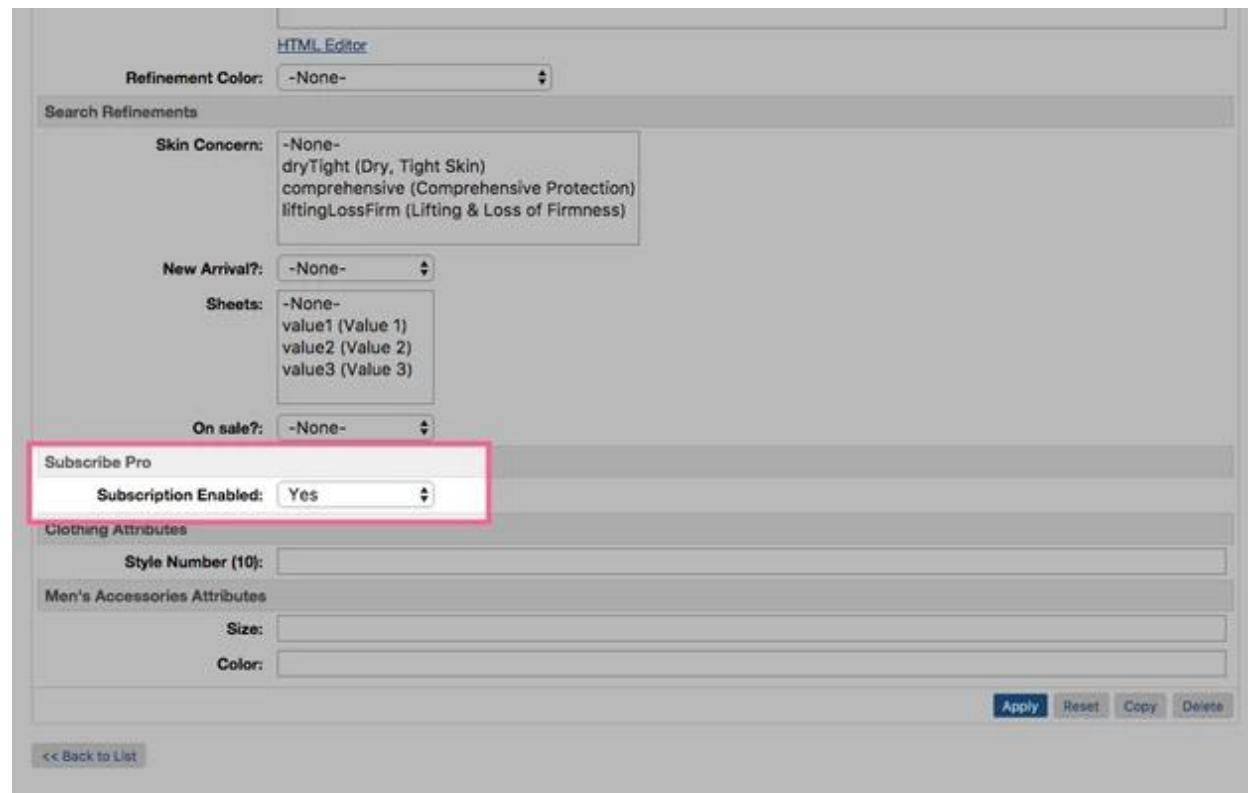
Configuring Product on the Commerce Cloud Platform

The PDP page will take the Commerce Cloud Product ID and make an API request to the Subscribe Pro Products API Endpoint using the Commerce Cloud Product ID as the "sku" parameter.
The following image shows the Product ID that should be configured as a SKU in the Subscribe Pro Platform.

The screenshot shows the 'General' tab of a product configuration page. The product ID '12345' is highlighted with a pink rectangle. A yellow warning message at the top states: 'You have not locked this product for editing. Click [Lock](#) if you need to edit the product'. Below the message, there's a note: 'Click Lock at the top of the page to edit this product. Click Apply to save changes. Click Reset to revert to the edit data in other languages; use the Select Language drop-down to define in which language you are viewing'. A 'Select Language' dropdown is set to 'Default'. The 'ID:' field contains '12345'. Other fields shown include 'Catalog:' (apparel-catalog), 'Tax Class:' (Standard), 'Searchable:' (Default, Yes), and 'Searchable If Unavailable:' (Default, -None-, All Site Values).

This API call will only be attempted if the Commerce Cloud Product has the: **Subscription Enabled** custom attribute set to: **true**.

The following image shows this setting in the following Business Manager Module:
Merchant Tools > Products and Catalogs > Products



The screenshot shows the 'Products and Catalogs > Products' module in the Business Manager. At the top, there's an 'HTML Editor' button and a 'Refinement Color' dropdown set to '-None-'. Below that is a 'Search Refinements' section with three dropdown menus:

- Skin Concern:** Options include '-None-', 'dryTight (Dry, Tight Skin)', 'comprehensive (Comprehensive Protection)', and 'liftingLossFirm (Lifting & Loss of Firmness)'.
- New Arrival?:** Options include '-None-'.
- Sheets:** Options include '-None-', 'value1 (Value 1)', 'value2 (Value 2)', and 'value3 (Value 3)'.

Below these is an 'On sale?' dropdown set to '-None-'. Underneath these sections is a 'Subscribe Pro' section, which is highlighted with a red box. It contains a 'Subscription Enabled' dropdown set to 'Yes'. Further down are sections for 'Clothing Attributes' (Style Number (10) field), 'Men's Accessories Attributes' (Size and Color fields), and a row of buttons for 'Apply', 'Reset', 'Copy', and 'Delete'.

Configuring Product on the Subscribe Pro Platform

In addition to configuring the product on Commerce Cloud, the product will also need to be configured on the Subscribe Pro Platform.

<https://platform.subscribeopro.com/login>

Subscription products can be managed on the platform at: *Manage > Subscription Products*

Within this module a new subscription product should be created that has a SKU that matches the Commerce Cloud Product ID.

The following is an example of a configured product that will match the product provided above for Commerce Cloud:

Subscribe Pro Manage Reporting System Welcome, James Kimmel

Fluid Magento 1 Demo (#2893)

Edit Product #587218 - "Trial Test Product"

[Product Settings](#) [Product Trial Settings](#)

Product Information

SKU	Product Name
12345	Trial Test Product

Show Product On Subscribe Pro Management UI

Subscription Widget - Subscription Option

Subscription Option Mode: Subscription and One-Time Purchase

Default Subscription Option: One-Time Purchase

Product Options Handling

Product Options Mode: Pass-through Options in Reorders

Product Shipping Options

Shipping Mode: Product Requires Shipping

Product Pricing and Subscription Discount

Full Price: \$ 100.00

Discount By Percentage (Otherwise use Fixed Discount)

Discount Percentage: 20.00	Fixed Discount Amount: \$ 0.00
----------------------------	--------------------------------

Subscription Quantities

Min. Qty.: 1	Max. Qty.: 9
--------------	--------------

Product Intervals

Select which intervals shall be available for this product.
Drag the selected intervals into the order you wish them to appear in the subscription widget drop-down on your Magento site.

Unselected	Selected
IT 30 Days	IT Every 2 Months
IT 45 Days	IT Monthly
IT Daily	IT Weekly
IT Every 2 Weeks	

Default Interval:

Select the interval which should be selected by default for this product.

[Back](#) [Save](#)

Variation Product

Variation Product is a product that is a specific variation. For example, if a Vegas brand t-shirt is represented by a variation master, then a product variation is a size 10, blue Vegas brand t-shirt. Variation products associated with the same variation master share most of the attributes defined for the product master, but have their own product IDs and images.

Variation Products should be configured the same way as a "Standard Product". See instructions above.

Variation Masters

Variation Masters is a product that represents all the variations for a particular product. For example, a specific style of t-shirt.

A variation master cannot be purchased directly, only product variations with specific variation information can be purchased. For example, if a master has color and size variation attributes, you must specify the color and size of a specific variant to purchase it.

If you have more than three variation attributes, you might want to consider using options.

To enable subscriptions for variation master products:

1. A Subscribe Pro Subscription Product will need to be created for each of the Variants contained within the Master Variant.
2. The master variant will need to have the "Subscriptions Enabled" custom attribute set to "yes".
 - a. Any attributes set on the Master Product will flow down to all of the variant products, unless the variant product has overridden that particular attribute.

Example:

Master Product (in Commerce Cloud)

ID	Name
25752235	Checked Silk Tie

Variant Product(s) (in Commerce Cloud)

Parent ID	ID	Options
25752235	682875090845	Color: Cobalt
25752235	682875719029	Color: Navy
25752235	682875540326	Color: Yellow

Subscribe Pro - Subscription Products (in Subscribe Pro Platform)

ID	SKU	Name
587218	682875090845	Checked Silk Tie - Cobalt
587219	682875719029	Checked Silk Tie - Navy

587220	682875540326	Checked Silk Tie - Yellow
--------	--------------	---------------------------

Product Bundles

Commerce Cloud Digital product bundles:

- Are a separate product with its own SKU and price
- Are unaffected by the price of the individual products
- Contain specified products in specified numbers

In respect to subscription products, product bundles can be treated the same as product variants. The end user will select subscription options for the entire bundle and the single product ID for the bundle is what will be configured and transmitted to Subscribe Pro when creating subscriptions.

See the instructions how to configure a Standard Product above.

Sales Force Commerce Cloud will automatically add all of the bundled products (children) to the Basket when the product bundle is added, whether it be from the storefront or from OCAPI.

The following is an example of an OCAPI request and response using product bundle: sony-ps3-bundle:

Request

```
{
  "customer_info": {
    "customer_no": "00000001",
    "email": "jkimmell@fluid.com"
  },
  "billing_address": {
    "first_name": "James",
    "last_name": "Kimmell",
    "address1": "14137 Louise Drive",
    "city": "Cumberland",
    "state_code": "MD",
    "country_code": "US"
  },
  "product_items": [
    {
      "product_id": "sony-ps3-bundle",
      "quantity": 1
    }
  ],
  "shipments": [
    {
      "gift": false,
      "shipping_method": {
        "id": "001"
      },
      "shipping_address": {
        "first_name": "James",
        "last_name": "Kimmell",
        "address1": "14137 Louise Drive",
        "city": "Cumberland",
        "state_code": "MD",
        "country_code": "US"
      }
    }
  ]
}
```

Response

```
{  
    "_v": "17.3",  
    "_type": "basket",  
    "_resource_state": "22e95720919f1eaa894bda606664b5a61a9575e0a942a1452bbfdb7062da0f69",  
    "_flash": [  
        {  
            "_type": "flash",  
            "type": "PaymentMethodRequired",  
            "message": "No payment method ID was specified. Please provide a valid payment  
method ID.",  
            "path": "$.payment_instruments[0].payment_method_id"  
        }  
    ],  
    "adjusted_merchandise_total_tax": 22.45,  
    "adjusted_shipping_total_tax": 0.5,  
    "agent_basket": true,  
    "basket_id": "16f3006604e77fff7105caf63a",  
    "billing_address": {  
        "_type": "order_address",  
        "address1": "14137 Louise Drive",  
        "city": "Cumberland",  
        "country_code": "US",  
        "first_name": "James",  
        "full_name": "James Kimmell",  
        "id": "b600c40aec47165cacc48c7ad2",  
        "last_name": "Kimmell",  
        "state_code": "MD"  
    },  
    "channel_type": "callcenter",  
    "creation_date": "2017-04-19T15:26:23.314Z",  
    "currency": "USD",  
    "customer_info": {  
        "_type": "customer_info",  
        "customer_id": "abKT2hbWsyyV8alhk6VJg90w0E",  
        "customer_no": "00000001",  
        "email": "jkimmell@fluid.com"  
    },  
    "last_modified": "2017-04-19T15:26:23.360Z",  
    "merchandise_total_tax": 22.45,  
    "notes": {  
        "_type": "simple_link",  
        "link": "https://subscribepr01-tech-prtnr-na02-  
dw.demandware.net/s/SiteGenesis/dw/shop/v17_3/baskets/16f3006604e77fff7105caf63a/notes"  
    },  
    "order_total": 481.94,  
    "product_items": [  
        {  
            "_type": "product_item",  
            "adjusted_tax": 22.45,  
            "base_price": 449,  
            "bonus_product_line_item": false,  
            "bundled_product_items": [  
                {  
                    "_type": "product_item",  
                    "adjusted_tax": null,  
                    "base_price": null,  
                    "bonus_product_line_item": false,  
                    "item_id": "94921b91f537474e71376abaf2",  
                    "quantity": 1  
                }  
            ]  
        }  
    ]  
}
```

```

    "item_text": "Sony Playstation 3 Game Console",
    "price": null,
    "price_after_item_discount": null,
    "price_after_order_discount": 0,
    "product_id": "sony-ps3-console",
    "product_name": "Sony Playstation 3 Game Console",
    "quantity": 1,
    "shipment_id": "me",
    "tax": null,
    "tax_basis": null,
    "tax_class_id": "standard",
    "tax_rate": 0.05
},
{
    "_type": "product_item",
    "adjusted_tax": null,
    "base_price": null,
    "bonus_product_line_item": false,
    "item_id": "6356afdf70d4a120da36a9d3baf",
    "item_text": "Nascar 09 (for Sony PS3)",
    "price": null,
    "price_after_item_discount": null,
    "price_after_order_discount": 0,
    "product_id": "easports-nascar-09-ps3",
    "product_name": "Nascar 09 (for Sony PS3)",
    "quantity": 1,
    "shipment_id": "me",
    "tax": null,
    "tax_basis": null,
    "tax_class_id": "standard",
    "tax_rate": 0.05
},
{
    "_type": "product_item",
    "adjusted_tax": null,
    "base_price": null,
    "bonus_product_line_item": false,
    "item_id": "8b606524f0c7a7325cc4e4e23a",
    "item_text": "Monopoly Here and Now: The World Edition (for Sony PS3)",
    "price": null,
    "price_after_item_discount": null,
    "price_after_order_discount": 0,
    "product_id": "easports-monopoly-ps3",
    "product_name": "Monopoly Here and Now: The World Edition (for Sony PS3)",
    "quantity": 1,
    "shipment_id": "me",
    "tax": null,
    "tax_basis": null,
    "tax_class_id": "standard",
    "tax_rate": 0.05
},
{
    "_type": "product_item",
    "adjusted_tax": null,
    "base_price": null,
    "bonus_product_line_item": false,
    "item_id": "0798adadda95b0a5c4d282980f",
    "item_text": "Eternal Sonata (for Sony PS3)",
    "price": null,
    "price_after_item_discount": null,
    "price_after_order_discount": 0,
    "product_id": "namco-eternal-sonata-ps3",
}

```

```

        "product_name": "Eternal Sonata (for Sony PS3)",
        "quantity": 1,
        "shipment_id": "me",
        "tax": null,
        "tax_basis": null,
        "tax_class_id": "standard",
        "tax_rate": 0.05
    },
    {
        "_type": "product_item",
        "adjusted_tax": null,
        "base_price": null,
        "bonus_product_line_item": false,
        "item_id": "b0bfeff3a8214ea603c570d73b",
        "item_text": "Warhawk (for Sony PS3)",
        "price": null,
        "price_after_item_discount": null,
        "price_after_order_discount": 0,
        "product_id": "sony-warhawk-ps3",
        "product_name": "Warhawk (for Sony PS3)",
        "quantity": 1,
        "shipment_id": "me",
        "tax": null,
        "tax_basis": null,
        "tax_class_id": "standard",
        "tax_rate": 0.05
    }
],
"item_id": "8ddceb64e1d3856008a753122a",
"item_text": "Playstation 3 Bundle",
"option_items": [
    {
        "_type": "option_item",
        "adjusted_tax": 0,
        "base_price": 0,
        "bonus_product_line_item": false,
        "item_id": "392333b488e166df0b46927548",
        "item_text": "Extended Warranty: None",
        "option_id": "consoleWarranty",
        "option_value_id": "000",
        "price": 0,
        "price_after_item_discount": 0,
        "price_after_order_discount": 0,
        "product_id": "000",
        "product_name": "Extended Warranty: None",
        "quantity": 1,
        "shipment_id": "me",
        "tax": 0,
        "tax_basis": 0,
        "tax_class_id": "standard",
        "tax_rate": 0.05
    }
],
"price": 449,
"price_after_item_discount": 449,
"price_after_order_discount": 449,
"product_id": "sony-ps3-bundle",
"product_name": "Playstation 3 Bundle",
"quantity": 1,
"shipment_id": "me",
"tax": 22.45,
"tax_basis": 449,

```

```

        "tax_class_id": "standard",
        "tax_rate": 0.05
    }
],
"product_sub_total": 449,
"product_total": 449,
"shipments": [
{
    "_type": "shipment",
    "adjusted_merchandise_total_tax": 22.45,
    "adjusted_shipping_total_tax": 0.5,
    "gift": false,
    "merchandise_total_tax": 22.45,
    "product_sub_total": 449,
    "product_total": 449,
    "shipment_id": "me",
    "shipment_total": 481.94,
    "shipping_address": {
        "_type": "order_address",
        "address1": "14137 Louise Drive",
        "city": "Cumberland",
        "country_code": "US",
        "first_name": "James",
        "full_name": "James Kimmell",
        "id": "29410763e35c40915b515a0cdd",
        "last_name": "Kimmell",
        "state_code": "MD"
    },
    "shipping_method": {
        "_type": "shipping_method",
        "description": "Order received within 7-10 business days",
        "id": "001",
        "name": "Ground",
        "price": 9.99
    },
    "shipping_status": "not_shipped",
    "shipping_total": 9.99,
    "shipping_total_tax": 0.5,
    "tax_total": 22.95
},
],
"shipping_items": [
{
    "_type": "shipping_item",
    "adjusted_tax": 0.5,
    "base_price": 9.99,
    "item_id": "1ad26e6d3776f228e7d56228d7",
    "item_text": "Shipping",
    "price": 9.99,
    "price_after_item_discount": 9.99,
    "shipment_id": "me",
    "tax": 0.5,
    "tax_basis": 9.99,
    "tax_class_id": "standard",
    "tax_rate": 0.05
}
],
"shipping_total": 9.99,
"shipping_total_tax": 0.5,
"taxation": "net",
"tax_total": 22.95
}
]
}

```

Product Sets

Product Set is a product that includes several products that are displayed together and can be purchased together or separately. For example, an accessories kit that includes a hairbrush, comb, and mirror, each of which can also be bought separately. Another common example is an outfit, with a jacket, shirt, and pants, which can be purchased together or each item can be purchased separately.

Unlike Product Bundles, Products Set Products are added to the cart individually. The end consumer has the option to add each product individually or to add all of the products at once.

When the consumer adds all of the products at once, the form fields for each individual product are preserved, for example quantity and subscription options.

Products within a product set can be configured the same as a standard product and the user will have the ability to configure the subscription options and interval for each individual product. See the instructions for configuring a Standard Product above.

Option Product

Commerce Cloud Digital, product options enable you to sell configurable products that have optional accessories, upgrades, or additional services. Options are always purchased with a product and cannot be purchased separately. They have their own price and display name, but do not have their own thumbnail images. They cannot be searched by the customer, but are usually visible on the product detail page.

Products that are configured with Product Options should be configured the same as a normal subscription product in Commerce Cloud and Subscribe Pro. The only difference with these products is that the Order Processing Script, that creates the subscription at Subscribe Pro, will also include any product options, selected for the product line item, in the platform_specific_fields attribute.

Subscribe Pro Subscription Request

```
{  
    "subscription":{  
        "customer_id":"348328",  
        "payment_profile_id":"410028",  
        "requires_shipping":true,  
        "shipping_address_id":455367,  
        "product_sku":"canon-powershot-e1",  
        "qty":1,  
        "use_fixed_price":false,  
        "interval":"Month",  
        "next_order_date":"2017-04-19T21:33:56.954Z",  
        "first_order_already_created":true,  
        "send_customer_notification_email":true,  
        "platform_specific_fields":{  
            "sfcc":{  
                "product_options": [  
                    {  
                        "id":"digitalCameraWarranty",  
                        "value":"002"  
                    }  
                ]  
            }  
        }  
    }  
}
```

```

        }
    ]
}
}
}
```

These product options will need to be added to the Basket when recurring orders are placed through OCAPI. The following is an example of a Basket object, with a product option, that could be sent back to Commerce Cloud via OCAPI:

OCAPI Basket Object

```
{
  "customer_info": {
    "customer_no": "00000001",
    "email": "jkimmell@fluid.com"
  },
  "billing_address": {
    "first_name": "James",
    "last_name": "Kimmell",
    "address1": "14137 Louise Drive",
    "city": "Cumberland",
    "state_code": "MD",
    "country_code": "US"
  },
  "product_items": [
    {
      "product_id": "samsung-h172a650",
      "quantity": 1,
      "option_items": [
        {
          "option_id": "tvWarranty",
          "option_value_id": "002"
        }
      ]
    }],
  "shipments": [
    {
      "gift": false,
      "shipping_method": {
        "id": "001"
      },
      "shipping_address": {
        "first_name": "James",
        "last_name": "Kimmell",
        "address1": "14137 Louise Drive",
        "city": "Cumberland",
        "state_code": "MD",
        "country_code": "US"
      }
    }
  ]
}
```

Configure OCAPI client

Subscribe Pro will make use of the "Shop" Open Commerce API (OCAPI) in order to place recurring orders for subscription products. This will be accomplished server to server using a Business Manager login.

Configure client's resources as following:

```
{  
    "_v": "17.3",  
    "clients": [  
        {  
            "client_id": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",  
            "allowed_origins": [],  
            "resources": [  
                {  
                    "resource_id": "/baskets",  
                    "methods": [  
                        "post"  
                    ],  
                    "read_attributes": "(**)",  
                    "write_attributes": "(**)"  
                },  
                {  
                    "resource_id": "/baskets/*",  
                    "methods": [  
                        "get",  
                        "patch",  
                        "delete"  
                    ],  
                    "read_attributes": "(**)",  
                    "write_attributes": "(**)"  
                },  
                {  
                    "resource_id": "/baskets/*/payment_instruments",  
                    "methods": [  
                        "post"  
                    ],  
                    "read_attributes": "(**)",  
                    "write_attributes": "(**)"  
                },  
                {  
                    "resource_id": "/orders",  
                    "methods": [  
                        "post"  
                    ],  
                    "read_attributes": "(**)",  
                    "write_attributes": "(**)"  
                }  
            ]  
        }  
    ]  
}
```

Configure Payments

Subscribe Pro will make use of OCAPI to place orders on behalf of the customer. This is one of the reason's that a customer must be logged in and have a stored payment instrument in order to place an order with subscriptions.

Credit Cards

A Payment token and the Payment Instrument ID will be transmitted to Subscribe Pro as part of the Subscription creation process.

Credit Card tokenization will need to be implemented by a tokenization provider when this cartridge is implemented, on an actual site. This will be accomplished by making use of the: setCreditCardToken method in the: PaymentInstrument object.

See SFCC documentation:

https://documentation.demandware.com/DOC1/topic/com.demandware.dochelp/DWAPI/scriptapi/html/api/class_dw_order_PaymentInstrument.html

In order to provide an example of this, on the base site, some code has been added the COBilling controller to generate a random token for credit cards:

```
/**  
 * This needs to be replaced with a valid PSP Tokenizer before going to production  
 */  
let random = new dw.crypto.SecureRandom();  
let bytes = random.nextBytes(32);  
let tokenString = newCreditCard.getUUID() + new Date().getTime() + bytes.toString();  
let messageDigest = new dw.crypto.MessageDigest(dw.crypto.MessageDigest.DIGEST_SHA_512);  
let hash = dw.crypto.Encoding.toBase64(messageDigest.digestBytes(new  
dw.util.Bytes(tokenString)));  
  
newCreditCard.setCreditCardToken(hash);
```

Apple Pay

Customers will be able to use Apple Pay, while purchasing subscription products, using the Subscribe Pro Apple Pay service. Subscribe Pro will handle making the necessary API calls, transactions with Apple and will provide Sales Force with a transaction id of the payment.

The following is the SFCC documentation for Apple Pay:

<https://documentation.demandware.com/DOC1/topic/com.demandware.dochelp/ApplePay/APWProcess.html>

Basic ApplePay for Site Genesis

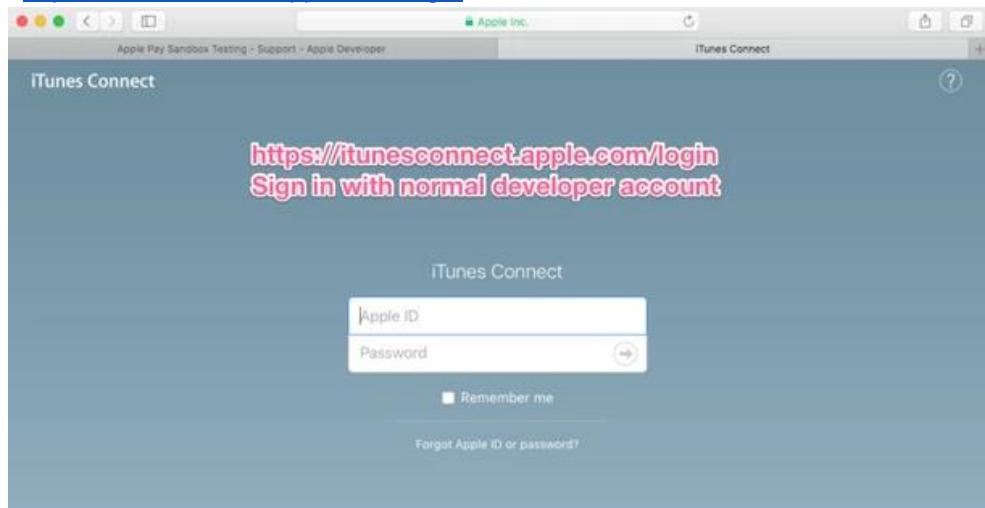
Create Apple Sandbox User

In order to complete a payment, in a Sandbox environment, not using real payment information, an Apple

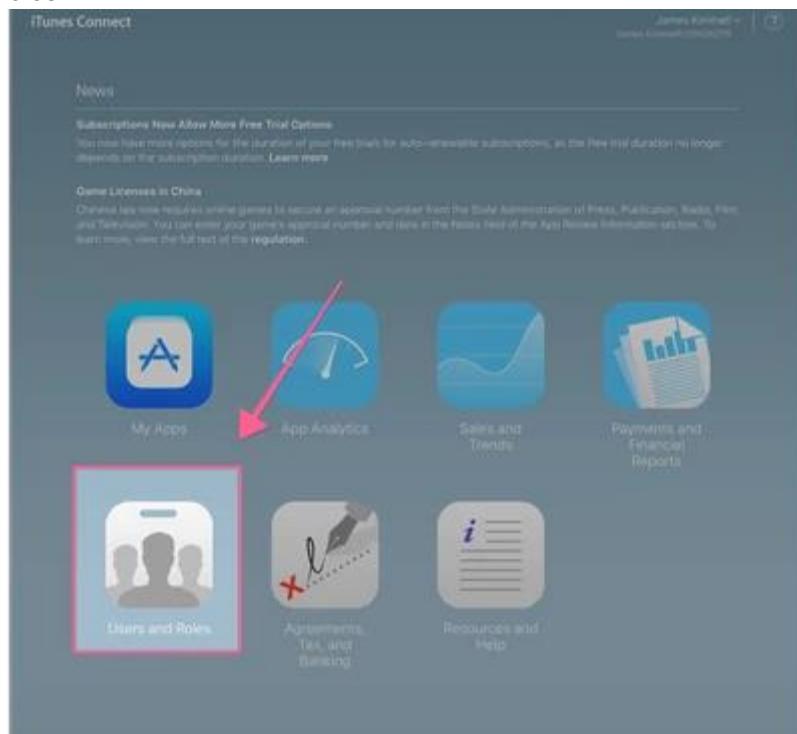
Sandbox Tester account needs to be created.

The following screenshots show how to accomplish this.

Proceed to: <https://itunesconnect.apple.com/login>



Click: Users and Roles



Click: Sandbox Testers

iTunes Connect Users and Roles

Sandbox Testers

User (1)	Apple ID	Name	Role	Apps
	kimmellj@gmail.com	James Kimmell	Admin, Legal	All Apps

Click the plus icon to create a new user

iTunes Connect Users and Roles

TestFlight Beta Testers

Testers (2)	Email	Name	iTunes Store	Apple Pay
	jamie@jkimmell.com	Jamie Sandbox	United States	✓

Enter the details of the user on the provided form

Note: You will need a separate email address from your developer account email address

Enter details and save

Note that you will need a separate email account from the one you logged in with that you have access to. You will need to verify the account.

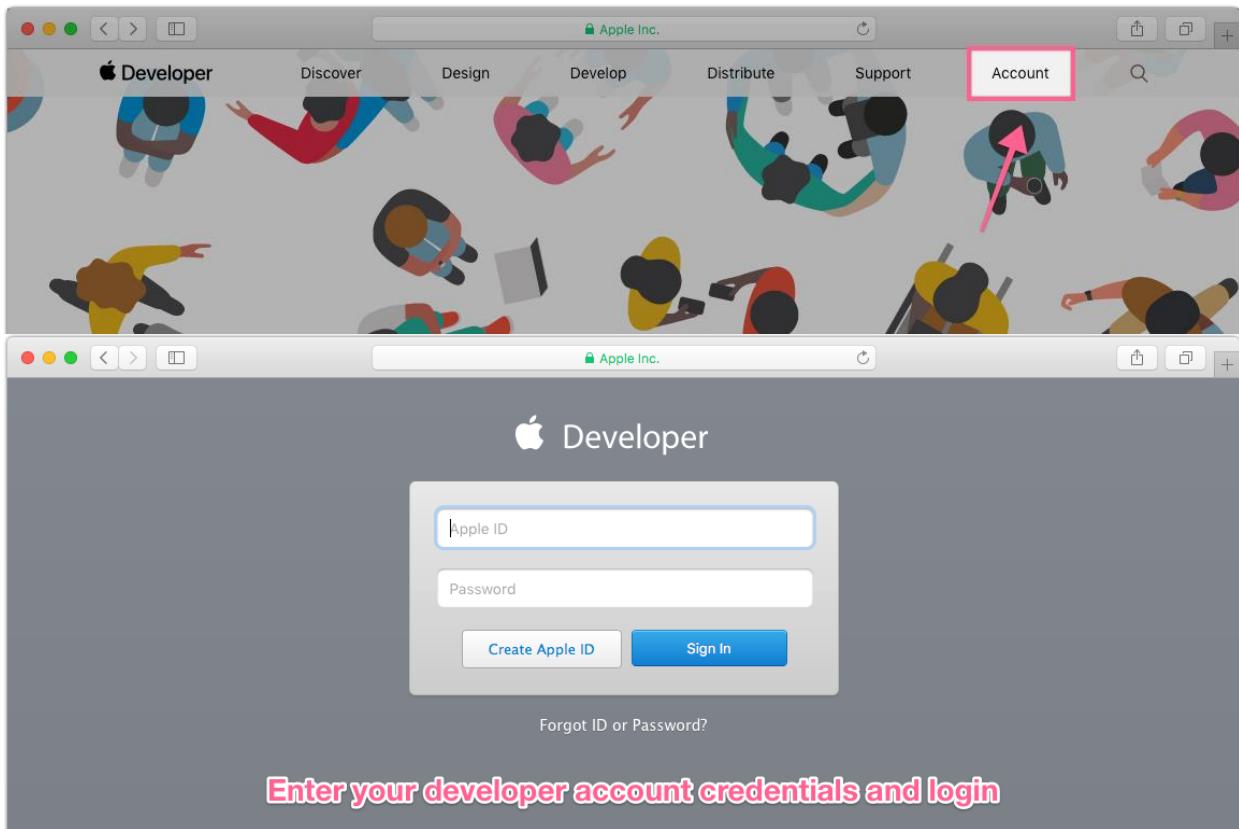
Sandbox testers can test your apps in development mode and buy In-App Purchases without making real transactions. Users added after June 13, 2016 can also test transactions using Apple Pay.

Tester Information

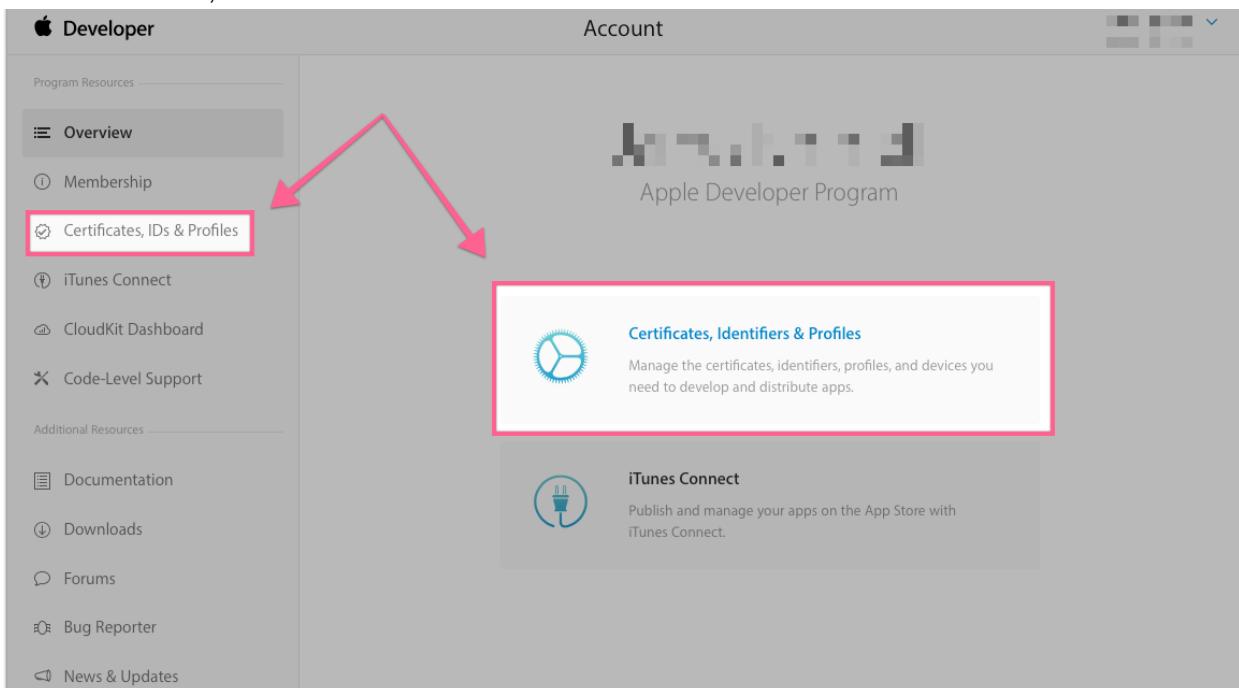
First Name	Last Name
<input type="text"/>	<input type="text"/>
Email	
<input type="text"/>	
Password	Confirm Password
<input type="password"/>	<input type="password"/>
Secret Question	Secret Answer
<input type="text"/>	<input type="text"/>
Date of Birth	App Store Territory
Month	Day
<input type="text"/>	<input type="text"/>

Create Merchant and Payment Provider Certificate

Login to the Apple Developer Portal: <https://developer.apple.com>



Click: Certificates, IDs & Profiles



Click: Merchant IDs and then Click: the plus icon to create a new Merchant

Add new Merchant ID

Merchant IDs

2 Merchant IDs total.

Name	ID
[REDACTED]	[REDACTED]
[REDACTED]	[REDACTED]

Go to the Merchant IDs section

Enter a Merchant ID and Description, these will be needed to configure Business Manager

Register Merchant IDs

ID **Registering a Merchant ID**

Register your Merchant Identifiers (Merchant IDs) to enable your apps to process transactions for physical goods and services to be used outside of your apps. Generate a Apple Pay certificate for each registered Merchant ID to validate transactions initiated within your app.

Merchant ID Description

Description: Foo Bar Merchant ID
You cannot use special characters such as @, &, *, ^, "

In BM this will be the: Apple Merchant Name

Identifier

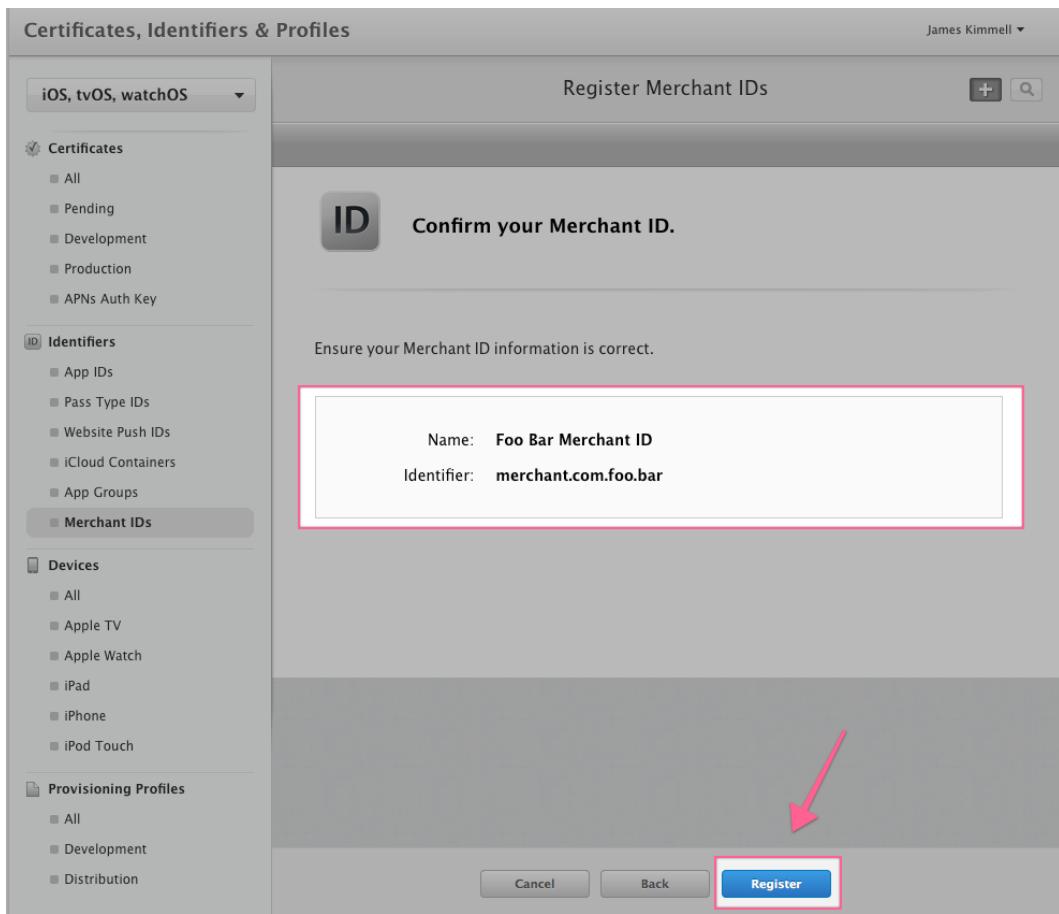
Enter a unique identifier for your Merchant ID, starting with the string 'merchant'.

ID: merchant.com.foo.bar
We recommend using a reverse-domain name style string (i.e., merchant.com.example.merchantname).

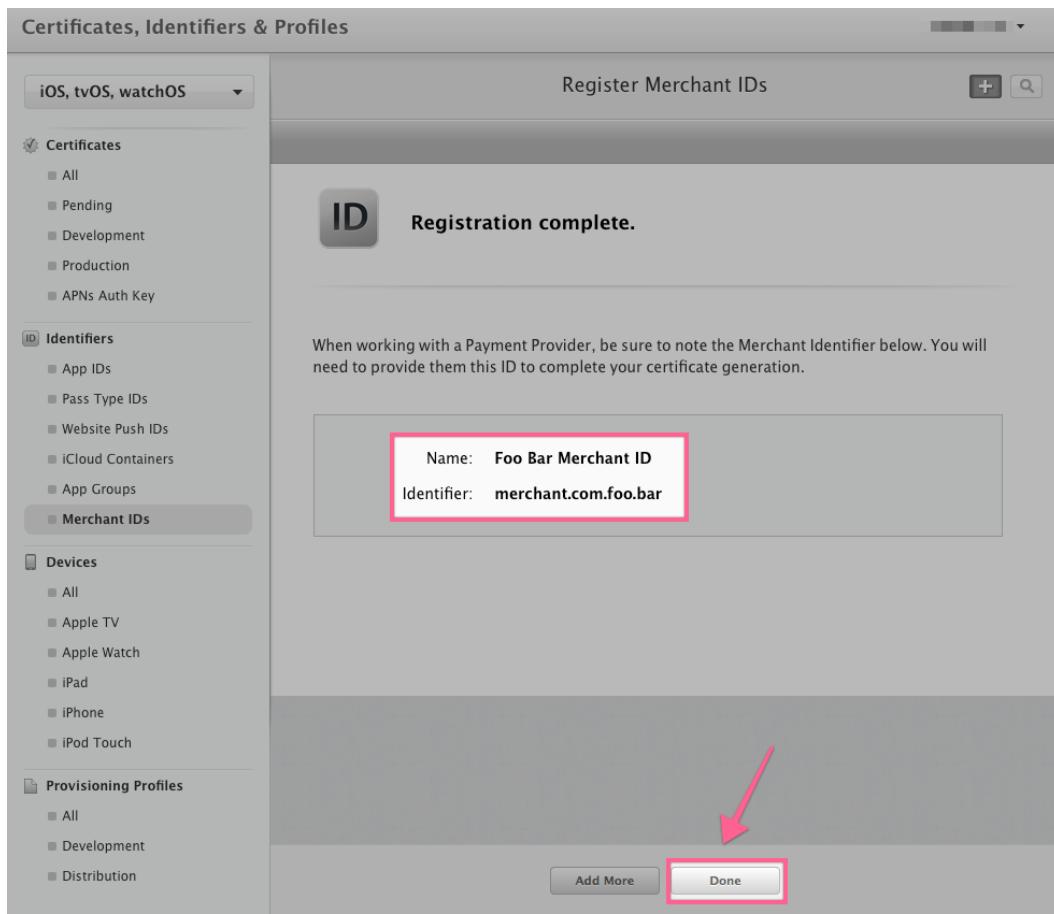
In BM this will be the: Apple Merchant ID

Cancel Continue

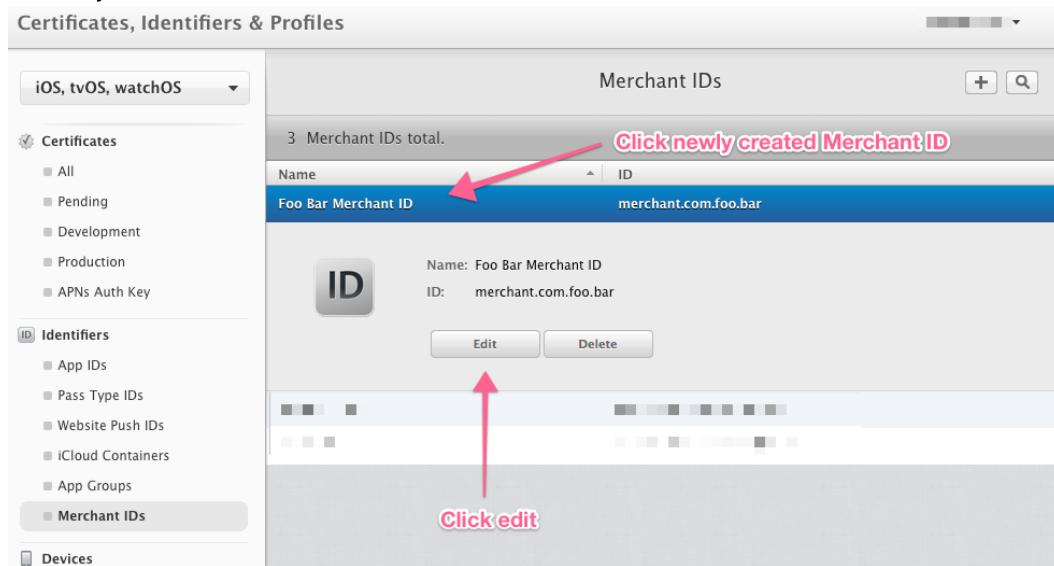
Click: Register



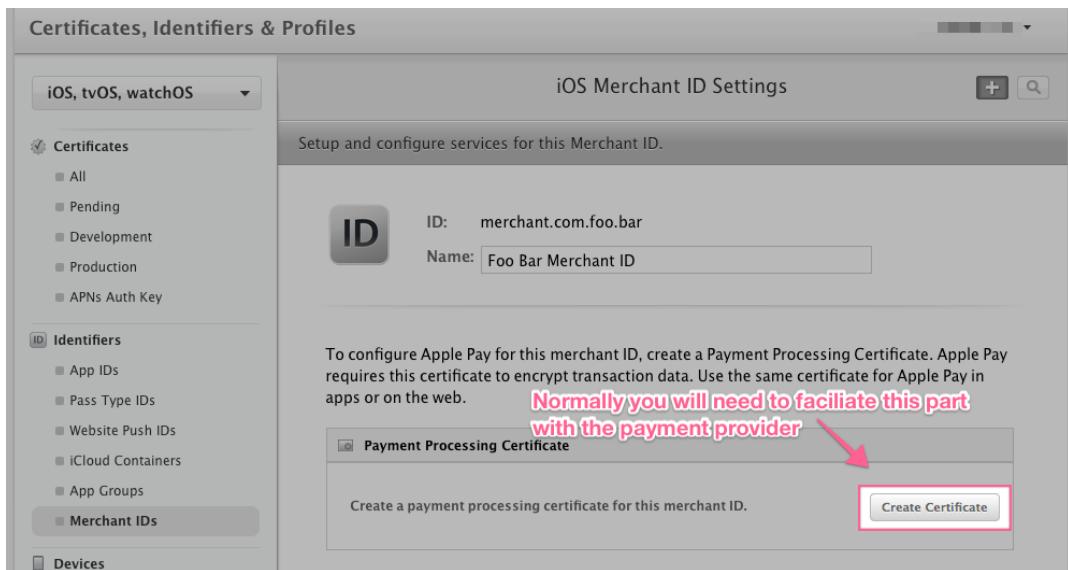
Click: Finish



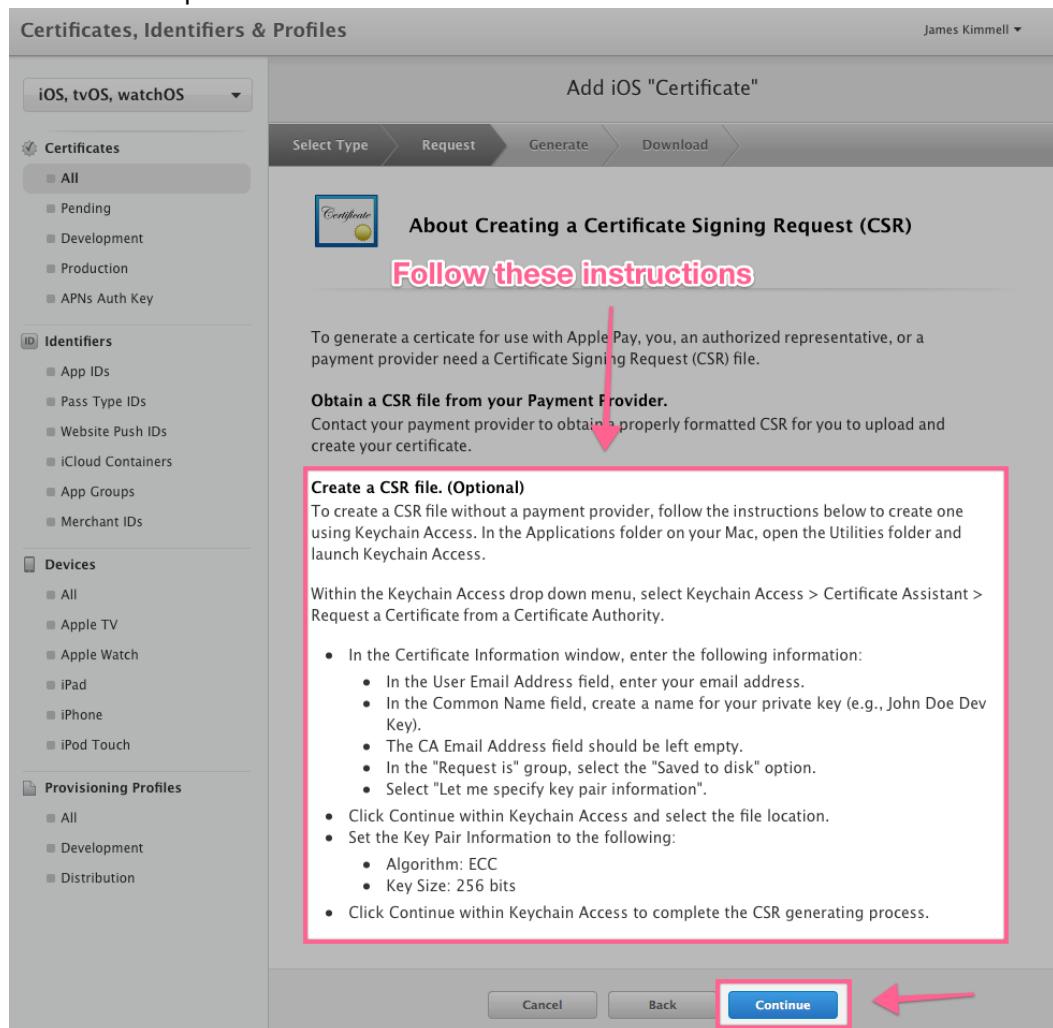
Select the newly created Merchant and then click: Edit



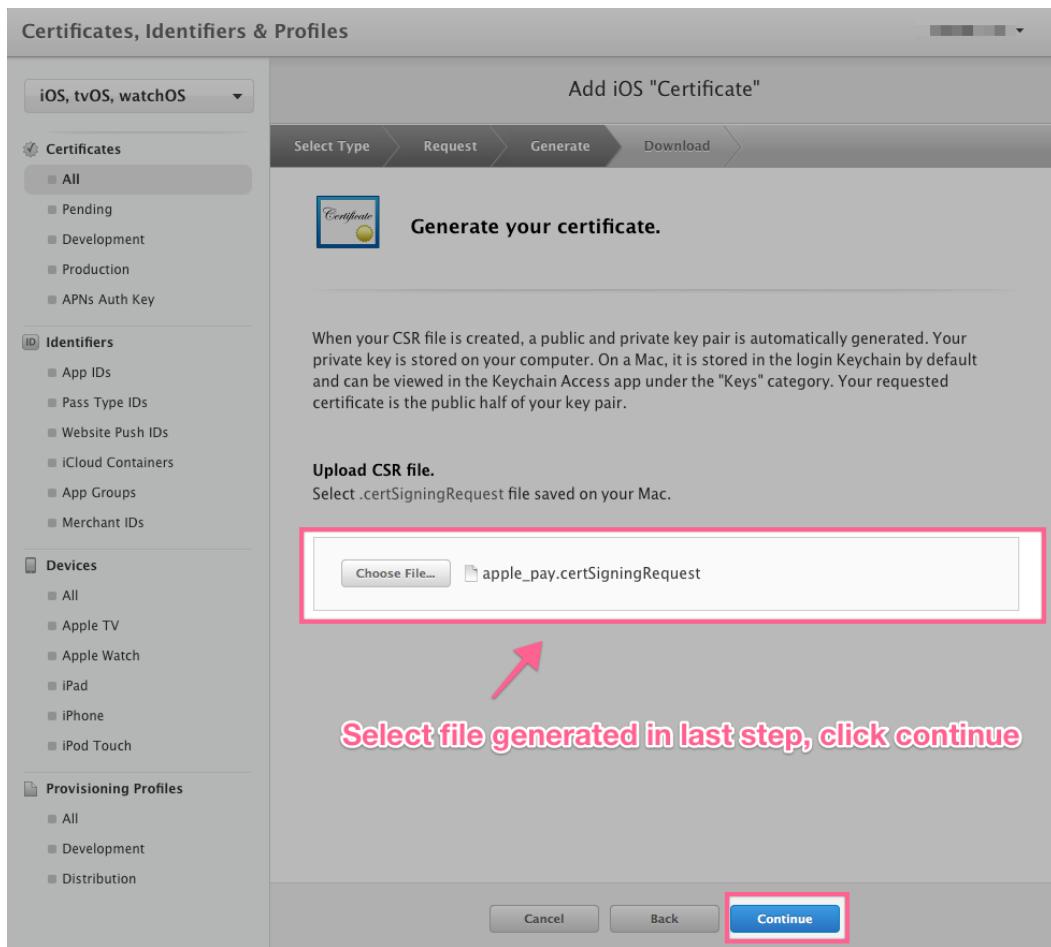
Click: Create Certificate



Follow the instructions provided to create a CSR and then click: Continue



Choose newly created CSR and click: Continue



Download, save, backup and click: Done

Certificates, Identifiers & Profiles

Add iOS "Certificate"

Select Type Request Generate Download

Your certificate is ready.

Follow instructions to create a backup

Download, Install and Backup

Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure.

Name: Apple Pay Payment Processing:merchant.com.foo.bar
Type: Apple Pay
Expires: Apr 30, 2019

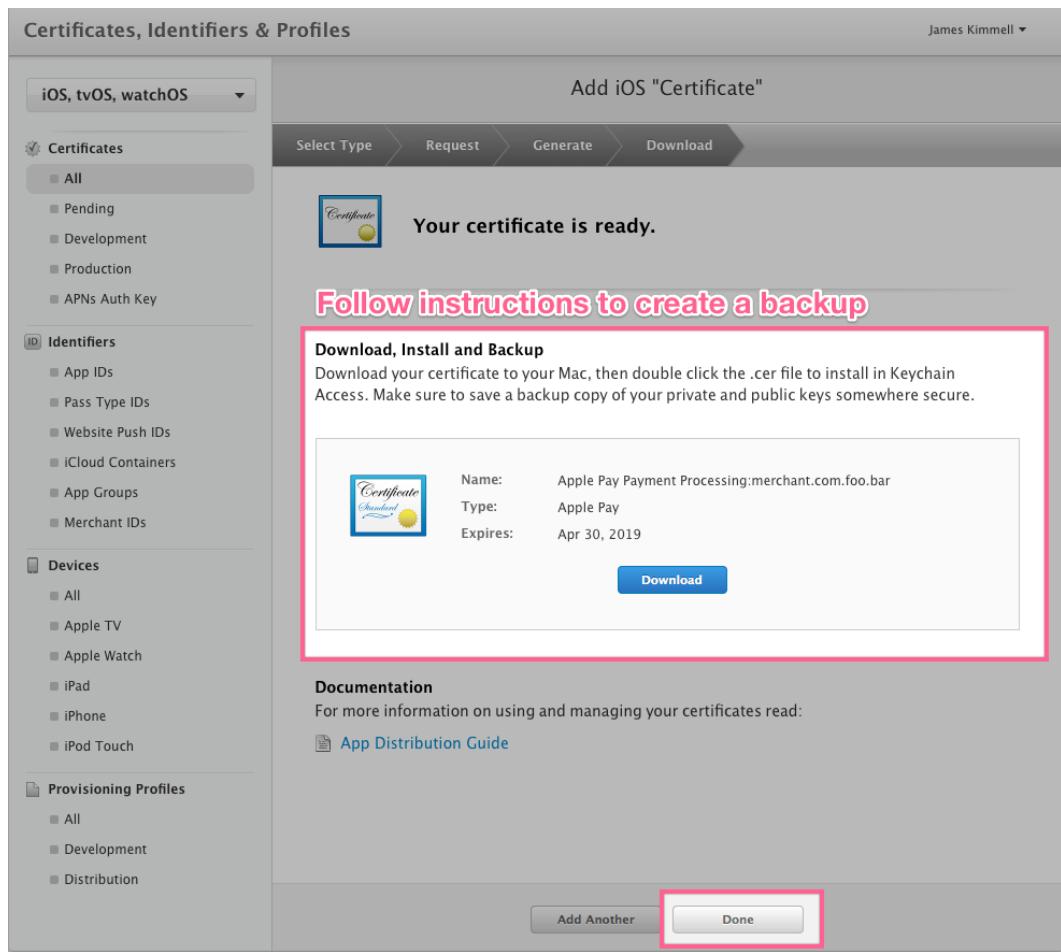
Download

Documentation

For more information on using and managing your certificates read:

[App Distribution Guide](#)

Add Another Done



Configure Site in Business Manager

Click: Site Preferences

Salesforce Sandbox - subscribepr... SiteGenesis Merchant Tools Administration Storefront James Kimmell

Merchant Tools

Use the modules below to manage site-specific aspects of the application.

 Products and Catalogs Manage the catalogs and products of this site.	 Content Manage the non-product content of this site.
 Search Manage storefront search indexes and sorting options of this site.	 Online Marketing Manage the online marketing activities of this site.
 Customers Manage the customers of this site.	 Custom Objects Manage custom objects of this site.
 Ordering Manage the orders of this site.	 Analytics Browse reports of this site.
 Site URLs Manage site aliases and URL mapping rules.	 Site Preferences Set preferences for this site.

Click: Apple Pay

Salesforce Site Genesis

Sandbox - subscribepr... SiteGenesis

Merchant Tools Administration Storefront

(James Kimmell)

Site Preferences

Business administrators use the following modules to define various preferences for the storefront site.

- Locking** Define the locking preferences for products and content assets.
- Baskets** Define the behavior of baskets on this site.
- A/B Tests** Define the behavior of A/B tests on this site.
- Locales** Define the site's allowed and default locales.
- Currencies** Define the site's currencies.
- Source Codes** Manage source code settings.
- Gift Certificates** Manage the gift certificates preferences for this site.
- Search Preferences** Manage the search preferences for this site.
- Sequence Numbers** Manage sequence numbers assigned to your orders.
- Order** Manage essential order settings for this site.
- Coupons** Manage the coupon preferences for this site.
- Promotions** Manage the promotion preferences for this site.
- Storefront Toolkit** Manage the storefront toolkit preferences for this site.
- Storefront URLs** Configure storefront URL preferences.
- Custom Preferences** Configure custom site preferences.
- Pinterest Commerce** Manage Pinterest Commerce configuration.
- Customer Service Center Preferences** Manage the Customer Service Center preferences for this site.
- Apple Pay** Manage Apple Pay configuration.

© 2017 salesforce.com, inc. All Rights Reserved. SiteGenesis Time Zone: Coordinated Universal Time | instance time zone: Eastern Daylight Time | version: 17.3 , last updated Mar 14, 2017 (Compatibility Mode: 16.2)

Update Settings

Note: Merchant information must match info entered at Apple. Select only 3DS and Visa.

Apple Pay

Please enter your site information below. Fields marked with an asterisk (*) are mandatory. Click **Submit** when you have finished updating your configuration.

Instance Type: Sandbox/Development

Apple Pay Enabled?

Onboarding

Merchant Settings should match the ones at Apple

Apple Merchant ID*: merchant.com.foo.bar

Apple Merchant Name*: Foo Bar Merchant ID

Country Code*: US

Merchant Capabilities*: 3DS EMV Credit Debit **Select only 3DS**

Supported Networks*: Amex China UnionPay Discover Interac MasterCard Private Label Visa **Select only Visa**

Required Shipping Address Fields: Email Name Phone Postal Address

Required Billing Address Fields: Name Postal Address

Storefront Injection

Inject Apple Pay Button on Mini Cart?

Inject Apple Pay Button on Cart Page?

Redirect Pages to HTTPS?

Payment Integration

Use Commerce Cloud Apple Pay Payment API?

Payment Provider URL*:

Payment Provider Merchant ID*:

API Version*: v1

Use Basic Authorization?

Save and Register Sandbox

Sandbox - subscribepr... SiteGenesis

Merchant Tools Administration Storefront

James Kimmell

Required Billing Address Fields: Name, Postal Address

Storefront Injection

Inject Apple Pay Button on Mini Cart?

Inject Apple Pay Button on Cart Page?

Redirect Pages to HTTPS?

Payment Integration

Use Commerce Cloud Apple Pay Payment API?

Payment Provider URL*:

Payment Provider Merchant ID*:

API Version*: v1

Use Basic Authorization?

Payment Provider User*:

Payment Provider Password*:

Use JWS?

JWS Private Key Alias*:

Domain Registration

You must register the domain for your site with Apple in order to use Apple Pay.

The currently configured domain for this site is subscribepro01-tech-prtnr-na02-dw.demandware.net.

[Unregister Apple Sandbox](#) [Unregister Apple Production](#) [Register Apple Sandbox](#) [Register Apple Production](#)

Save first

Then register domain with Apple

Closing

You should see a blue message at the top of the page confirming that the Sandbox has been registered. You can now browse to a PDP page on HTTPS or the Shopping cart to pay with Apple Pay.

Payment Provider Certificate

In order to make use of Apple Pay payments for subscriptions a Payment Provider certificate will need to be generated at Subscribe Pro and added to the merchant in the relevant Apple Developer account. You can generate this certificate within the Subscribe Pro Platform in the following module:
System > ApplyPay Certificates

Hooks

Some modifications need to occur for Apple Pay to work on the Site Genesis site.

The following hook has been added to Site Genesis as an example but this will need to be removed:

Hook	File Location
dw.extensions.applepay.paymentAuthorized.authorizeOrderPayment	sitegenesis/app_storefront_core/cartridge/scripts/hooks.json

However, the following hook has been added in the Subscribe Pro cartridge and will need to remain:

Hook	File Location
dw.extensions.applepay.paymentAuthorized.createOrder	int_subscribe_pro/cartridge/scripts/hooks.json

The above hook is used to populate the Basket with the necessary flags before creating the order. This is done so that the Basket custom attributes transfer to the create Order Object.

Subscribe Pro Configuration

Go to *Merchant Tools > Site Preferences > Apple Pay*

Apple Pay

Please enter your site information below. Fields marked with an asterisk (*) are mandatory. Click Submit when you have finished updating your configuration.

Instance Type:	Sandbox/Development	Must match Apple Developer merchant																		
Apple Pay Enabled? <input checked="" type="checkbox"/>																				
Onboarding <table border="1"> <tr> <td>Apple Merchant ID:</td> <td>merchant.com.subscribe-pro.test-merchant</td> </tr> <tr> <td>Apple Merchant Name:</td> <td>subprotest</td> </tr> <tr> <td>Country Code:</td> <td>US</td> </tr> <tr> <td>Merchant Capabilities:</td> <td><input checked="" type="checkbox"/> 3DS <input type="checkbox"/> EMV <input type="checkbox"/> Credit <input type="checkbox"/> Debit</td> </tr> <tr> <td>Supported Networks:</td> <td><input type="checkbox"/> Amex <input type="checkbox"/> China UnionPay <input type="checkbox"/> Discover <input type="checkbox"/> Interac <input type="checkbox"/> MasterCard <input type="checkbox"/> Private Label <input checked="" type="checkbox"/> Visa</td> </tr> <tr> <td>Required Shipping Address Fields:</td> <td><input checked="" type="checkbox"/> Email <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Phone <input checked="" type="checkbox"/> Postal Address</td> </tr> <tr> <td>Required Billing Address Fields:</td> <td><input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Postal Address</td> </tr> </table>			Apple Merchant ID:	merchant.com.subscribe-pro.test-merchant	Apple Merchant Name:	subprotest	Country Code:	US	Merchant Capabilities:	<input checked="" type="checkbox"/> 3DS <input type="checkbox"/> EMV <input type="checkbox"/> Credit <input type="checkbox"/> Debit	Supported Networks:	<input type="checkbox"/> Amex <input type="checkbox"/> China UnionPay <input type="checkbox"/> Discover <input type="checkbox"/> Interac <input type="checkbox"/> MasterCard <input type="checkbox"/> Private Label <input checked="" type="checkbox"/> Visa	Required Shipping Address Fields:	<input checked="" type="checkbox"/> Email <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Phone <input checked="" type="checkbox"/> Postal Address	Required Billing Address Fields:	<input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Postal Address				
Apple Merchant ID:	merchant.com.subscribe-pro.test-merchant																			
Apple Merchant Name:	subprotest																			
Country Code:	US																			
Merchant Capabilities:	<input checked="" type="checkbox"/> 3DS <input type="checkbox"/> EMV <input type="checkbox"/> Credit <input type="checkbox"/> Debit																			
Supported Networks:	<input type="checkbox"/> Amex <input type="checkbox"/> China UnionPay <input type="checkbox"/> Discover <input type="checkbox"/> Interac <input type="checkbox"/> MasterCard <input type="checkbox"/> Private Label <input checked="" type="checkbox"/> Visa																			
Required Shipping Address Fields:	<input checked="" type="checkbox"/> Email <input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Phone <input checked="" type="checkbox"/> Postal Address																			
Required Billing Address Fields:	<input checked="" type="checkbox"/> Name <input checked="" type="checkbox"/> Postal Address																			
Storefront Injection <table border="1"> <tr> <td>Inject Apple Pay Button on Mini Cart?</td> <td><input checked="" type="checkbox"/></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Inject Apple Pay Button on Cart Page?</td> <td><input checked="" type="checkbox"/></td> <td></td> <td></td> <td></td> </tr> </table>			Inject Apple Pay Button on Mini Cart?	<input checked="" type="checkbox"/>				Inject Apple Pay Button on Cart Page?	<input checked="" type="checkbox"/>											
Inject Apple Pay Button on Mini Cart?	<input checked="" type="checkbox"/>																			
Inject Apple Pay Button on Cart Page?	<input checked="" type="checkbox"/>																			
Redirect Pages to HTTPS? <input checked="" type="checkbox"/>																				
Payment Integration <table border="1"> <tr> <td>Use Commerce Cloud Apple Pay Payment API?</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Payment Provider URL:</td> <td><input type="text"/></td> </tr> <tr> <td>Payment Provider Merchant ID:</td> <td><input type="text"/></td> </tr> <tr> <td>API Version:</td> <td>v1</td> </tr> <tr> <td>Use Basic Authorization?</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Payment Provider User:</td> <td><input type="text"/></td> </tr> <tr> <td>Payment Provider Password:</td> <td><input type="text"/></td> </tr> <tr> <td>Use JWS?</td> <td><input type="checkbox"/></td> </tr> <tr> <td>JWS Private Key Alias:</td> <td><input type="text"/></td> </tr> </table>			Use Commerce Cloud Apple Pay Payment API?	<input checked="" type="checkbox"/>	Payment Provider URL:	<input type="text"/>	Payment Provider Merchant ID:	<input type="text"/>	API Version:	v1	Use Basic Authorization?	<input checked="" type="checkbox"/>	Payment Provider User:	<input type="text"/>	Payment Provider Password:	<input type="text"/>	Use JWS?	<input type="checkbox"/>	JWS Private Key Alias:	<input type="text"/>
Use Commerce Cloud Apple Pay Payment API?	<input checked="" type="checkbox"/>																			
Payment Provider URL:	<input type="text"/>																			
Payment Provider Merchant ID:	<input type="text"/>																			
API Version:	v1																			
Use Basic Authorization?	<input checked="" type="checkbox"/>																			
Payment Provider User:	<input type="text"/>																			
Payment Provider Password:	<input type="text"/>																			
Use JWS?	<input type="checkbox"/>																			
JWS Private Key Alias:	<input type="text"/>																			

Subscribe Pro API Endpoint

Subscribe Pro OAuth Credentials

Domain Registration

You must register the domain for your site with Apple in order to use Apple Pay.

The currently configured domain for this site is subscribe-pro01-tech-prtnr-na02-dw.demandware.net.

Apple Sandbox

No domains are registered for Apple Sandbox environment.

[Unregister Apple Sandbox](#)

[Register Apple Sandbox](#)

Apple Production

No domains registered for Apple Production environment.

[Unregister Apple Production](#)

[Register Apple Production](#)

Notes:

Field	Notes
Apple Merchant ID	This must match the merchant created at Apple
Apple Merchant Name	This must match the merchant created at Apple

Use Commerce Cloud Apple Pay Payment API?	This must be checked
Payment Provider URL	This will be supplied by Subscribe Pro but the current URL in use is: https://feat-sfcc-cd7514i-cdbrvib5kpzoa.us.platform.sh/services/v2/vault/sfcc/applepay/authorize-and-store.json
Payment Provider Merchant ID	This will be supplied by Subscribe Pro
Use Basic Authorization?	This must be checked
Payment Provider User	This is the Subscribe Pro OAuth: Client ID
Payment Provider Password	This is the Subscribe Pro OAuth: Client Secret

Subscribe Pro - Payment Profile

Payment profile should be used to pay for subscriptions.

The use of this profile could be divided into two steps:

- creating subscription
- recurring subscription charging

Creating subscription

As part of the order processing script, we send the relevant Payment Data to the Subscribe Pro using the */vault/paymentprofile/external-vault end-point*.

The following is the relevant snippet of code in the Order Processing Logic:

/int_subscribe_pro/cartridge/scripts/subpro/jobs/ProcessOrderSubscriptions.js

```
/**
 * Apple Pay otherwise assume Credit Card
 */
if (paymentInstrument.getPaymentMethod() === "DW_APPLE_PAY") {
    let transactionID = paymentInstrument.getPaymentTransaction().getTransactionID();
    let response = SubscribeProLib.getPaymentProfile(null, transactionID);

    /**
     * If there was a problem creating the payment profile, error out
     */
    if (response.error) {
        logError(response, 'getPaymentProfile');
        continue;
    } else {
        paymentProfileID = response.result.payment_profiles.pop().id;
    }
} else {
    paymentProfileID = ('subproPaymentProfileID' in customerPaymentInstrument.custom) ?
customerPaymentInstrument.custom.subproPaymentProfileID : false;

    /**
     * If Payment Profile already exists,
     * Call service to verify that it still exists at Subscribe Pro
    */
}
```

```

/*
if (paymentProfileID) {

    let response = SubscribeProLib.getPaymentProfile(paymentProfileID);

    /**
     * Payment Profile not found, create new Payment Profile record
     * Otherwise create the payment profile
     */
    if (response.error && response.result.code === 404) {
        paymentProfileID = createSubproPaymentProfile(customerProfile,
customerPaymentInstrument, order.billingAddress);
        /**
         * Some other error occurred, error out
         */
    } else if (response.error) {
        logError(response, 'getPaymentProfile');
        continue;
    }
} else {
    paymentProfileID = createSubproPaymentProfile(customerProfile,
customerPaymentInstrument, order.billingAddress);
}
}

```

Recurring subscription charging

To charge a regular fee we should create OCAPI order using payment profile information saved earlier. Then using this data SubPro side will create order for paying for subscription via OCAPI.

1. Request POST /dw/shop/v17_3/baskets to create basket for order.
2. Use POST /dw/shop/v17_3/baskets/{basket_id}/payment_instruments to add payment instruments to basket. This requires fields payment_instrument_id and amount.
3. Request POST /dw/shop/v17_3/orders with basket_id attribute to create order.

3.3 Custom Code

The included site genesis cartridges, “LINK-SiteGenesis-subscribe-pro”, include any changes that would need to be made assuming you are using the Site Genesis reference storefront.

Templates

- addressinclude.isml
- cart.isml
- checkoutlogin.isml
- displayliproduct.isml
- minilineitems.isml
- multishippingshipments.isml
- mysubscriptions.isml
- orderdetails.isml
- orderdetailsemail.isml
- paymentinstrumentlist.isml
- paymentmethods.isml
- productcontent.isml
- producttopcontentPS.isml
- productsetproduct.isml
- pt_productdetails_UI.isml
- summary.isml
- creditcardjson.isml

Controllers

- Address.js
- PaymentInstruments.js

JS

- js/subscriptionOptions.js

Scripts

- Account.js
- COBilling.js
- COCustomer.js
- COSummary.js
- CartModel.js
- cart.js
- variant.js
- GetProductDetailUrl.ds
- billing.js

Forms

- creditcard.xml

Hooks

- hooks.json

Resource

- Resource.ds
- account.properties

subscriptionOptions.js Script

The cartridge/js/subscriptionOptions.js file in the int_subscribe_pro cartridge should be copied into the app_storefront_core/cartridge/js directory.

PDP

Changes need to be done to display product subscription options (is they are available) and Subscribe Pro “Add to Order” widget.

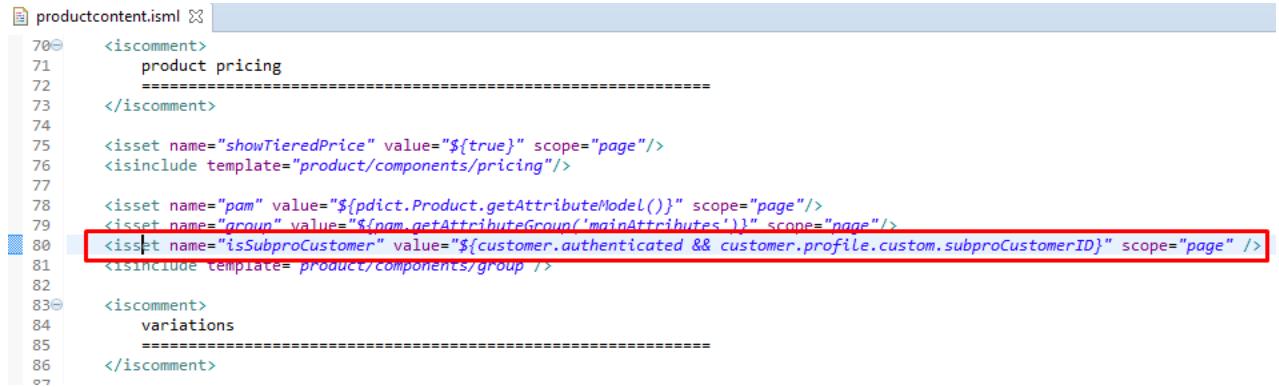
Modify *productcontent.isml*

Find template /app_storefront_core/cartridge/templates/default/product/productcontent.isml.

- 1) Set the variable specifying if current customer is a Subscribe Pro customer:

```
<isset name="isSubproCustomer" value="${customer.authenticated && customer.profile.custom.subproCustomerID}" scope="page" />
```

Here is the expected result:



```
70@    <iscomment>
71        product pricing
72        =====
73    </iscomment>
74
75    <isset name="showTieredPrice" value="${true}" scope="page"/>
76    <isinclude template="product/components/pricing"/>
77
78    <isset name="pam" value="${pdict.Product.getAttributeModel()}" scope="page"/>
79    <isset name="group" value="${nam.getAttributeGroup('mainAttributes')}" scope="page"/>
80    <isset name="isSubproCustomer" value="${customer.authenticated && customer.profile.custom.subproCustomerID}" scope="page" />
81    <isinclude template="product/components/group" />
82
83@    <iscomment>
84        variations
85        =====
86    </iscomment>
87
```

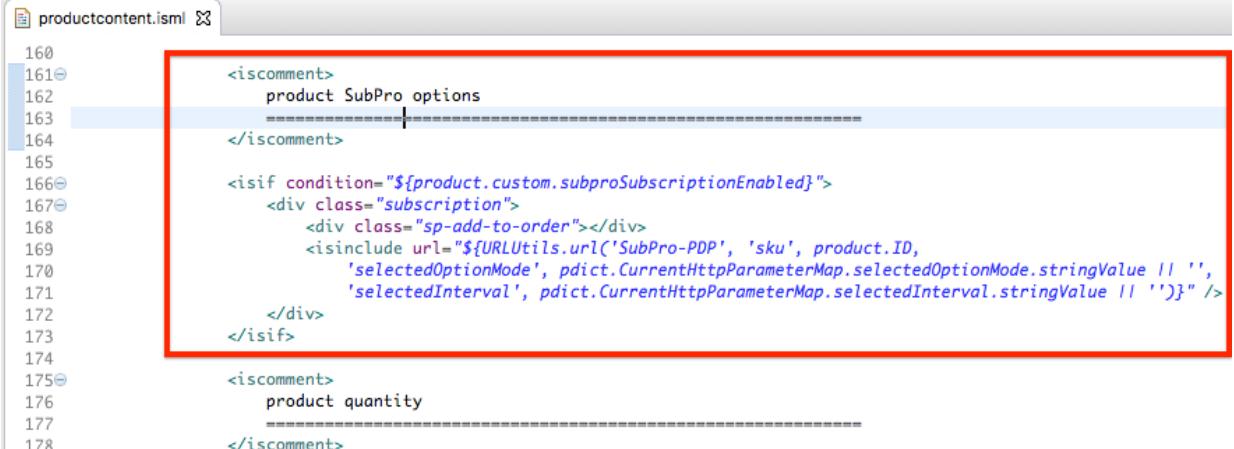
- 2) Scroll down to add to cart form, find “*product quantity*” section and paste code below before it to render product subscription options:

```
<iscomment>
    product SubPro options
    =====
</iscomment>

<isif condition="${product.custom.subproSubscriptionEnabled}">
    <div class="subscription">
        <div class="sp-add-to-order"></div>
        <isinclude url="${URLUtils.url('SubPro-PDP', 'sku', product.ID,
            'selectedOptionMode', pdict.CurrentHttpParameterMap.selectedOptionMode.stringValue
        || '',
            'selectedInterval', pdict.CurrentHttpParameterMap.selectedInterval.stringValue ||
        '')}" />
    </div>
```

```
</isif>
```

Here is the expected result:



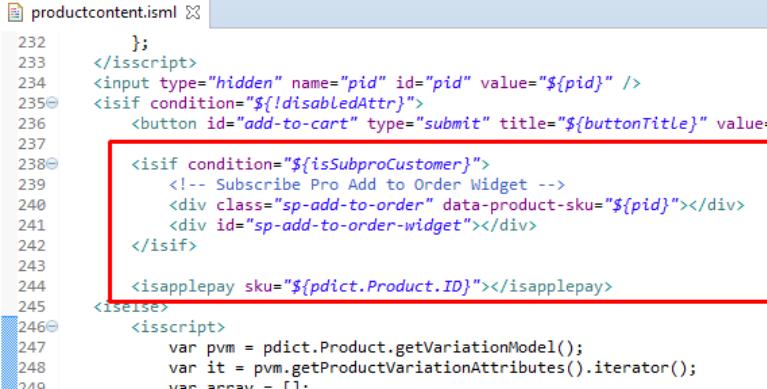
```
160
161<!--
162     product SubPro options
163     -----
164 -->
165
166<!--
167     ${product.custom.subproSubscriptionEnabled}
168     -----
169     <div class="subscription">
170         <div class="sp-add-to-order"></div>
171         <isinclude url="${URLUtils.url('SubPro-PDP', 'sku', product.ID,
172             'selectedOptionMode', pdict.CurrentHttpParameterMap.selectedOptionMode.stringValue || ''),
173             'selectedInterval', pdict.CurrentHttpParameterMap.selectedInterval.stringValue || '')}" />
174     </div>
175 -->
176     product quantity
177     -----
178 -->
```

- 3) Scroll down to “add-to-cart” button and insert code below after it to render Subscribe Pro “Add to Order” widget and inject Apple Pay button:

```
<isif condition="${isSubproCustomer}">
    <!-- Subscribe Pro Add to Order Widget -->
    <div class="sp-add-to-order" data-product-sku="${pid}"></div>
    <div id="sp-add-to-order-widget"></div>
</isif>

<isapplepay sku="${pdict.Product.ID}"></isapplepay>
```

Here is the expected result:



```
232     };
233     </isscript>
234     <input type="hidden" name="pid" id="pid" value="${pid}" />
235     <isif condition="${!disabledAttr}">
236         <button id="add-to-cart" type="submit" title="${buttonTitle}" value="${buttonTitle}" class="button-fancy-large
237
238             <isif condition="${isSubproCustomer}">
239                 <!-- Subscribe Pro Add to Order Widget -->
240                 <div class="sp-add-to-order" data-product-sku="${pid}"></div>
241                 <div id="sp-add-to-order-widget"></div>
242             </isif>
243
244             <isapplepay sku="${pdict.Product.ID}"></isapplepay>
245         </iselse>
246         <isscript>
247             var pvm = pdict.Product.getVariationModel();
248             var it = pvm.getProductVariationAttributes().iterator();
249             var array = [];
250             var options = '';
```

- 4) Add code to configure Subscribe Pro “Add to Order” widget:

```
<isif condition="${isSubproCustomer}">
    <iscomment>
        Configure the Subscribe Pro Add to Order Widget
    </iscomment>
```

```

<!-- Load the Subscribe Pro widget script -->
<isset name="widgetConfig" scope="page"
value="${require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/WidgetsHelper').getWidgetConfig(customer.profile.custom.subproCustomerID, 'client_credentials', 'widget', 'sp-add-to-order-widget', 'sp-add-to-order')}">

<!-- Pass configuration and init the Subscribe Pro widget -->
<script>
    // Setup config for Subscribe Pro
    var widgetConfig = {
        widgetElementId: '${widgetConfig.widgetElementId}',
        apiBaseUrl: '${widgetConfig.apiAccessToken}',
        apiAccessToken: '${widgetConfig.apiAccessToken}',
        environmentKey: '${widgetConfig.environmentKey}',
        customerId: '${widgetConfig.customerId}',
        addToOrderElementClass: '${widgetConfig.addToOrderElementClass}'
    };
    // Call init
    if (typeof subscribePro !== 'undefined') {
        subscribePro.widgets.init('add-to-order', widgetConfig);
    } else {
        $.getScript(
            '${require("dw/system/Site").getCurrent().getCustomPreferenceValue("subproWidgetScriptUrl")}',
            function( data, textStatus, jqxhr ) {
                subscribePro.widgets.init('add-to-order', widgetConfig);
            });
    }
</script>
</isset>
</isif>

```

Here is the expected result:

```

productcontent.isml
299@     <div id="update-images" style="display:none">
300         <isinclude template="product/components/productimages"/>
301     </div>
302 </isif>
303
304@ <isif condition="${isSubproCustomer}">
305@     <iscomment>
306         Configure the Subscribe Pro Add to Order Widget
307     </iscomment>
308
309     <!-- Load the Subscribe Pro widget script -->
310
311     <isset name="widgetConfig" scope="page"
312         value="${require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/WidgetsHelper').getWidgetConfig(custo
313
314     <!-- Pass configuration and init the Subscribe Pro widget -->
315@     <script>
316         // Setup config for Subscribe Pro
317         var widgetConfig = {
318             widgetElementId: '${widgetConfig.widgetElementId}',
319             apiBaseUrl: '${widgetConfig.apiAccessToken}',
320             apiAccessToken: '${widgetConfig.apiAccessToken}',
321             environmentKey: '${widgetConfig.environmentKey}',
322             customerId: '${widgetConfig.customerId}',
323             addToOrderElementClass: '${widgetConfig.addToOrderElementClass}'
324         };
325         // Call init
326         if (typeof subscribePro !== 'undefined') {
327             subscribePro.widgets.init('add-to-order', widgetConfig);
328         } else {
329             $.getScript( '${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subproWidgetScriptUrl')}'
330                 subscribePro.widgets.init('add-to-order', widgetConfig);
331             });
332         }
333     </script>
334 </isif>
335 </isif>
336

```

Modify productsetproduct.isml

Locate template

/app_storefront_core/cartridge/templates/default/product/components/productsetproduct.isml. Find input

<input type="hidden" name="pid" value="\${pdict.Product.ID}" />

Insert code below after it to render product subscription options.

```

<iscomment>
    product SubPro options
    =====
</iscomment>

<isif condition="${pdict.Product.custom.subproSubscriptionEnabled}">
    <div class="subscription">
        <div class="sp-add-to-order"></div>
        <isinclude url="${URLUtils.url('SubPro-PDP', 'sku', pdict.Product.ID)}" />
    </div>
</isif>

```

Here is the expected result:

```

67@      <isif condition="${!pdict.Product.variationGroup}">
68@          <div class="availability-web">
69@              <label>${Resource.msg('global.availability','locale',null)}</label>
70@              <span class="value"><isinclude template="product/components/availability"/></span>
71@          </div>
72@      </isif>
73@ 
74@ 
75@      <form action="${URLUtils.url('Cart-AddProduct')}" method="post" id="${uniqueFormID}">
76@          <input type="hidden" name="availability" value="${pdict.Product.availabilityModel.availabilityStatus}" />
77@          <input type="hidden" name="pid" value="${pdict.Product.ID}" />
78@ 
79@          <iscomment>
80@              product SubPro options
81@              =====
82@          </iscomment>
83@ 
84@          <isif condition="${pdict.Product.custom.subproSubscriptionEnabled}">
85@              <div class="subscription">
86@                  <div class="sp-add-to-order"></div>
87@                  <isinclude url="${URLUtils.url('SubPro-PDP', 'sku', pdict.Product.ID)}" />
88@              </div>
89@          </isif>
90@ 
91@          <div class="inventory">
92@              <div class="quantity">
93@                  <label for="${uniqueFormID}-quantity">${Resource.msg('global.qty','locale',null)}</label>
94@                  <input type="text" name="Quantity" id="${uniqueFormID}-quantity" maxlength="3" class="input-text">
95@              </div>
96@          </div>
97@          <isset name="disabledText" value="${pdict.available ? '' : ' disabled="disabled' }" scope="page"/>
98@          <button type="submit" value="${Resource.msg('global.addtocart','locale',null)}" class="button-fancy-med">
99@              ${Resource.msg('global.addtocart','locale',null)}
100@          </button>
101@      </form>
102@  </div>
103@</isif>
104@
```

Modify producttopcontentPS.isml

Locate `/app_storefront_core/cartridge/templates/default/product/producttopcontentPS.isml` template.
Insert code below as shown on the screenshot.

```

<iscomment>
    product SubPro options
    =====
</iscomment>

<isif condition="${pdict.Product.custom.subproSubscriptionEnabled}">
    <h1>SUBPRO</h1>
    <div class="subscription">
        <div class="sp-add-to-order"></div>
        <isinclude url="${URLUtils.url('SubPro-PDP', 'sku', pdict.Product.ID)}" />
    </div>
</isif>
```

Here is the expected result:

```

168    <button id="add-all-to-cart" type="submit" value="${Resource.msg('global.addalltocart','Locale',null)}" c:
169        ${Resource.msg('global.addalltocart','locale',null)}
170    </button>
171    <iselse/>
172    <isscript>
173        var updateSources = ["cart", "giftregistry", "wishlist"];
174        var source = pdict.CurrentHttpParameterMap.source.stringValue;
175        var buttonTitle = (empty(source) || updateSources.indexOf(source)<0) ? dw.web.Resource.msg('global.a
176    </isscript>
177
178    <iscomment>
179        product SubPro options
180        =====
181    </iscomment>
182
183    <isif condition="#{pdict.Product.custom.subproSubscriptionEnabled}">
184        <h1>SUBPRO</h1>
185        <div class="subscription">
186            <div class="sp-add-to-order"></div>
187            <isinclude url="#{URLUtils.url('SubPro-PDP', 'sku', pdict.Product.ID)}" />
188        </div>
189    </isif>
190
191    <isset name="cartAction" value="add" scope="page"/>
192    <isif condition="#{pdict.CurrentHttpParameterMap.uuid.stringValue}">
193        <input type="hidden" name="uuid" id="uuid" value="#{pdict.CurrentHttpParameterMap.uuid.stringValue}" /
194        <isset name="cartAction" value="update" scope="page"/>
195    </isif>
196    <input type="hidden" name="cartAction" id="cartAction" value="#{cartAction}" />
197    <input type="hidden" name="pid" id="pid" value="#{pdict.Product.ID}" />
198    <button id="add-to-cart" type="submit" title="#{buttonTitle}" value="#{buttonTitle}" class="button-fancy-i
199    </isif>
200
201    </div><!-- END .addalltocart -->
202</form>
203

```

Modify pt_productdetails_UI.isml

Locate `/app_storefront_core/cartridge/templates/default/product/pt_productdetails_UI.isml` template. Add line below to load styles.

```
<link rel="stylesheet" href="#{URLUtils.staticURL('/css/subscribe.css')}" />
```

Here is the expected result:

```

pt_productdetails_UI.isml
1 <iscontent type="text/html" charset="UTF-8" compact="true"/>
2 <link rel="stylesheet" href="#{URLUtils.staticURL('/css/subscribe.css')}" />

```

Modify variant.js script

Locate `/app_storefront_core/cartridge/js/pages/product/variant.js` script. Add logic to handle displaying of subscription options. Include script from the int_subscribe_pro cartridge (*note this path may be different depending on the implementation*):

```
var subscriptionOptions = require('../../../../../LINK-subscribe-
pro/int_subscribe_pro/cartridge/js/subscriptionOptions');
```

and call function:

```
subscriptionOptions.variantInit();
```

(see picture below).

Here is the expected result:

```
variant.js
3 var ajax = require('../ajax'),
4     image = require('./image'),
5     progress = require('../progress'),
6     productStoreInventory = require('../storeinventory/product'),
7     tooltip = require('../tooltip'),
8     util = require('../util'),
9     subscriptionOptions = require(' ../../LINK-subscribe-pro/int_subscribe_pro/cartridge/js/subscriptionOptions');
10 /**
11  * @description update product content with new variant from href, load new content to #product-content panel
12  * @param {String} href - url of the new product variant
13 */
14 var updateContent = function (href) {
15     var $pdpForm = $('.pdpForm');
16     var qty = $pdpForm.find('input[name="Quantity"]').first().val();
17     var params = {};
18     progress.show($('#pdpMain'));
19     ajax.load({
20         url: util.appendParamsToUrl(href, params),
21         target: $('#product-content'),
22         callback: function () {
23             if (SitePreferences.STORE_PICKUP) {
24                 productStoreInventory.init();
25             }
26             image.replaceImages();
27             tooltip.init();
28             subscriptionOptions.variantInit();
29         }
30     });
31 };
32
33 module.exports = function () {
34     var $pdpMain = $('#pdpMain');
35     // hover on swatch - should update main image with swatch image
36     $pdpMain.on('mouseenter mouseleave', '.swatchanchor', function () {
37         var largeImg = $(this).data('lgimg'),
38             $imgZoom = $pdpMain.find('.main-image'),
39             $mainImage = $pdpMain.find('.primary-image');
40         if (!largeImg) { return; }
41         // store the old data from main image for mouseleave handler
42         $(this).data('lgimg', {
43             hires: $imgZoom.attr('href'),
44             url: $mainImage.attr('src'),
45             alt: $mainImage.attr('alt'),
46             title: $mainImage.attr('title')
47         });
48         // set the main image
49         image.setMainImage(largeImg);
50     });
51     // click on swatch - should replace product content with new variant
52     $pdpMain.on('click', '.product-detail .swatchanchor', function (e) {
53         e.preventDefault();
54         if ($(this).parents('li').hasClass('unselectable')) { return; }
55         updateContent(this.href);
56     });
57     // change drop down variation attribute - should replace product content with new variant
58     $pdpMain.on('change', '.variation-select', function () {
59         if ($(this).val().length === 0) { return; }
60         updateContent($(this).val());
61     });
62     subscriptionOptions.variantInit();
63 }
```

Modify GetProductDetailUrl.ds

Locate /app_storefront_core/cartridge/scripts/product/GetProductDetailUrl.ds script. Find GetProductDetailUrl function and add code below to append selected subscription option mode and

interval to URL.

```
if (pdict.CurrentHttpParameterMap.selectedOptionMode.stringValue) {  
    urlParams['selectedOptionMode'] =  
    pdict.CurrentHttpParameterMap.selectedOptionMode.stringValue;  
}  
  
if (pdict.CurrentHttpParameterMap.selectedInterval.stringValue) {  
    urlParams['selectedInterval'] =  
    pdict.CurrentHttpParameterMap.selectedInterval.stringValue;  
}
```

Here is the expected result:

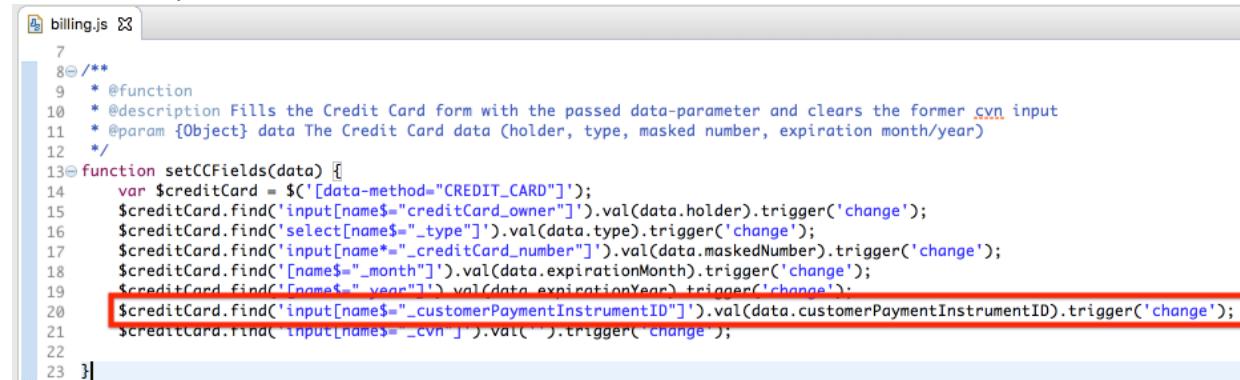
```
GetProductDetailUrl.ds [ ]  
120 /**  
121 * Create Product-Detail url for use in product/product.isml template  
122 *  
123 * @param pdict PipelineDictionary  
124 *  
125 **/  
126 function GetProductDetailUrl(pdict) {  
127     var urlArgs = ['Product-Detail'];  
128     var urlParams = {  
129         pid: pdict.Product.ID,  
130         source: pdict.CurrentHttpParameterMap.source.stringValue,  
131         uuid: pdict.CurrentHttpParameterMap.uuid.stringValue,  
132         Quantity: pdict.CurrentHttpParameterMap.Quantity.stringValue,  
133         productlistid: pdict.CurrentHttpParameterMap.productlistid.stringValue  
134     };  
135  
136     if (pdict.CurrentHttpParameterMap.selectedOptionMode.stringValue) {  
137         urlParams['selectedOptionMode'] = pdict.CurrentHttpParameterMap.selectedOptionMode.stringValue;  
138     }  
139  
140     if (pdict.CurrentHttpParameterMap.selectedInterval.stringValue) {  
141         urlParams['selectedInterval'] = pdict.CurrentHttpParameterMap.selectedInterval.stringValue;  
142     }  
143  
144     var variationAttributes = GetVariationAttributes(pdict);  
145     var productOptions = GetProductOptions(pdict.CurrentOptionModel);  
146  
147     // construct the urlArgs array with params names and values  
148     for (var param in urlParams) {  
149         urlArgs.push(param);  
150         urlArgs.push(urlParams[param]);  
151     }  
152     for (var attr in variationAttributes) {
```

Modify billing.js

Locate `/app_storefront_core/cartridge/js/pages/checkout/billing.js` script. Find `setCCFields` function and add code below to update the customer payment instrument input field with the selected UUID.

```
$creditCard.find('input[name$="_customerPaymentInstrumentID"]').val(data.customerPaymentInstrumentID).trigger('change');
```

Here is the expected result:



```
7
8 /**
9  * @function
10 * @description Fills the Credit Card form with the passed data-parameter and clears the former cvn input
11 * @param {Object} data The Credit Card data (holder, type, masked number, expiration month/year)
12 */
13 function setCCFields(data) {
14     var $creditCard = $('[data-method="CREDIT_CARD"]');
15     $creditCard.find('input[name$="creditCard_owner"]').val(data.holder).trigger('change');
16     $creditCard.find('select[name$="_type"]').val(data.type).trigger('change');
17     $creditCard.find('input[name$="_creditCard_number"]').val(data.maskedNumber).trigger('change');
18     $creditCard.find('input[name$="_month"]').val(data.expirationMonth).trigger('change');
19     $creditCard.find('input[name$="_year"]').val(data.expirationYear).trigger('change');
20     $creditCard.find('input[name$=\"_customerPaymentInstrumentID\"]').val(data.customerPaymentInstrumentID).trigger('change');
21     $creditCard.find('input[name$=\"_cvn\"]').val('').trigger('change');
22
23 }
```

Changes need to be done to display Subscribe Pro “Add to Order” widget.

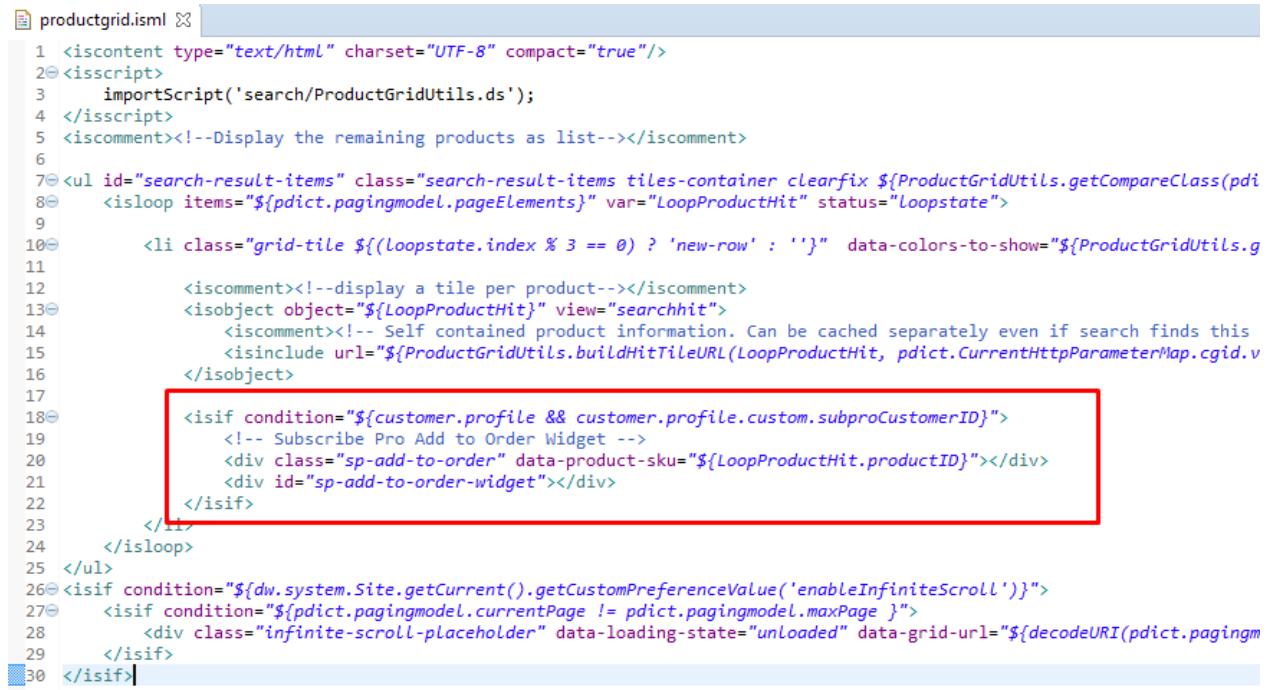
Modify productgrid.isml

Find template `/app_storefront_core/cartridge/templates/default/search/productgrid.isml`. Insert code below to add tag which the Subscribe Pro “Add to Order” widget will be rendered to.

```
<isif condition="${customer.profile && customer.profile.custom.subproCustomerID}">
    <!-- Subscribe Pro Add to Order Widget --&gt;
    &lt;div class="sp-add-to-order" data-product-sku="${LoopProductHit.productID}"&gt;&lt;/div&gt;
    &lt;div id="sp-add-to-order-widget"&gt;&lt;/div&gt;
&lt;/isif&gt;</pre>

```

Here is the expected result:



```
productgrid.isml
1  <iscontent type="text/html" charset="UTF-8" compact="true"/>
2  <isscript>
3      importScript('search/ProductGridUtils.ds');
4  </isscript>
5  <iscomment><!--Display the remaining products as list--></iscomment>
6
7  <ul id="search-result-items" class="search-result-items tiles-container clearfix ${ProductGridUtils.getCompareClass(pdi
8      <isloop items="${pdict.pagingmodel.pageElements}" var="LoopProductHit" status="loopstate">
9
10     <li class="grid-tile ${((loopstate.index % 3 == 0) ? 'new-row' : '')" data-colors-to-show="${ProductGridUtils.g
11
12         <iscomment><!--display a tile per product--></iscomment>
13         <isobject object="${LoopProductHit}" view="searchhit">
14             <iscomment><!-- Self contained product information. Can be cached separately even if search finds this
15             <isinclude url="${ProductGridUtils.buildHitTileURL(LoopProductHit, pdict.CurrentHttpParameterMap.cgiid.v
16         </isobject>
17
18         <isif condition="${customer.profile && customer.profile.custom.subproCustomerID}">
19             <!-- Subscribe Pro Add to Order Widget -->
20             <div class="sp-add-to-order" data-product-sku="${LoopProductHit.productID}"></div>
21             <div id="sp-add-to-order-widget"></div>
22         </isif>
23
24     </li>
25   </isloop>
26   <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('enableInfiniteScroll')}">
27       <isif condition="${pdict.pagingmodel.currentPage != pdict.pagingmodel.maxPage }">
28           <div class="infinite-scroll-placeholder" data-loading-state="unloaded" data-grid-url="${decodeURI(pdict.pagingm
29       </isif>
30   </isif>|
```

Modify pt_productsearchresult.isml

Find template `/app_storefront_core/cartridge/templates/default/search/pt_productsearchresult.isml`. Add code below at the end of body to render the Subscribe Pro “Add to Order” widget.

```
<isif condition="${customer.authenticated}">
    <iscomment>
        Configure the Subscribe Pro Add to Order Widget
    </iscomment>
    <!-- Load the Subscribe Pro widget script -->
```

```

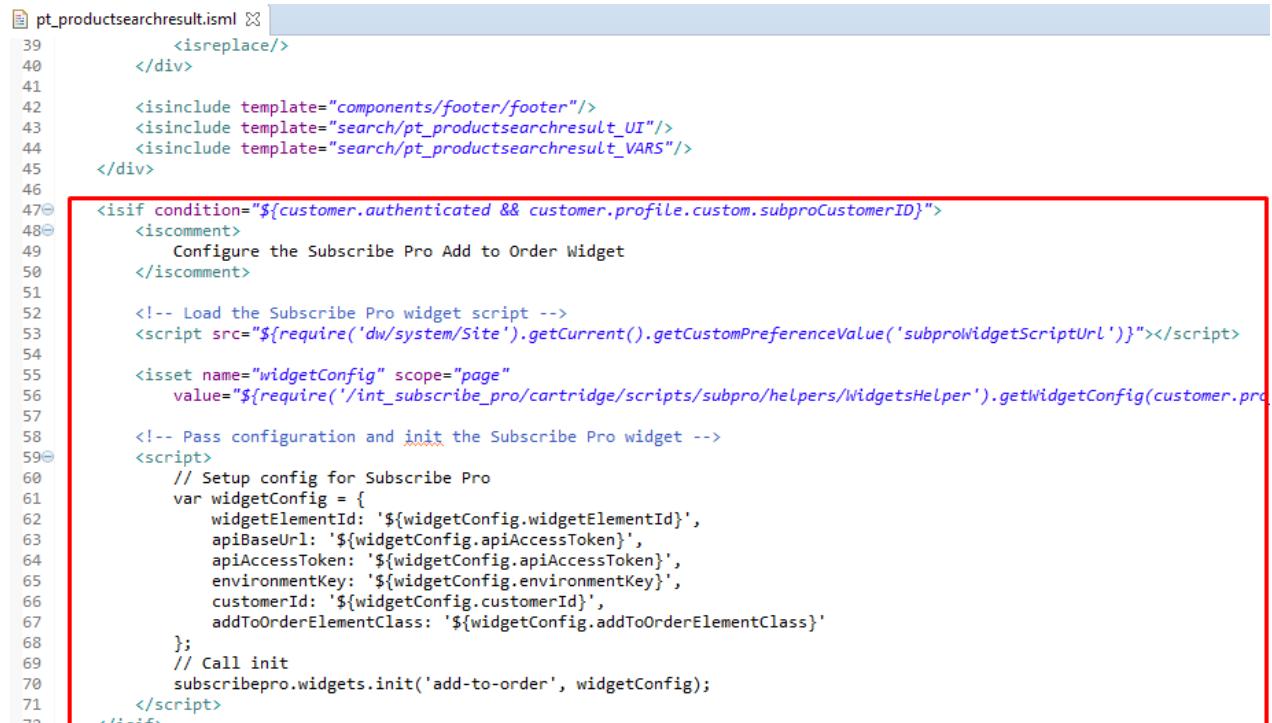
<script
src="${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subscribeProWidgetScriptUrl')}"></script>

<isset name="widgetConfig" scope="page"
value="${require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/WidgetsHelper').getWidgetConfig(customer.profile.custom.subprocustomerID, 'client_credentials', 'widget', 'sp-add-to-order-widget', 'sp-add-to-order')}">

<!-- Pass configuration and init the Subscribe Pro widget -->
<script>
    // Setup config for Subscribe Pro
    var widgetConfig = {
        widgetElementId: '${widgetConfig.widgetElementId}',
        apiBaseUrl: '${widgetConfig.apiAccessToken}',
        apiAccessToken: '${widgetConfig.apiAccessToken}',
        environmentKey: '${widgetConfig.environmentKey}',
        customerId: '${widgetConfig.customerId}',
        addToOrderElementClass: '${widgetConfig.addToOrderElementClass}'
    };
    // Call init
    subscribePro.widgets.init('add-to-order', widgetConfig);
</script>
</isif>

```

Here is the expected result:



```

39         <isreplace>
40             </div>
41
42             <isinclud template="components/footer/footer"/>
43             <isinclud template="search/pt_productsearchresult_UI"/>
44             <isinclud template="search/pt_productsearchresult_VARS"/>
45         </div>
46
47 @<isif condition="${customer.authenticated && customer.profile.custom.subprocustomerID}">
48 @    <iscomment>
49 @        Configure the Subscribe Pro Add to Order Widget
50 @    </iscomment>
51
52     <!-- Load the Subscribe Pro widget script -->
53     <script src="${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subprocWidgetScriptUrl')}"></script>
54
55     <isset name="widgetConfig" scope="page"
56         value="${require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/WidgetsHelper').getWidgetConfig(customer.pro
57
58     <!-- Pass configuration and init the Subscribe Pro widget -->
59     <script>
60         // Setup config for Subscribe Pro
61         var widgetConfig = {
62             widgetElementId: '${widgetConfig.widgetElementId}',
63             apiBaseUrl: '${widgetConfig.apiAccessToken}',
64             apiAccessToken: '${widgetConfig.apiAccessToken}',
65             environmentKey: '${widgetConfig.environmentKey}',
66             customerId: '${widgetConfig.customerId}',
67             addToOrderElementClass: '${widgetConfig.addToOrderElementClass}'
68         };
69         // Call init
70         subscribePro.widgets.init('add-to-order', widgetConfig);
71     </script>
72 </isif>
73 </body>
74 </html>
75

```

Basket (cart)

Changes need to be done to display subscription options for products in cart.

Modify cart.isml

Find template `/app_storefront_core/cartridge/templates/default/checkout/cart/cart.isml`. Add code below to include template with subscription options if product line item has SubPro subscriptions.

```
<isdisplayliproduct p_productli="${lineItem}" p_formli="${FormLi}" p_editable="${true}"  
p_hideprice="${true}" p_hidepromo="${false}" />  
<isif condition="${lineItem.custom.subproSubscriptionOptionMode}">  
    <div class="subscription">  
        <isinclude url="${URLUtils.url('SubPro-Cart', 'pliUUID',  
productLineItem.getUUID())}" />  
    </div>  
</isif>
```

Insert it after each `<isdisplayliproduct>` tag (there should be two such places).

Here is the expected result:

```
cart.isml (top)  
169         <iselse/>  
170               
172         <td class="item-details">  
173             <iscomment>Call module to render product</iscomment>  
174             <isdisplayliproduct p_productli="${lineItem}" p_formli="${FormLi}" p_editable="${true}" p_hideprice="${true}" p_  
175                 <isif condition="${lineItem.custom.subproSubscriptionOptionMode}">  
176                     <div class="subscription">  
177                         <isinclude url="${URLUtils.url('SubPro-Cart', 'pliUUID', productLineItem.getUUID())}" />  
178                     </div>  
179                 </isif>  
180             </td>  
181             <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('enableStorePickUp')}">  
182                 <td class="item-delivery-options">  
183                     <isinclude template="checkout/cart/storepickup/deliveryoptions" />  
184                 </td>  
185             </isif>  
186             ...  
187         </td>  
188     </tr>  
189     ...  
190  
cart.isml (bottom)  
634           
636         <td class="item-details">  
637             <iscomment>Call module to render product</iscomment>  
638             <isdisplayliproduct p_productli="${lineItem}" p_formli="${FormLi}" p_editable="${false}" p_hideprice="${true}"  
639                 <isif condition="${lineItem.custom.subproSubscriptionOptionMode}">  
640                     <div class="subscription">  
641                         <isinclude url="${URLUtils.url('SubPro-Cart', 'pliUUID', productLineItem.getUUID())}" />  
642                     </div>  
643                 </isif>  
644             <div class="bonusproducts">  
645                 <a href="${URLUtils.url('Product-GetBonusProducts', 'bonusDiscountLineItemUUID', bonusDiscountLineItem.UUID, bonusButtonText)}"  
646                     <${bonusButtonText}>  
647                 </a>  
648             </div>  
649         </td>  
650         <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('enableStorePickUp')}">  
651             <td class="item-delivery-options">  
652                 <isinclude template="checkout/cart/storepickup/deliveryoptions" />  
653             </td>  
654         </isif>  
655     </tr>  
656     ...
```

Modify displayliproduct.isml

Locate `/app_storefront_core/cartridge/templates/default/product/components/displayliproduct.isml` template. Find `<isscript>` where URL is built. Add following lines to urlParams object to save subscription selected option mode and interval.

```
selectedOptionMode: productLineItem.custom.subproSubscriptionSelectedOptionMode,  
selectedInterval: productLineItem.custom.subproSubscriptionInterval
```

Here is the expected result:

```
displayliproduct.isml ✘  
118@ <iscomment>  
119      Show Edit Details link if  
120      Product is not null and it is either a variant and online or options product and online  
121  </iscomment>  
122@ <isif condition="${!productLineItem.bonusProductLineItem && productLineItem.product != null && ((pdict.p_editable &&  
123@   <div class="item-edit-details">  
124@     <isif condition="${empty(pdict.p_editable) || pdict.p_editable}">  
125@       <isscript>  
126         var liUrl = '';  
127         var urlArgs = ['Product>Show'];  
128         var urlParams = {  
129           pid: productLineItem.productID,  
130           Quantity: productLineItem.quantity.value.toFixed(),  
131           uuid: productLineItem.UUID,  
132           source: 'cart',  
133           selectedOptionMode: productLineItem.custom.subproSubscriptionSelectedOptionMode,  
134           selectedInterval: productLineItem.custom.subproSubscriptionInterval  
135         };  
136  
137         // if item has a category context, forward this category context  
138         if (productLineItem.categoryID != null) {  
139           urlParams.cgid = productLineItem.categoryID;  
140         }  
141  
142         if (productLineItem.optionProductLineItems.size() > 0) {  
143           var pom : dw.catalog.ProductOptionModel = productLineItem.optionModel;  
144           var it : dw.util.Iterator = productLineItem.optionProductLineItems.iterator();  
145  
146           while (it.hasNext()) {  
147             var oli : dw.order.ProductLineItem = it.next();  
148             var opt : dw.catalog.ProductOption = pom.getOption(oli.optionID);  
149             urlParams[opt.htmlName] = pom.getOptionValue(opt, oli.optionValueID).ID;  
150           }  
151         }  
      }
```

Modify Resource.ds

Find `/app_storefront_core/cartridge/scripts/util/Resource.ds`. Find method `getUrls()`, variable `urls` and insert code below to add url to SubPro-UpdateOptions controller node which will update Subscription Options on Product Line Item in cart.

```
subproSubscriptionOptions : URLUtils.url('SubPro-UpdateOptions').toString()
```

Here is the expected result:

```

8 *Resource.ds ✘
135 resources["REMAIN_" + ProductAvailabilityModel.AVAILABILITY_STATUS_BACKORDER] = Resource.msg('global.remainingbackorder', 'lo
136 resources[ProductAvailabilityModel.AVAILABILITY_STATUS_NOT_AVAILABLE] = Resource.msg('global.allnotavailable', 'locale', null
137 resources["REMAIN_" + ProductAvailabilityModel.AVAILABILITY_STATUS_NOT_AVAILABLE] = Resource.msg('global.remainingnotavailable
138
139     return resources;
140 }
141 /**
142 * Get the client-side URLs of a given page
143 * @returns {Object} An objects key key-value pairs holding the URLs
144 */
145
146 ResourceHelper.getUrls = function(pageContext) {
147     var URLUtils = require('dw/web/URLUtils');
148     var Resource = require('dw/web/Resource');
149
150     // application urls
151     var urls = {
152         subproSubscriptionOptions : URLUtils.url('SubPro-UpdateOptions').toString(),
153         appResources : URLUtils.url('Resources-Load').toString(),
154         pageInclude : URLUtils.url('Page-Include').toString(),
155         continueUrl : request.isHttpSecure() ? URLUtils.httpsContinue().toString() : URLUtils.httpContinue().toString(),
156         staticPath : URLUtils.staticURL('/').toString(),
157         addGiftCert : URLUtils.url('GiftCert-Purchase').toString(),
158         minicartGC : URLUtils.url('GiftCert-ShowMiniCart').toString(),
159         addProduct : URLUtils.url('Cart-AddProduct').toString(),
160         minicart : URLUtils.url('Cart-MiniAddProduct').toString(),
161         cartShow : URLUtils.url('Cart-Show').toString(),
162         giftRegAdd : URLUtils.https('Address-GetAddressDetails', 'addressID', '').toString(),

```

Modify cart.js

Find script `/app_storefront_core/cartridge/js/pages/cart.js`. Add logic to handle displaying and modifying of the subscription options. Include script from the `int_subscribe_pro` cartridge (*Note: this path may change depending on implementation*):

```

var subscriptionOptions = require('.../.../.../.../LINK-subscribe-
pro/int_subscribe_pro/cartridge/js/subscriptionOptions');
```

and at the end of `initializeEvents()` function call function `cartInit()`:

```

subscriptionOptions.cartInit();
```

Here is the expected result:

```

cart.js ✘
1 'use strict';
2
3 var account = require('../account'),
4     bonusProductsView = require('../bonus-products-view'),
5     quickview = require('../quickview'),
6     cartStoreInventory = require('../cartstoreinventory'),
7     subscriptionOptions = require('.../.../.../LINK-subscribe-pro/int_subscribe_pro/cartridge/js/subscriptionOptions');
8
9 /**
10  * @private
11  * @function
12  * @description Binds events to the cart page (edit item's details, bonus item's actions, coupon code entry)
13 */
14 function initializeEvents() {
15     $('#cart-table').on('click', '.item-edit-details a', function (e) {
16         e.preventDefault();
17         quickview.show({
18             url: e.target.href,
19             source: 'cart'
20         });
21     })
22     .on('click', '.bonus-item-actions a, .item-details .bonusproducts a', function (e) {
23         e.preventDefault();
24         bonusProductsView.show(this.href);
25     });
26
27     // override enter key for coupon code entry
28     $('form input[name$="_couponCode"]').on('keydown', function (e) {
29         if (e.which === 13 && $(this).val().length === 0) {
30             return false;
31         }
32     });
33
34     //to prevent multiple submissions of the form when removing a product from the cart
35     var removeItemEvent = false;
36     $('button[name$="deleteProduct"]').on('click', function (e) {
37         if (removeItemEvent) {
38             e.preventDefault();
39         } else {
40             removeItemEvent = true;
41         }
42     });
43
44     subscriptionOptions.cartInit();
45 }
46
47 exports.init = function () {
    ...
}

```

Modify CartModel.js

Find script model /app_storefront_controllers/cartridge/scripts/models/CartModel.js.

Add logic to get product subscription options from template and then save them to Product Line Item custom attributes. Data that will be stored in Product Line Item:

- subproSubscriptionAvailableOptionMode
- subproSubscriptionOptionMode
- subproSubscriptionInterval
- subproSubscriptionAvailableIntervals
- subproSubscriptionDiscount
- subproSubscriptionIsDiscountPercentage
- subproContainsSubscriptions
- subproSubscriptionsToBeProcessed

1) Find method *addProductToCart*. Find line:

```
cart.updateLineItem(lineItem, productToAdd, quantity, productOptionModel);
```

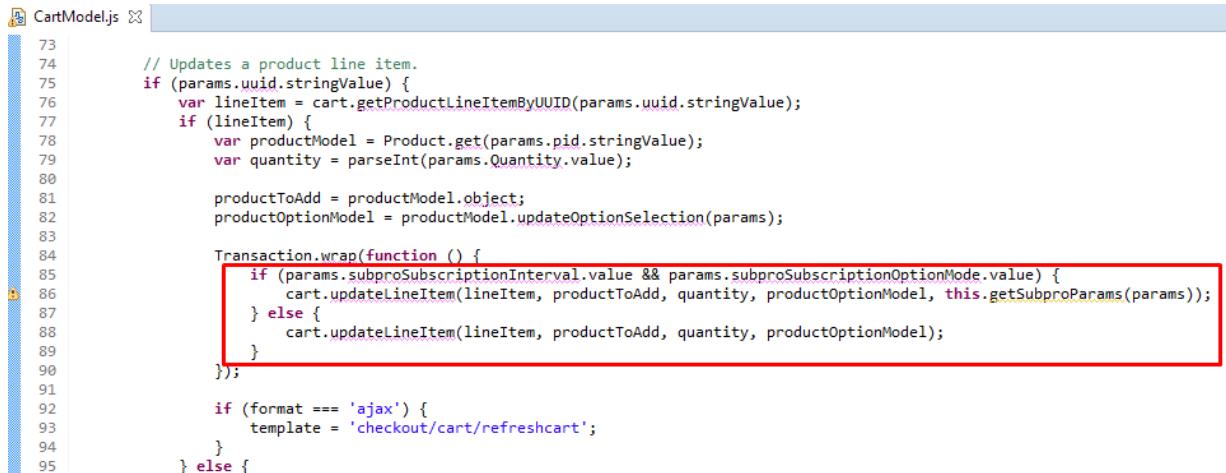
and replace with the code below:

```

if (params.subproSubscriptionInterval.value && params.subproSubscriptionOptionMode.value) {
    cart.updateLineItem(lineItem, productToAdd, quantity, productOptionModel,
    cart.getSubproParams(params));
} else {
    cart.updateLineItem(lineItem, productToAdd, quantity, productOptionModel);
}

```

Here is the expected result:



```

CartModel.js ✘

73
74     // Updates a product line item.
75     if (params.uuid.stringValue) {
76         var lineItem = cart.getProductLineItemByUUID(params.uuid.stringValue);
77         if (lineItem) {
78             var productModel = Product.get(params.pid.stringValue);
79             var quantity = parseInt(params.Quantity.value);

80             productToAdd = productModel.object;
81             productOptionModel = productModel.updateOptionSelection(params);

82             Transaction.wrap(function () {
83                 if (params.subproSubscriptionInterval.value && params.subproSubscriptionOptionMode.value) {
84                     cart.updateLineItem(lineItem, productToAdd, quantity, productOptionModel, this.getSubproParams(params));
85                 } else {
86                     cart.updateLineItem(lineItem, productToAdd, quantity, productOptionModel);
87                 }
88             });
89
90             if (format === 'ajax') {
91                 template = 'checkout/cart/refreshcart';
92             }
93         } else {
94
95

```

2) Find method `addProductToCart`. Find line:

```
cart.addProductItem(productToAdd.object, params.Quantity.doubleValue, productOptionModel);
```

and replace with the code below:

```

if (params.subproSubscriptionInterval.value && params.subproSubscriptionOptionMode.value) {
    cart.addProductItem(productToAdd.object, params.Quantity.doubleValue, productOptionModel,
    cart.getSubproParams(params));
} else {
    cart.addProductItem(productToAdd.object, params.Quantity.doubleValue, productOptionModel);
}

```

Here is the expected result:

```

CartModel.js ✘
109 var previousBonusDiscountLineItems = cart.getBonusDiscountLineItems();
110 productToAdd = Product.get(params.pid.stringValue);
111
112 if (productToAdd.object.isProductSet()) {
113     var childPids = params.childPids.stringValue.split(',');
114     var childQtys = params.childQtys.stringValue.split(',');
115     var counter = 0;
116
117     for (var i = 0; i < childPids.length; i++) {
118         var childProduct = Product.get(childPids[i]);
119
120         if (childProduct.object && !childProduct.isProductSet()) {
121             var childProductOptionModel = childProduct.updateOptionSelection(params);
122             cart.addProductItem(childProduct.object, parseInt(childQtys[counter]), childProductOptionModel);
123         }
124         counter++;
125     }
126 } else {
127     productOptionModel = productToAdd.updateOptionSelection(params);
128
129     if (params.subproSubscriptionInterval.value && params.subproSubscriptionOptionMode.value) {
130         cart.addProductItem(productToAdd.object, params.Quantity.doubleValue, productOptionModel, this.getSubproParams(params));
131     } else {
132         cart.addProductItem(productToAdd.object, params.Quantity.doubleValue, productOptionModel);
133     }
134 }
135
136 // When adding a new product to the cart, check to see if it has triggered a new bonus discount line item.
137 newBonusDiscountLineItem = cart.getNewBonusDiscountLineItem(previousBonusDiscountLineItems);
138 }
139
140 return {
141     format: format,
142     template: template,
143     BonusDiscountLineItem: newBonusDiscountLineItem
144 };

```

- 3) Find method `addProductItem`. Add `subproParams` to function parameters.

Add `subproParams` parameter to method (see image below).

Find line:

```
var productLineItem = cart.createProductLineItem(product, productOptionModel, shipment);
```

Add code below after it to update Product Line Item with Subscribe Pro parameters:

```
if (subproParams) {
    cart.addSubproParamsToPLI(productLineItem, subproParams);
}
```

Here is the expected result:

```

CartModel.js ②

181 * @param {dw.catalog.ProductOptionModel} productOptionModel - The option model of the product that is to be added to the basket.
182 * @param {Object} [subproParams] - Object containing options for SubPro subscription
183
184 addProductItem: function (product, quantity, productOptionModel, subproParams) {
185     var cart = this;
186     Transaction.wrap(function () {
187         var i;
188         if (product) {
189             var productInCart;
190             var productListItems = cart.object.productLineItems;
191             var quantityInCart;
192             var quantityToSet;
193             var shipment = cart.object.defaultShipment;
194
195             for (var q = 0; q < cart.object.productLineItems.length; q++) {
196                 if (productListItems[q].productID === product.ID) {
197                     productInCart = productListItems[q];
198                     break;
199                 }
200             }
201
202             if (productInCart) {
203                 quantityInCart = productInCart.getQuantity();
204                 quantityToSet = quantity ? quantity + quantityInCart : quantityInCart + 1;
205                 productInCart.setQuantityValue(quantityToSet);
206             } else {
207                 var productLineItem = cart.createProductLineItem(product, productOptionModel, shipment);
208
209                 if (subproParams) {
210                     cart.addSubproParamsToPLI(productLineItem, subproParams);
211                 }
212
213                 if (quantity) {
214                     productLineItem.setQuantityValue(quantity);
215                 }
216             }
217         }

```

4) Find method *updateLineItem*. Add *subproParams* to function parameters.

Add *subproParams* parameter to method (see image below).

Add code below after it to update Product Line Item with Subscribe Pro parameters:

```

if (subproParams) {
    this.addSubproParamsToPLI(lineItem, subproParams);
}

```

Here is the expected result:

```

CartModel.js ✘

505 * pids of the bundled products that are variations and any options.
506 *
507 * @transactional
508 * @alias module:models/CartModel~CartModel/updateLineItem
509 * @param {dw.order.ProductLineItem} lineItem - The product line item to replace.
510 * @param {dw.catalog.Product} product - The new product.
511 * @param {Number} quantity - The quantity of the product line item after replacement.
512 * @param {dw.catalog.ProductOptionModel} productOptionModel - The option model of the product to add to the basket.
513 * @param {Object} [subproParams] - Object containing options for SubPro subscription
514 */
515 updateLineItem: function (lineItem, product, quantity, productOptionModel, subproParams) {
516     var optionValues = productOptionModel.getOptions();
517
518     lineItem.replaceProduct(product);
519     lineItem.setQuantityValue(quantity);
520
521     if (optionValues.length) {
522         lineItem.updateOptionValue(optionValues[0]);
523     }
524
525     if (subproParams) {
526         this.addSubproParamsToPLI(lineItem, subproParams);
527     }
528
529     if (product.isBundle()) {
530
531         if (request.httpParameterMap.childPids.stringValue) {
532             var childPids = request.httpParameterMap.childPids.stringValue.split(',');
533
534             for (var i = 0; i < childPids.length; i++) {
535                 var childProduct = Product.get(childPids[i]).object;
536
537                 if (childProduct) {
538                     // why is this needed ?
539                     childProduct.updateOptionSelection(request.httpParameterMap);
540
541             }
542         }
543     }
544 }

```

5) Find method *createOrder*.

Add code below to update basket with Subscribe Pro properties.

```

let isSubPro =
require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib.js').isSubPro();
if (isSubPro) {
    Transaction.wrap(function () {
        basket.custom.subproContainsSubscriptions = isSubPro;
        basket.custom.subproSubscriptionsToBeProcessed = true;
    });
}

```

Here is the expected result:

```

CartModel.js ✘
1420     * registry, the purchase history of the respective gift registry item is updated.
1421     *
1422     * @transactional
1423     * @alias module:models/CartModel~CartModel/createOrder
1424     * @returns {dw.order.Order} The created order in status CREATED or null if an error occurred.
1425     */
1426     createOrder: function () {
1427         var basket = this.object;
1428         var order;
1429         try {
1430             let isSubPro = require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib.js').isSubPro();
1431             if (isSubPro) {
1432                 Transaction.wrap(function () {
1433                     basket.custom.subproContainsSubscriptions = isSubPro;
1434                     basket.custom.subproSubscriptionsToBeProcessed = true;
1435                 });
1436             }
1437             order = Transaction.wrap(function () {
1438                 return OrderMgr.createOrder(basket);
1439             });
1440         } catch (error) {
1441             return;
1442         }
1443         return order;
1444     },
1445     1446 },
1447

```

6) Finally, add methods to CartModel object:

```

/**
 * Get Subscribe Pro subscription parameters and return them as an object.
 *
 * @param {object} params
 * @returns {{subscriptionInterval, subscriptionOptionMode, subscriptionSelectedOptionMode,
 * subscriptionAvailableIntervals, subscriptionDiscount, subscriptionIsDiscountPercentage:
 * boolean}}
 */
getSubproParams: function (params) {
    return {
        "subscriptionInterval": params.subproSubscriptionInterval.value,
        "subscriptionOptionMode": params.subproSubscriptionAvailableOptionMode.value,
        "subscriptionSelectedOptionMode": params.subproSubscriptionOptionMode.value,
        "subscriptionAvailableIntervals":
            params.subproSubscriptionAvailableIntervals.value,
        "subscriptionDiscount": params.subproSubscriptionDiscount.value,
        "subscriptionIsDiscountPercentage":
            params.subproSubscriptionIsDiscountPercentage.booleanValue
    }
},
/***
 * Update Product Line Item with Subscribe Pro subscription parameters
 *
 * @param {dw.order.ProductLineItem} pli The product line item to update
 * @param {object} subproParams Object containing Subscribe Pro subscription parameters
 * which should be added to product line item
 */
addSubproParamsToPLI: function (pli, subproParams) {
    var discountValue = parseFloat(subproParams.subscriptionDiscount),
        discountToApply = subproParams.subscriptionIsDiscountPercentage
            ? new dw.campaign.PercentageDiscount(discountValue * 100)
            : new dw.campaign.AmountDiscount(discountValue);

    if (subproParams.subscriptionSelectedOptionMode === 'regular') {
        pli.createPriceAdjustment("SubscribeProDiscount", discountToApply);
    }
}

```

```

    pli.custom.subproSubscriptionOptionMode = subproParams.subscriptionOptionMode;
    pli.custom.subproSubscriptionSelectedOptionMode =
subproParams.subscriptionSelectedOptionMode;
    pli.custom.subproSubscriptionInterval = subproParams.subscriptionInterval;
    pli.custom.subproSubscriptionAvailableIntervals =
subproParams.subscriptionAvailableIntervals;
    pli.custom.subproSubscriptionDiscount = discountValue;
    pli.custom.subproSubscriptionIsDiscountPercentage =
subproParams.subscriptionIsDiscountPercentage;
}

```

Here is the expected result:

```

CartModel.js ②
1479     return null;
1480 },
1481
1482 /**
1483 * Get Subscribe Pro subscription parameters and return them as an object.
1484 *
1485 * @param {object} params
1486 * @returns {{subscriptionInterval, subscriptionOptionMode, subscriptionSelectedOptionMode, subscriptionAvailableIntervals, subscr
1487 */
1488 getSubproParams: function (params) {
1489     return {
1490         "subscriptionInterval": params.subproSubscriptionInterval.value,
1491         "subscriptionOptionMode": params.subproSubscriptionAvailableOptionMode.value,
1492         "subscriptionSelectedOptionMode": params.subproSubscriptionOptionMode.value,
1493         "subscriptionAvailableIntervals": params.subproSubscriptionAvailableIntervals.value,
1494         "subscriptionDiscount": params.subproSubscriptionDiscount.value,
1495         "subscriptionIsDiscountPercentage": params.subproSubscriptionIsDiscountPercentage.booleanValue
1496     };
1497 },
1498
1499 /**
1500 * Update Product Line Item with Subscribe Pro subscription parameters
1501 *
1502 * @param {dw.order.ProductLineItem} pli The product line item to update
1503 * @param {object} subproParams Object containing Subscribe Pro subscription parameters which should be added to product line item
1504 */
1505 addSubproParamsToPLI: function (pli, subproParams) {
1506     var discountValue = parseFloat(subproParams.subscriptionDiscount),
1507         discountToApply = subproParams.subscriptionIsDiscountPercentage
1508             ? new dw.campaign.PercentageDiscount(discountValue * 100)
1509             : new dw.campaign.AmountDiscount(discountValue);
1510
1511     if (subproParams.subscriptionSelectedOptionMode === 'regular') {
1512         pli.createPriceAdjustment("SubscribeProDiscount", discountToApply);
1513     }
1514     pli.custom.subproSubscriptionOptionMode = subproParams.subscriptionOptionMode;
1515     pli.custom.subproSubscriptionSelectedOptionMode = subproParams.subscriptionSelectedOptionMode;
1516     pli.custom.subproSubscriptionInterval = subproParams.subscriptionInterval;
1517     pli.custom.subproSubscriptionAvailableIntervals = subproParams.subscriptionAvailableIntervals;
1518     pli.custom.subproSubscriptionDiscount = discountValue;
1519     pli.custom.subproSubscriptionIsDiscountPercentage = subproParams.subscriptionIsDiscountPercentage;
1520
1521 });
1522 });

```

Checkout

Changes need to be done to checkout process with subscription products.

Modify paymentmethods.isml

Locate template

/app_storefront_core/cartridge/templates/default/checkout/billing/paymentmethods.isml. Add code below to force show form with credit card payment methods.

- 1) Set isSubPro variable which will indicate if cart has items with subscription:

```
<isscript>
var SubscribeProLib =
require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib');
</isscript>

<isset name="isSubPro" value="${SubscribeProLib.isSubPro()}" scope="page" />
```

- 2) Change following condition to:

```
<isif condition="${pdict.OrderTotal > 0 || isSubPro}">
```

- 3) Add code below to render message, informing customer that credit card is required.

```
<isif condition="${pdict.OrderTotal <= 0 && isSubPro}"></fieldset>
    <div class="subpro-
message">${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subproCreditCar
dRequirementMsg')}</div>
</isif>
```

- 4) Add to force show form with credit card payment methods.

```
<isif condition="${pdict.CurrentCustomer.authenticated}">
    <isif condition="${isSubPro}">
        <isset name="saveInputField"
value="${pdict.CurrentForms.billing.paymentMethods.creditCard.saveCard}" scope="page" />

        <div class="form-row label-inline form-indent">
            <div class="field-wrapper">
                <input class="input-checkbox" type="checkbox" disabled="disabled"
checked="checked" value="true">
                <input type="hidden" id="${saveInputField.htmlName}"
name="${saveInputField.htmlName}" value="true">
            </div>
            <label for="${saveInputField.htmlName}">
                <span><isprint value="${Resource.msg(saveInputField.label, 'forms', null)}">
/></span>
            </label>
            <div class="form-caption"><isprint
value="${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subproSavedCredit
Msg')}" /></div>
        </div>
```

```
<iselse>
    <isinputfield
formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.saveCard}"
type="checkbox" />
    </isif>
</isif>
```

5) Add hidden form element to track the selected Customer Payment Instrument.

```
<isinputfield  
formfield="${pdict.CurrentForms.billing.paymentMethods.creditCard.customerPaymentInstrument  
ID}" type="hidden" rowClass="visually-hidden" />
```

Here is the expected result:

Modify checkoutlogin.isml

Locate template `/app_storefront_core/cartridge/templates/default/checkout/checkoutlogin.isml`.

- 1) Add code below to display message to customer.

```
<isif condition="${pdict.isSubpro}">
    <div class="subpro-
message">${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subproCheckoutL
oginMsg')}</div>
</isif>
```

- 2) Find condition

```
<isif condition="${!pdict.CurrentCustomer.registered}">
```

and replace it with code below to hide 'Checkout as Guest' option if cart has items with subscription.

```
<isif condition="${!pdict.CurrentCustomer.registered && !pdict.isSubpro}">
```

Here is the expected result:



```
checkoutlogin.isml
1 <iscontent type="text/html" charset="UTF-8" compact="true"/>
2 <isdecorate template="checkout/cart/pt_cart">
3 <isinclud template="util/modules"/>
4
5 <iscomment>Report this checkout step</iscomment>
6
7 <isreportcheckout checkoutstep="${1}" checkoutname="${'CheckoutMethod'}"/>
8 <div class="checkoutlogin">
9
10 <isif condition="${pdict.isSubpro}">
11     <div class="subpro-message">${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subproCheckoutLoginMsg')}</div>
12 </isif>
13 <div class="col-1">
14
15     <div class="login-box">
16
17         <h2>${Resource.msg('checkoutlogin.guestheader','checkout',null)}</h2>
18
19         <div class="login-box-content clearfix">
20             <isif condition="${!pdict.CurrentCustomer.registered && !pdict.isSubpro}">
21                 <p>${Resource.msg('checkoutlogin.guestmessage','checkout',null)}</p>
22                 <form action="${URLUtils_httpsContinue()}" method="post">
23                     <fieldset>
24                         <div class="form-row formbuttonrow">
25                             <button type="submit" value="${Resource.msg('checkoutlogin.checkoutguestbutton','checkout',null)}" name="${
26                             </div>
27                             <input type="hidden" name="${dw.web.CSRFProtection.getTokenName()}" value="${dw.web.CSRFProtection.generateToker
28                         </fieldset>
29                     </form>
30             </isif>
31             <iscomment>new customer</iscomment>
32             <p>${Resource.msg('globalaccount.createmessage','locale',null)}</p>
33             <form action="${URLUtils_httpsContinue()}" method="post">
```

Modify minilineitems.isml

Locate template

`/app_storefront_core/cartridge/templates/default/checkout/components/minilineitems.isml`. Add logic to show selected subscription options on mini cart and Order Summary Right Rail.

Add code block below after `<div class="mini-cart-pricing">` tag.

```
<isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
```

```

<div class="mini-cart-subscription-options">
    <isinclude url="\${URLUtils.url('SubPro-OrderSummary', 'pliUUID',
productLineItem.getUUID())}" />
</div>
</isif>

```

Here is the expected result:

```

95
96     <isdisplayproductavailability p_productli="\${productLineItem}" p_displayinstock="\${false}" p_displaypreorder="\${true}"
97
98     <div class="mini-cart-pricing">
99
100        <span class="Label">\${Resource.msg('global.qty','locale',null)}:</span>
101        <span class="value"><isprint value="\${productLineItem.quantity}" /></span>
102
103        <isif condition="\${productLineItem.bonusProductLineItem}">
104            <isset name="bonusProductPrice" value="\${productLineItem.getAdjustedPrice()}" scope="page"/>
105            <isinclude template="checkout/components/displaybonusproductprice" />
106            <isprint value="\${bonusProductPriceValue}" />
107        </iselse>
108        <isset name="productTotal" value="\${productLineItem.adjustedPrice}" scope="page"/>
109        <isif condition="\${productLineItem.optionProductLineItems.size() > 0}">
110            <isloop items="\${productLineItem.optionProductLineItems}" var="optionLI">
111                <isset name="productTotal" value="\${productTotal.add(optionLI.adjustedPrice)}" scope="page"/>
112            </isloop>
113        </isif>
114        <span class="mini-cart-price"><isprint value="\${productTotal}" /></span>
115    </isif>
116
117 </div>
118
119     <isif condition="\${productLineItem.custom.subproSubscriptionOptionMode}">
120         <div class="mini-cart-subscription-options">
121             <isinclude url="\${URLUtils.url('SubPro-OrderSummary', 'pliUUID', productLineItem.getUUID())}" />
122         </div>
123     </isif>
124
125 </div>
126
127 <isset name="itemCount" value="\${itemCount+1}" scope="page"/>
128
129 </isloop>

```

Modify multishippingshipments.isml

Locate template

/app_storefront_core/cartridge/templates/default/checkout/shipping/multishipping/multishippingshipments.isml.

Add code below after `<isdisplayliproduct />` tag to show subscription options for multishipping.

```

<isif condition="\${productLineItem.custom.subproSubscriptionOptionMode}">
    <div class="subscription">
        <isinclude url="\${URLUtils.url('SubPro-Cart', 'pliUUID',
productLineItem.getUUID())}" />
    </div>
</isif>

```

Here is the expected result:

```

multishippingshipments.isml ✘
126      </isif>
127      </td>
128
129      <td class="item-details">
130          <iscollection>Display product line and product using module</iscollection>
131          <isdisplayliproduct p_productli="${productLineItem}" p_editable="${false}" />
132          <isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
133              <div class="subscription">
134                  <isinclude url="${URLUtils.url('SubPro-Cart', 'pliUUID', productLineItem.getUUID())}" />
135              </div>
136          </isif>
137      </td>
138

```

Modify summary.isml

Locate template `/app_storefront_core/cartridge/templates/default/checkout/summary/summary.isml`. Add code below after `<isdisplayliproduct />` tag to show selected subscription options on Order Review page.

```

<isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
    <div class="subscription">
        <isinclude url="${URLUtils.url('SubPro-OrderSummary', 'pliUUID',
productLineItem.getUUID())}" />
    </div>
</isif>

```

Here is the expected result:

```

summary.isml ✘
64      </div>
65      </isif>
66      </td>
67
68      <td class="item-details">
69          <iscollection>Display product line and product using module</iscollection>
70          <isdisplayliproduct p_productli="${productLineItem}" p_editable="${false}" />
71          <isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
72              <div class="subscription">
73                  <isinclude url="${URLUtils.url('SubPro-OrderSummary', 'pliUUID', productLineItem.getUUID())}" />
74              </div>
75          </isif>
76      </td>
77
78      <td class="item-quantity">
79          <ispint value="${productLineItem.quantity}" />
80      </td>
81

```

Modify orderdetails.isml

Locate template `/app_storefront_core/cartridge/templates/default/components/order/orderdetails.isml`. Add code below after `<isdisplayliproduct />` tag to show elected subscription options on Order Confirmation page.

```

<isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
    <div class="subscription">
        <isinclude url="${URLUtils.url('SubPro-OrderConfirmation', 'orderNo', Order.orderNo,
'productID', productLineItem.productID)}" />

```

```

    </div>
</isif>
```

Here is the expected result:

```

128     </div>
129   </div>
130   <div class="line-items">
131     <isloop items="${shipment.productLineItems}" var="productLineItem" status="pliloopstate">
132       <div class="line-item">
133         <div class="line-item-details">
134           <isif condition="${pliloopstate.first}">
135             <div class="label">${Resource.msg('global.item','locale',null)}</div>
136           </isif>
137           <iscomment>Display product line and product using module</iscomment>
138           <isdisplayliproduct p_productID="${productLineItem.productID}" p_editable="${false}" />
139           <isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
140             <div class="subscription">
141               <isinclude url="${URLUtils.url('SubPro-OrderConfirmation', 'orderNo', Order.orderNo, 'productID', productLineItem.productID)}" />
142             </div>
143           </isif>
144         </div>
145         <div class="line-item-quantity">
146           <isif condition="${pliloopstate.first}">
147             <div class="label">${Resource.msg('global.qty','locale',null)}</div>
148           </isif>
149           <isprint value="${productLineItem.quantity}" />
150         </div>
151       </div>
152     </isloop>
153   </div>
```

Modify orderdetailsemail.isml

Locate template

/app_storefront_core/cartridge/templates/default/components/order/orderdetailsemail.isml.

Add code below after `<isdisplayliproduct />` tag to show selected subscription options in Order Confirmation Email.

```

<isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
  <table>
    <tr>
      <td colspan="2">${Resource.msg('order.subscription.info', 'order', null)}</td>
    </tr>
    <tr>
      <td>${Resource.msg('order.subscription.delivery.mode', 'order', null)}</td>
      <td>${productLineItem.custom.subproSubscriptionSelectedOptionMode}</td>
    </tr>
    <tr>
      <td>${Resource.msg('order.subscription.delivery.interval', 'order', null)}</td>
      <td>${productLineItem.custom.subproSubscriptionInterval}</td>
    </tr>
  </table>
</isif>
```

Here is the expected result:

```

orderdetailsemail.isml
90      </isit>
91    </tr>
92  </thead>
93  <isloop items="${shipment.productLineItems}" var="productLineItem" status="pliloopstate">
94    <tr>
95
96    <td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top;">
97      <iscollection>Display product line and product using module</iscollection>
98      <isdisplayliproduct p_productli="${productLineItem}" p_editable="${false}" />
99      <isif condition="${productLineItem.custom.subproSubscriptionOptionMode}">
100        <table>
101          <tr>
102            <td colspan="2">${Resource.msg('order.subscription.info', 'order', null)}</td>
103          </tr>
104          <tr>
105            <td>${Resource.msg('order.subscription.delivery.mode', 'order', null)}</td>
106            <td>${productLineItem.custom.subproSubscriptionSelectedOptionMode}</td>
107          </tr>
108          <tr>
109            <td>${Resource.msg('order.subscription.delivery.interval', 'order', null)}</td>
110            <td>${productLineItem.custom.subproSubscriptionInterval}</td>
111          </tr>
112        </table>
113      </isif>
114    </td>
115
116    <td style="font-size:12px;font-family:arial;padding:20px 10px;vertical-align:top;">
117      <isprint value="${productLineItem.quantity}" />
118    </td>
119

```

Modify creditcardjson.isml

Locate template `/app_storefront_core/cartridge/templates/default/checkout/billing/creditcardjson.isml`
 Add a new attribute to the returned JSON object that contains the saved payment instrument UUID

```

var cc = {
  maskedNumber:pdict.SelectedCreditCard.maskedCreditCardNumber,
  holder:pdict.SelectedCreditCard.creditCardHolder,
  type:pdict.SelectedCreditCard.creditCardType,
  expirationMonth:pdict.SelectedCreditCard.creditCardExpirationMonth,
  expirationYear:pdict.SelectedCreditCard.creditCardExpirationYear,
  customerPaymentInstrumentID:pdict.SelectedCreditCard.UUID
}

```

Here is the expected result:

```

creditcardjson.isml
1 <iscontent type="application/json" charset="UTF-8" compact="true"/>
2 <isinclude template="util/jsonmodule"/>
3<!--<comment>
4     This template renders the attributes of a customer credit card payment instrument as JSON response.
5 </comment>
6<!--<script>
7     var cc = {
8         maskedNumber:pdict.SelectedCreditCard.maskedCreditCardNumber,
9         holder:pdict.SelectedCreditCard.creditCardHolder,
10        type:pdict.SelectedCreditCard.creditCardType,
11        expirationMonth:pdict.SelectedCreditCard.creditCardExpirationMonth,
12        expirationYear:pdict.SelectedCreditCard.creditCardExpirationYear,
13        customerPaymentInstrumentID:pdict.SelectedCreditCard.UUID
14    }
15    var json = JSON.stringify(cc);
16 </script>
17 <isprint value="${json}" encoding="off"/>
```

Modify COBilling.js

Locate `/app_storefront_controllers/cartridge/controllers/COBilling.js` controller. Replace the code within the `Transaction.wrap` block to add save the credit cards first six digits and to update the saved customer payment instrument versus replacing it. This is important so that payment instruments UUID will persist for future orders placed via OCAPI.

```

/** function saveCreditCard() */
Transaction.wrap(function () {
    var creditCardForm = app.getForm('billing').object.paymentMethods.creditCard;
    var customerSelectedCreditCard = creditCardForm.customerPaymentInstrumentID ?
creditCardForm.customerPaymentInstrumentID.value : false;

    if (customerSelectedCreditCard && customerSelectedCreditCard.length > 0) {
        for (i = 0; i < creditCards.length; i++) {
            var creditcard = creditCards[i];

            if (creditcard.UUID == customerSelectedCreditCard) {
                newCreditCard = creditcard;
            }
        }
    }

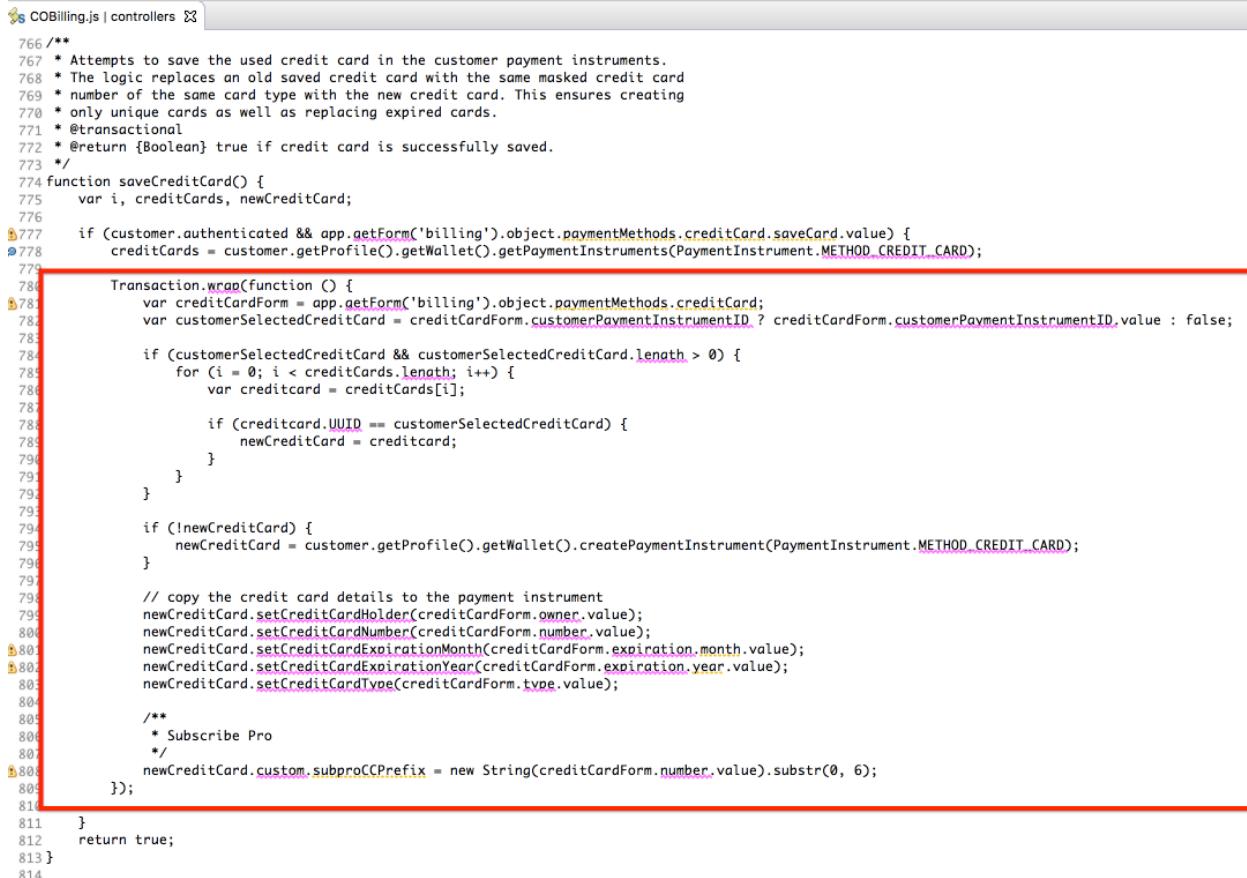
    if (!newCreditCard) {
        newCreditCard =
customer.getProfile().getWallet().createPaymentInstrument(PaymentInstrument.METHOD_CREDIT_CARD);
    }

    // copy the credit card details to the payment instrument
    newCreditCard.setCreditCardHolder(creditCardForm.owner.value);
    newCreditCard.setCreditCardNumber(creditCardForm.number.value);
    newCreditCard.setCreditCardExpirationMonth(creditCardForm.expiration.month.value);
    newCreditCard.setCreditCardExpirationYear(creditCardForm.expiration.year.value);
    newCreditCard.setCreditCardType(creditCardForm.type.value);

    /**
     * Subscribe Pro
     */
    newCreditCard.custom.subprocCPrefix = new String(creditCardForm.number.value).substr(0,
6);
```

```
});
```

Here is the expected result:



```
766 /**
767 * Attempts to save the used credit card in the customer payment instruments.
768 * The logic replaces an old saved credit card with the same masked credit card
769 * number of the same card type with the new credit card. This ensures creating
770 * only unique cards as well as replacing expired cards.
771 * @transactional
772 * @return {Boolean} true if credit card is successfully saved.
773 */
774 function saveCreditCard() {
775     var i, creditCards, newCreditCard;
776
777     if (customer.authenticated && app.getForm('billing').object.paymentMethods.creditCard.saveCard.value) {
778         creditCards = customer.getProfile().getWallet().getPaymentInstruments(PaymentInstrument.METHOD_CREDIT_CARD);
779
780         Transaction.wrap(function () {
781             var creditCardForm = app.getForm('billing').object.paymentMethods.creditCard;
782             var customerSelectedCreditCard = creditCardForm.customerPaymentInstrumentID ? creditCardForm.customerPaymentInstrumentID.value : false;
783
784             if (customerSelectedCreditCard && customerSelectedCreditCard.length > 0) {
785                 for (i = 0; i < creditCards.length; i++) {
786                     var creditcard = creditCards[i];
787
788                     if (creditcard.UUID == customerSelectedCreditCard) {
789                         newCreditCard = creditcard;
790                     }
791                 }
792             }
793
794             if (!newCreditCard) {
795                 newCreditCard = customer.getProfile().getWallet().createPaymentInstrument(PaymentInstrument.METHOD_CREDIT_CARD);
796             }
797
798             // copy the credit card details to the payment instrument
799             newCreditCard.setCreditCardHolder(creditCardForm.owner.value);
800             newCreditCard.setCreditCardNumber(creditCardForm.number.value);
801             newCreditCard.setCreditCardExpirationMonth(creditCardForm.expiration.month.value);
802             newCreditCard.setCreditCardExpirationYear(creditCardForm.expiration.year.value);
803             newCreditCard.setCreditCardType(creditCardForm.type.value);
804
805             /**
806             * Subscribe Pro
807             */
808             newCreditCard.custom.subproCCPrefix = new String(creditCardForm.number.value).substr(0, 6);
809         });
810     }
811     return true;
812 }
813 }
```

Modify COCustomer.js

Locate `/app_storefront_controllers/cartridge/controllers/COCustomer.js` controller. Add line below to send parameter `isSubpro`, which specify if cart has products with SubPro subscription, to pdict.

```
isSubpro: require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib.js').isSubPro()
```

Here is the expected result:

```

COCustomer.js ✘
25 * may not contain the form clearing, in order to support navigating forth and back in the checkout steps without losing
26 * already entered form values.
27 */
28@ function start() {
29     var oauthLoginForm = app.getForm('oauthlogin');
30     app.getForm('singleshipping').clear();
31     app.getForm('multishipping').clear();
32     app.getForm('billing').clear();
33
34     Transaction.wrap(function () {
35         Cart.goc().removeAllPaymentInstruments();
36     });
37
38 // Direct to first checkout step if already authenticated.
39 if (customer.authenticated) {
40     response.redirect(URLUtils.https('COShipping-Start'));
41     return;
42 } else {
43     var loginForm = app.getForm('login');
44     loginForm.clear();
45     oauthLoginForm.clear();
46
47 // Prepopulate login form field with customer's login name.
48 if (customer.registered) {
49     loginForm.setValue('username', customer.profile.credentials.login);
50 }
51
52 var loginAsset = Content.get('myaccount-login');
53
54 var pageMeta = require('~/cartridge/scripts/meta');
55 pageMeta.update(loginAsset);
56
57 app.getView({
58     ContinueURL: URLUtils.https('COCustomer-LoginForm').append('scope', 'checkout'),
59     isSubpro: require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib.js').isSubPro()
60 }).render('checkout/checkoutlogin');
61
62 }
63 }

```

Modify COSummary.js

Locate `/app_storefront_controllers/cartridge/controllers/COSummary.js` controller. Add logic to check if cart has any credit card as a payment method (it is required for subscription). If it doesn't we pass `isSubPro: true` parameter to `checkout/billing/billing` template, forcing it to show form with credit card payment methods.

Add logic from code block below to `start()` method

```

const SubscribeProLib =
require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib.js');
let isSubpro = SubscribeProLib.isSubPro(),
    hasCreditCard = SubscribeProLib.hasCreditCard(cart);

if (isSubpro && !hasCreditCard) {
    app.getView({
        Basket: cart.object,
        ContinueURL: URLUtils.https('COBilling-Billing'),
        isSubPro: true
    }).render('checkout/billing/billing');

    return;
}

```

Here is the expected result:

```

23 /**
24  * Renders the summary page prior to order creation.
25  * @param {Object} context context object used for the view
26 */
27 function start(context) {
28     var cart = Cart.get();
29
30     const SubscribeProLib = require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib.js');
31     let isSubPro = SubscribeProLib.isSubPro(),
32         hasCreditCard = SubscribeProLib.hasCreditCard(cart);
33
34     // Checks whether all payment methods are still applicable. Recalculates all existing non-gift certificate payment
35     // instrument totals according to redeemed gift certificates or additional discounts granted through coupon
36     // redemptions on this page.
37     var COBilling = app.getController('COBilling');
38
39     if (isSubpro && !hasCreditCard) {
40         app.getView({
41             Basket: cart.object,
42             ContinueURL: URLUtils.https('COBilling-Billing'),
43             isSubPro: true
44         }).render('checkout/billing/billing');
45
46         return;
47     }
48
49     if (!COBilling.ValidatePayment(cart)) {
50         COBilling.Start();
51         return;
52     } else {
53         Transaction.wrap(function () {

```

Modify creditcard.xml

Locate /app_storefront_core/cartridge/forms/default/creditcard.xml form definition. Add a new form field for the customer's saved payment instrument id.

Add the following form field:

```
<field formid="customerPaymentInstrumentID" type="string"
binding="customerPaymentInstrumentID" mandatory="false" />
```

Here is the expected result:

```

1  <?xml version="1.0"?>
2  <form xmlns="http://www.demandware.com/xml/form/2008-04-19">
3
4      <!-- field for credit card type --&gt;
5      &lt;field formid="type" label="creditcard.type" type="string" mandatory="true" binding="creditCardType"
6          missing-error="creditcard.typemissing"&gt;
7          &lt;options optionid-binding="cardType" value-binding="cardType" label-binding="name"/&gt;
8      &lt;/field&gt;
9
10     <!-- field for credit card number --&gt;
11     &lt;field formid="number" label="creditcard.number" type="string" mandatory="true" masked="4" max-length="16"
12         description="creditcard.numberexample" binding="creditCardNumber"
13         missing-error="creditcard.numbermissingerror" value-error="creditcard.numbervalueerror"/&gt;
14
15     &lt;field formid="customerPaymentInstrumentID" type="string" binding="customerPaymentInstrumentID" mandatory="false" /&gt;
16
17     &lt;group formid="expiration"&gt;
18         &lt;!-- field for credit card expiration month --&gt;
19         &lt;field formid="month" label="resource.month" type="integer" mandatory="true" binding="creditCardExpirationMonth"
20             missing-error="creditcard.monthmissingerror" value-error="creditcard.yearvalueerror"&gt;
21             &lt;options&gt;
22                 &lt;option label="month.january" value="01"/&gt;
23                 &lt;option label="month.february" value="02"/&gt;
24                 &lt;option label="month.march" value="03"/&gt;
25                 &lt;option label="month.april" value="04"/&gt;
26                 &lt;option label="month.may" value="05"/&gt;
27                 &lt;option label="month.june" value="06"/&gt;
</pre>

```

Modify hooks.json

Locate `/app_storefront_core/cartridge/scripts/hooks.json` hook and remove Apple Pay hook. This hook will be set within the Subscribe Pro cartridge.

```
{  
    "name": "dw.extensions.applepay.paymentAuthorized.authorizeOrderPayment",  
    "script": "./checkout/applepay.js"  
}
```

Account

Changes need to be done to My Account section for displaying customer Subscribe Pro subscriptions. To display such information Subscribe Pro Hosted Widgets are used. Changes need to be done to checkout process with subscription products.

Modify controller Address.js

Locate /app_storefront_controllers/cartridge/controllers/Address.js.

Replace the list() method with the following:

```
function list() {
    var pageMeta = require('~/cartridge/scripts/meta');

    var content = app.getModel('Content').get('myaccount-addresses');
    if (content) {
        pageMeta.update(content.object);
    }

    var viewParams = {};
    if (session.custom.newAddress) {
        viewParams.newAddress = {"address":session.custom.newAddress.sp};
        viewParams.newAddressSfccId = session.custom.newAddress.sfcc.getID();
        session.custom.newAddress = null;
    }
    if (session.custom.updatedOldAddress && session.custom.updatedNewAddress) {
        viewParams.updatedAddress = {"prev_address": session.custom.updatedOldAddress.sp,
"address": session.custom.updatedNewAddress.sp};
        session.custom.updatedNewAddress = null;
        session.custom.updatedOldAddress = null;
    }
    if (session.custom.deletedAddress) {
        viewParams.deletedAddress = {"address": session.custom.deletedAddress.sp};
        session.custom.deletedAddress = null;
    }
    app.getView(viewParams).render('account/addressbook/addresslist');
}
```

In the handleForm() method make the following changes.

Inside the addressForm.handleAction() call, replace the create() method with the following:

```
create: function () {
    var newAddress;
    if (!session.forms.profile.address.valid || !(newAddress =
Address.create(session.forms.profile.address))) {
        response.redirect(URLUtils_https('Address-Add'));
        success = false;
    }
    session.custom.newAddress = {
        "sfcc": newAddress,
        "sp": addressHelper.getSubproAddress(newAddress, session.customer.profile,
false, true)
    };
    success = true;
},
```

Replace the edit() method with the following:

```
edit: function () {
    if (!session.forms.profile.address.valid) {
```

```

        success = false;
        message = 'Form is invalid';
    }
    try {
        // We can't use Address.get() because it doesn't return a CustomerAddress
        // object, and we need to be able to access
        // the 'custom' attributes on the address. Instead, we use the address book
        let addressBook = customer.profile.addressBook;
        var prevAddress =
addressBook.getAddress(request.httpParameterMap.addressid.value);
        session.custom.updatedOldAddress = {
            "sfcc": prevAddress,
            "sp": addressHelper.getSubproAddress(prevAddress,
session.customer.profile, true, true)
        };
        var updatedAddress = Address.update(request.httpParameterMap.addressid.value,
session.forms.profile.address);
        session.custom.updatedNewAddress = {
            "sfcc": updatedAddress,
            "sp": addressHelper.getSubproAddress(updatedAddress,
session.customer.profile, true, true)
        };
        success = true;
    } catch (e) {
        success = false;
        message = e.message;
    }
},
},

```

And replace the remove() method with the following:

```

remove: function () {
    // We can't use Address.get() because it doesn't return a CustomerAddress object,
    and we need to be able to access
    // the 'custom' attributes on the address. Instead, we use the address book
    let addressBook = customer.profile.addressBook;
    var deletedAddress =
addressBook.getAddress(session.forms.profile.address.addressid.value);
    if (Address.remove(session.forms.profile.address.addressid.value)) {
        deletedAddress = null;
        success = false;
    }
    else {
        session.custom.deletedAddress = {
            "sfcc": deletedAddress,
            "sp": addressHelper.getSubproAddress(deletedAddress,
session.customer.profile, true, true)
        };
    }
}

```

In the Delete function, add the following code between the CustomerStatusCodes and deleteAddressResult variable declarations:

```

var addressHelper =
require('int_subscribe_pro/cartridge/scripts/subpro/helpers/AddressHelper');

// We can't use Address.get() because it doesn't return a CustomerAddress object, and we
need to be able to access
// the 'custom' attributes on the address. Instead, we use the address book
let addressBook = customer.profile.addressBook;
var deletedAddress = addressBook.getAddress(request.httpParameterMap.AddressID.value);
session.custom.deletedAddress = {

```

```

    "sfcc": deletedAddress,
    "sp": addressHelper.getSubproAddress(deletedAddress, session.customer.profile, true,
true)
};

```

Lastly, add the follow new route:

```

function SetSPAddressID() {
    var addressBook = customer.getProfile().getAddressBook();
    var address = addressBook.getAddress(request.httpParameterMap.addressId.value);
    var addressHelper =
require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/AddressHelper');
    addressHelper.setSubproAddressID(address, request.httpParameterMap.spAddressId.value);

    let r = require('~/cartridge/scripts/util/Response');
    r.renderJSON({success: true});
}
/** Sets Subscribe Pro Address ID on SFCC Address */
exports.SetSPAddressID = guard.ensure(['get', 'https', 'loggedIn'], SetSPAddressID);

```

Modify controller *PaymentInstruments.js*

Locate controller /app_storefront_controllers/cartridge/controllers/PaymentInstruments.js

Replace the list() method with the following:

```

function list() {
    var wallet = customer.getProfile().getWallet();
    var paymentInstruments =
wallet.getPaymentInstruments(dw.order.PaymentInstrument.METHOD_CREDIT_CARD);
    var pageMeta = require('~/cartridge/scripts/meta');
    var paymentForm = app.getForm('paymentinstruments');

    paymentForm.clear();
    paymentForm.get('creditcards.storedcards').copyFrom(paymentInstruments);

    pageMeta.update(dw.content.ContentMgr.getContent('myaccount-paymentsettings'));

    var viewParams = {
        PaymentInstruments: paymentInstruments
    };

    if (session.custom.newCard) {
        viewParams.newCard = {"payment_profile": session.custom.newCard.sp};
        viewParams.newCardSfccId = session.custom.newCard.sfcc.getUUID();
        session.custom.newCard = null;
    }
    if (session.custom.updatedCard) {
        viewParams.updatedCard = {"payment_profile": session.custom.updatedCard.sp};
        session.custom.updatedCard = null;
    }
    if (session.custom.deletedCard) {
        viewParams.deletedCard = {"payment_profile": session.custom.deletedCard.sp};
        session.custom.deletedCard = null;
    }

    app.getView(viewParams).render('account/payment/paymentinstrumentlist');
}

```

Find the create() method. Inside the if(isDuplicateCard) condition, add the following before the wallet.removePaymentInstrument() call:

```

if (oldCard.custom.subproPaymentProfileID) {
    subproPaymentProfileID = oldCard.custom.subproPaymentProfileID;
}

```

Directly between the `Transaction.commit()` and `paymentForm.clear()` calls, add the following code:

```

// Save data to session for hosted wallet widget integration
var paymentInstruments =
wallet.getPaymentInstruments(dw.order.PaymentInstrument.METHOD_CREDIT_CARD);
var saveCard;
for (var i = 0; i < paymentInstruments.length; i++) {
    var card = paymentInstruments[i];
    if (card.creditCardNumber === ccNumber) {
        saveCard = card;
        break;
    }
}
var paymentsHelper =
require('int_subscribe_pro/cartridge/scripts/subpro/helpers/PaymentsHelper');
if (!isDuplicateCard) {
    session.custom.newCard = {
        "sp": paymentsHelper.getSubscriptionPaymentProfile(session.customer.profile,
saveCard, {}, false),
        "sfcc": saveCard
    };

    if (null === session.custom.newCard) {
        Logger.info('New card could not be mapped to a payment profile.');
    }
} else {
    Transaction.begin();
    try {
        saveCard.custom.subproPaymentProfileID = subproPaymentProfileID;
    }
    catch (err) {
        Transaction.rollback();
        return false;
    }
    Transaction.commit();

    session.custom.updatedCard = {
        "sp": paymentsHelper.getSubscriptionPaymentProfile(session.customer.profile,
saveCard, {}, true),
        "sfcc": saveCard
    };
    if (null === session.custom.updatedCard) {
        Logger.info('New card could not be mapped to a payment profile.');
    }
}

```

In the `Delete()` method, right inside the `remove()` function, add the following:

```

var paymentsHelper =
require('int_subscribe_pro/cartridge/scripts/subpro/helpers/PaymentsHelper');
session.custom.deletedCard = {
    "sp": paymentsHelper.getSubscriptionPaymentProfile(session.customer.profile,
action.object, {}, true),
    "sfcc": action.object
};

```

Add the following route:

```
function SetSPPaymentProfileID() {
    var wallet = customer.getProfile().getWallet();
    var paymentInstruments = wallet.getPaymentInstruments('CREDIT_CARD');
    let paymentInstrumentId = request.httpParameterMap.paymentInstrumentId.value;
    var paymentsHelper =
require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/PaymentsHelper');
    let paymentInstrument = null;
    for (let i in paymentInstruments) {
        if (paymentInstrumentId == paymentInstruments[i].getUUID()) {
            paymentInstrument = paymentInstruments[i];
        }
    }

    let success = paymentInstrument != null;

    if (success) {
        paymentsHelper.setSubproPaymentProfileID(paymentInstrument,
request.httpParameterMap.spPaymentProfileId.value);
    }

    let r = require('~/cartridge/scripts/util/Response');
    r.renderJSON({success: true});
}

/** Sets the Subscribe Pro Payment Profile ID on a SFCC Payment Instrument */
/* @see module:controllers/PaymentInstruments~SetSPPaymentProfileID */
exports.SetSPPaymentProfileID = guard.ensure(['https', 'get', 'loggedIn'],
SetSPPaymentProfileID);
```

Modify addressinclude.isml

Locate template

/app_storefront_core/cartridge/templates/default/account/addressbook/addressinclude.isml.

Insert code below at the end of the file to facilitate configuring and loading the address book widget, "AddressBookAssist".

```
<isset name="hasNewAddress" value="${ pdict.newAddress != null }" scope="page" />
<isset name="hasUpdatedAddress" value="${ pdict.updatedAddress != null }" scope="page" />
<isset name="hasDeletedAddress" value="${ pdict.deletedAddress != null }" scope="page" />
<isset name="loadWidget" value="${ hasNewAddress || hasUpdatedAddress || hasDeletedAddress
}" scope="page" />
<isif condition="${customer.profile.custom.subproCustomerID && loadWidget}">
    <!-- Subscribe Pro Address Book Assist Widget -->
    <div id="sp-address-book-widget"></div>

    <iscomment>
        Configure the Subscribe Pro Address Book Assist Widget
    </iscomment>

    <!-- Load the Subscribe Pro widget script -->
    <script
src="${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subproAddressWidgetsScriptUrl')}"></script>

    <isset name="widgetConfig" scope="page"
value="${require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/WidgetsHelper')}
```

```

.getWidgetConfig(customer.profile.custom.subproCustomerID, 'client_credentials', 'widget',
'sp-address-book-widget')"/>

<!-- Pass configuration and init the Subscribe Pro widget -->
<script>
    // Setup config for Subscribe Pro
    var widgetConfig = <isprint value="${JSON.stringify(widgetConfig)}" encoding="off" />;
        AddressBookAssist.init(widgetConfig);
        <if condition="${ hasNewAddress }">
            var newAddress = <isprint value="${JSON.stringify(pdct.newAddress)}" encoding="off" />;
            var newAddressSfccId = '<isprint value="${pdct.newAddressSfccId}" encoding="off" />';
            AddressBookAssist.onAddressCreated(newAddress).then(function (result)
{
                $.get("${URLUtils.url('Address-SetSPAddressID')}",
{"addressId": newAddressSfccId, "spAddressId": result.id});
            });
        </if>
        <if condition="${ hasUpdatedAddress }">
            var updatedAddress = <isprint value="${JSON.stringify(pdct.updatedAddress)}" encoding="off" />;
            AddressBookAssist.onAddressUpdated(updatedAddress);
        </if>
        <if condition="${ hasDeletedAddress }">
            var deletedAddress = <isprint value="${JSON.stringify(pdct.deletedAddress)}" encoding="off" />;
            AddressBookAssist.onAddressDeleted(deletedAddress);
        </if>
    </script>
</if>

```

Modify paymentinstrumentlist.isml

Locate template

/app_storefront_core/cartridge/templates/default/account/payment/paymentinstrumentlist.isml.

- 1) Inside the isdecorate tag, set the variable to define if customer is Subscribe Pro customer

```
<isset name="isSubproCustomer" value="${customer.profile.custom.subproCustomerID}" scope="page" />
```

- 2) Insert code below just before the closing div tag for the <div class="paymentslist"> block to add Subscribe Pro “Address Wallet Assist” widget div.

```
<if condition="${isSubproCustomer}">
    <!-- Subscribe Pro Wallet Assist Widget -->
    <div id="sp-wallet-assist-widget"></div>
</if>
```

- 3) Insert code below to load, configure and init the Subscribe Pro “Address Wallet Assist” widget

```

<isif condition="${isSubproCustomer && loadWidget}">
    <iscomment>
        Configure the Subscribe Pro Wallet Assist Widget
    </iscomment>

    <!-- Load the Subscribe Pro widget script -->
    <script
src="${require('dw/system/Site').getCurrent().getCustomPreferenceValue('subproWalletWidgetScriptUrl')}"></script>

    <isset name="widgetConfig" scope="page" value="${require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/WidgetsHelper').getWidgetConfig(customer.profile.custom.subproCustomerID, 'client_credentials', 'widget', 'sp-wallet-assist-widget')}">

        <!-- Pass configuration and init the Subscribe Pro widget -->
        <script>
            // Setup config for Subscribe Pro
            var widgetConfig = <isprint value="${JSON.stringify(widgetConfig)}" encoding="off" />;
            WalletAssist.init(widgetConfig);
            <if condition="${ hasNewCard }">
                var newCard = <isprint value="${JSON.stringify(pdct.newCard)}" encoding="off" />;
                WalletAssist.onPaymentProfileCreated(newCard, function(result) {
                    var newCardSfccId = '<isprint value="${pdct.newCardSfccId}" encoding="off" />';
                    $.get("${URLUtils.url('PaymentInstruments-SetSPPaymentProfileID')}", {"paymentInstrumentId": newCardSfccId, "spPaymentProfileId": result.id});
                });
            </if>
            <if condition="${ hasUpdatedCard }">
                var updatedCard = <isprint value="${JSON.stringify(pdct.updatedCard)}" encoding="off" />;
                WalletAssist.onPaymentProfileUpdated(updatedCard);
            </if>
            <if condition="${ hasDeletedCard }">
                var deletedCard = <isprint value="${JSON.stringify(pdct.deletedCard)}" encoding="off" />;
                WalletAssist.onPaymentProfileDeleted(deletedCard);
            </if>
        </script>
    </if>

```

My Account side navigation

To display My subscriptions in the My Account side navigation edit asset account-nav-registered (*Merchant Tools > Content > Content Assets > account-nav-registered*).

Add the following:

```
<li><a title="Manage subscriptions" href="$httpsUrl(Subscriptions-List)$">My Subscriptions</a></li>
```

Modify controller Account.js

Locate `/app_storefront_controllers/cartridge/controllers/Account.js` controller. Add the following includes to the top of the controller to be used later on in the controller:

```
/* Subscribe Pro Module */
var SubscribeProLib = require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib');
var CustomerHelper = require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/CustomerHelper');
var Logger = require('dw/system/Logger');
```

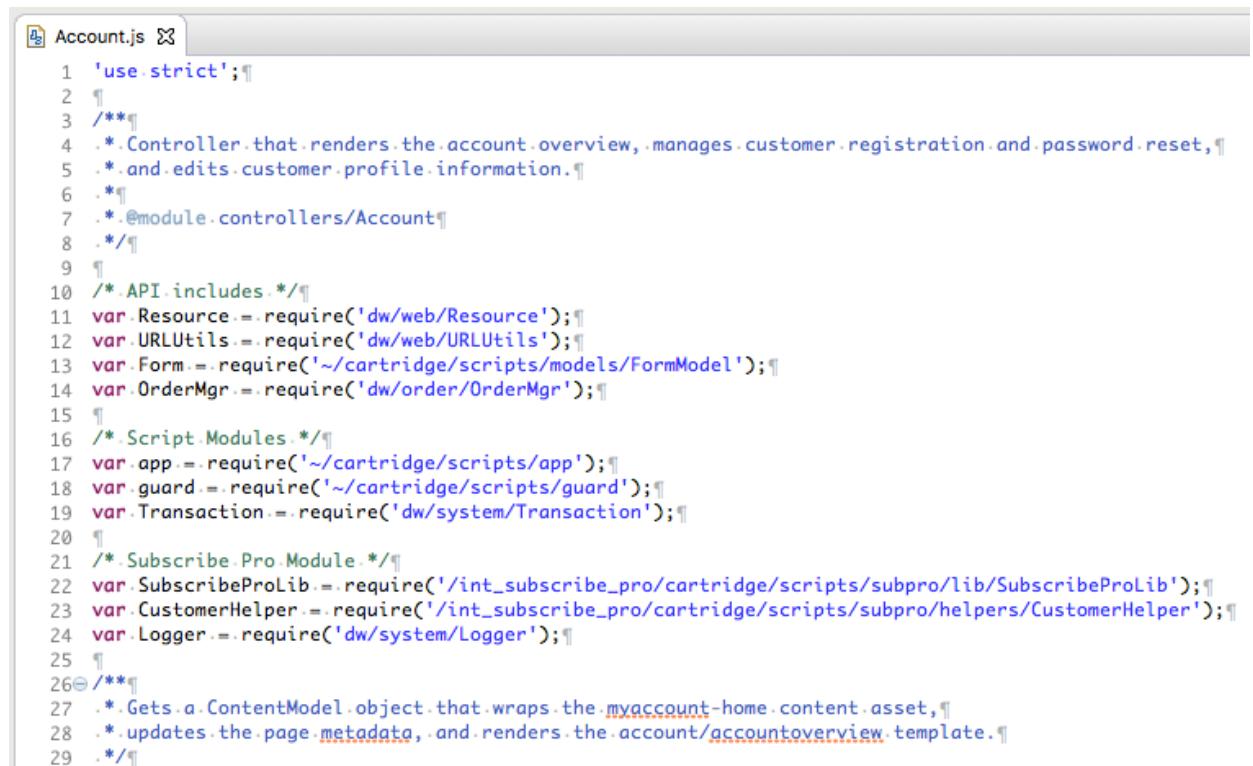
Modify the `hasEditSucceeded` conditional in the `editForm` method to have some logic in the case of the conditional returning true, to update the customer profile at Subscribe Pro:

```
if (hasEditSucceeded) {
    let customer = Customer.get();

    if (customer && customer.object && customer.object.profile &&
        customer.object.profile.custom.subproCustomerID) {
        let customerToPost = CustomerHelper.getSubproCustomer(customer);
        let updateResponse =
        SubscribeProLib.updateCustomer(customer.object.profile.custom.subproCustomerID, customerToPost);

        if (updateResponse.error) {
            Logger.error('There was an issue updating the customer at subscribe pro: ' +
            JSON.stringify(updateResponse));
        }
    } else {
        /* Existing false logic for hasEditSucceeded */
```

Here is the expected result:



The screenshot shows a code editor window with the title "Account.js". The code is a JavaScript file with several require statements and annotations. Lines 10 through 24 are new, added by the user, which include the required modules for Subscribe Pro. Lines 27 and 28 are annotations explaining the purpose of the code.

```
1  'use strict';¶
2  ¶
3  /**¶
4   * .Controller.that.render.the.account.overview,.manages.customer.registration.and.password.reset,¶
5   * .and.edits.customer.profile.information.¶
6   * .¶
7   * .@module.controllers/Account¶
8   * .*/¶
9  ¶
10 /*.API.includes.*/¶
11 var.Resource.=.require('dw/web/Resource');¶
12 var.URLUtils.=.require('dw/web/URLUtils');¶
13 var.Form.=.require('~/cartridge/scripts/models/FormModel');¶
14 var.OrderMgr.=.require('dw/order/OrderMgr');¶
15 ¶
16 /*.Script.Modules.*/¶
17 var.app.=.require('~/cartridge/scripts/app');¶
18 var.guard.=.require('~/cartridge/scripts/guard');¶
19 var.Transaction.=.require('dw/system/Transaction');¶
20 ¶
21 /*.Subscribe.Pro.Module.*/¶
22 var.SubscribeProLib.=.require('/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib');¶
23 var.CustomerHelper.=.require('/int_subscribe_pro/cartridge/scripts/subpro/helpers/CustomerHelper');¶
24 var.Logger.=.require('dw/system/Logger');¶
25 ¶
26/**¶
27 *.Gets.a.ContentModel.object.that.wraps.the.myaccount-home.content.asset,¶
28 *.updates.the.page.metadata,.and.renders.the.account/accountoverview.template.¶
29 .*/¶
```

```
Account.js
```

```
72 .*.Handles the form submission on profile.update.of edit.profile..Handles cancel and confirm actions.¶
73 .*...cancel--clears the profile.form and redirects to the Account>Show controller.function.¶
74 .*...confirm--gets a CustomerModel object that wraps the current customer..Validates several form fields.¶
75 .*.If any of the profile.validation conditions fail, the user is redirected to the Account>EditProfile.controller.func
76 */
77 function editForm(){¶
78 ... app.getForm('profile').handleAction({¶
79 ... ... cancel: function() {¶
80 ... ... ... app.getForm('profile').clear();¶
81 ... ... ... response.redirect(URLUtils_https('Account>Show'));¶
82 ... ... },¶
83 ... ... confirm: function(){¶
84 ... ... ... var isProfileUpdateValid = true;¶
85 ... ... ... var hasEditSucceeded = false;¶
86 ... ... ... var Customer = app.getModel('Customer');¶
87 ... ¶
88 ... ... if (!Customer.checkUserName()){¶
89 ... ... ... app.getForm('profile.customer.email').invalidate();¶
90 ... ... ... isProfileUpdateValid = false;¶
91 ... ... }¶
92 ... ¶
93 ... ... if (app.getForm('profile.customer.email').value() != app.getForm('profile.customer.emailconfirm').value())¶
94 ... ... ... app.getForm('profile.customer.emailconfirm').invalidate();¶
95 ... ... ... isProfileUpdateValid = false;¶
96 ... ... }¶
97 ... ¶
98 ... ... if (!app.getForm('profile.login.password').value()){¶
99 ... ... ... app.getForm('profile.login.password').invalidate();¶
100 ... ... ... isProfileUpdateValid = false;¶
101 ... ... }¶
102 ... ¶
103 ... ... if (isProfileUpdateValid) {¶
104 ... ... ... hasEditSucceeded = Customer.editAccount(app.getForm('profile.customer.email').value(), app.getForm('pr
105 ... ¶
106 ... ... if (hasEditSucceeded) {¶
107 ... ... ... let customer = Customer.get();¶
108 ... ¶
109 ... ... ... if (customer && customer.object && customer.object.profile && customer.object.profile.custom.subpr
110 ... ... ... let customerToPost = CustomerHelper.getSubproCustomer(customer);¶
111 ... ... ... let updateResponse = SubscribeProLib.updateCustomer(customer.object.profile.custom.subproCusto
112 ... ¶
113 ... ... ... if (updateResponse.error) {¶
114 ... ... ... ... Logger.error('There was an issue updating the customer at subscribe.pro: ' + JSON.stringify(
115 ... ... ... ... })¶
116 ... ... ... } else {¶
117 ... ... ... ... app.getForm('profile.login.password').invalidate();¶
118 ... ... ... ... isProfileUpdateValid = false;¶
119 ... ... ... }¶
120 ... ... }¶
121 ... ... }¶
122 ... ¶
123 ... ... if (isProfileUpdateValid && hasEditSucceeded) {¶
```

3.4 External Interfaces

Web service calls made to the Subscribe Pro RESTful API will make use of Sales Force Commerce Cloud Services:

<https://documentation.demandware.com/DOC1/topic/com.demandware.dochelp/WebServices/Webservices.html>

Application code should not call the web services directly but should instead call methods within the Subscribe Pro Javascript module:

LINK-subscribe-pro/int_subscribe_pro/cartridge/scripts/subpro/lib/SubscribeProLib.js

Service Profile

There are separate profiles for GET vs POST request to enable / disable the circuit breaker.

Name	subpro.http.prof.get	subpro.http.prof.post
Description	Profile for GET requests	Profile for POST requests
Connection Timeout (ms)	5000	5000
Enable Circuit Breaker	yes	no
Max Circuit Breaker Calls	3	0
Enable Rate Limit	1000	1000
Max Rate Limit Calls		
Rate Limit Interval (ms)		

Service Credentials

The "siteName" portion of the Service Credential comes from the Site Preference: *subproAPICredSuffix*. This allows merchants to have different credentials for each site.

The default value of the Site Preference is *siteName*, so that it will automatically match the default credentials.

Please note that the user password will need to be updated after import.

Name	subpro.http.cred.siteName
URL	https://api.subscribe.com/services/v2/{ENDPOINT}?{PARAMS}
User	Subscribe Pro Client ID
Password	Subscribe Pro Client Secret
Name	subpro.http.cred.oauth.siteName

URL	https://api.subscribe.com/oauth/v2/{ENDPOINT}?{PARAMS}
User	Subscribe Pro Client ID
Password	Subscribe Pro Client Secret

Service Endpoints

ID	API endpoint	Description
subpro.http.get.config	GET /services/v2/config	Retrieve configuration for the authenticated account
subpro.http.get.subscriptions	GET /services/v2/subscriptions	Retrieve a collection of subscriptions.
subpro.http.post.subscription	POST /services/v2/subscription	Create a new subscription
subpro.http.post.addresses	POST /services/v2/address	Create / Update a Subscribe Pro addresses
subpro.http.post.addressfindcreate	POST /services/v2/address/find-or-create	Find a matching address or create a new one
subpro.http.get.addresses	GET /services/v2/addresses	Retrieve a collection of addresses
subpro.http.get.products	GET /services/v2/products/{id}	Retrieve a single product by id
subpro.http.get.customers	GET /services/v2/customers/{id}	Retrieve a single customer by id
subpro.http.post.customer	POST /services/v2/customer	Create a new customer
subpro.http.post.customers	POST /services/v2/customers/{id}	Update a single customer, identified by customer id
subpro.http.get.token	GET /oauth/v2/token	Get an OAuth2 access token
subpro.http.get.paymentprofile	GET /services/v2/vault/paymentprofiles/{id}	Retrieve a single payment profile by id
subpro.http.post.paymentprofile.vault	POST /services/v2/vault/paymentprofile/external-vault	Create a new payment profile for an external vault

For more details visit Subscribe Pro REST API Documentation:

<https://feat-sfcc-cd75l4i-cdbrvib5kpzoa.us.platform.sh/docs/rest>

3.5 Testing

Sandbox(es)

In order to test the application in a non-production environment, you will need to have a Subscribe Pro Sandbox.

Subscribe Pro provides "sandbox" accounts to all merchants.

- One Sandbox account is included with standard plans
- Additional sandbox accounts (or those which aren't associated with a paid plan) can be added but will incur a fee.

Integration Testing

Subscription Products

The following are some test cases that can be completed to ensure subscription products are configured correctly.

1. User's should see subscription options for products, on the product detail page, that have subscription's enabled and a corresponding product has been configured on the Subscribe Pro Platform.
2. User's should see an option for a one time purchase, if this option has been configured on Subscribe Pro.
3. User's should see an option to purchase a product on a recurring basis, if this option has been configured on Subscribe Pro.
4. User's should see the configured repeat Order options and be able to select the desired frequency of recurring orders.
5. User's should be able to add a product, with subscription options, to the Basket and see the selected options present in the: Mini-Cart, Basket, Order Details (in the right rail of Checkout), the Order Confirmation Screen and in the Order Confirmation Email.
6. User's should be able to change their subscription options while viewing the product in the Basket.
7. Subscription details should be transmitted to Subscribe Pro through a configured Job / Job Schedule.
8. Failed requests to create a subscription should be emailed to the configured user for further investigation.

My Account

The following are some test cases that can be completed to ensure that the My Account has been configured correctly.

1. User's should be able to see a list of their Subscription orders in the "My Subscriptions" section of "My Account".
2. User's should see an option to update their Subscription Payment option upon changing their saved payment instrument in "My Account".
3. User's should see an option to update their Subscription Address option upon changing their saved address in "My Account".
4. User's should be able to update their profile information in "My Account" and those details are automatically transmitted to Subscribe Pro.

Pixel / Widget Tracking

The following are some test cases that can be completed to ensure that the Subscribe Pro Widget is included in all of the necessary locations.

1. The Javascript Widget should be executed and transmitting details to Subscribe Pro via Ajax on the following pages:
 - a. My Account - Addresses
 - b. My Account - Payment Instruments
 - c. My Account - My Subscriptions
 - d. Product Detail Page
 - e. Product Search Results

4. Operations, Maintenance

4.1 Data Storage

Subscription selections and Subscribe Pro identifiers (customer, address, subscription) will be stored to the custom attributes of the Sales Force Commerce Cloud System Objects (Product, Product Line item, Order, Basket).

Customer information such as name, email, address and some payment information will be stored at Subscribe Pro for any customers that have purchased subscription products. This information will be used to process future recurring orders for the customer.

4.2 Availability

Subscribe Pro hosted services and API's perform with an uptime of 99.9% reliability.

If external services are not available for products, subscription options will not be available for subscription products. The end user will still be able to purchase products as normal on the platform but subscription options will not be available.

If external services are not available for My Account, the end user will see a My Subscription page without any content.

If external services are not available for transmitting subscription order details, via a scheduled job, an email should be sent to the configured contact and they will need to manually process the subscription.

4.3 Support

Support will be provided through the Subscribe Pro support system.

All requests should include: details of issue, email, phone, and URL.

Requests can be made at the following location:
<https://www.subscribepr.com/support/>

Technical Support Contact:

John Sparwasser

john.sparwasser@subscribepr.com

5. User Guide

5.1 Roles, Responsibilities

<LIST RECURRING TASKS THAT NEED TO BE FULFILLED BY CUSTOMER, MERCHANT TO RUN THE INTEGRATION, E.G. MANUAL FEED OF CATALOG DATA INTO 3RD PARTY SERVICE, IF APPLICABLE>

5.2 Business Manager

Subscribe Pro group is added to Custom Site Preference Groups (*Site > Site Preferences > Custom Preferences*). It contains following attributes:

The screenshot shows a configuration interface for 'Custom Site Preference Groups'. At the top, there's a breadcrumb navigation: Merchant Tools / Site Preferences / Custom Site Preference Groups / Subscribe Pro. To the right are three buttons: Cancel, Apply to Other Sites, and Save. Below the navigation is a dropdown for 'Instance Type' set to 'Sandbox'. A search bar with placeholder 'Search by IDs...' and a magnifying glass icon is followed by a dropdown menu showing '1-7 of 7'. The main area is a table with columns: Name, Value, and Default Value. Each row represents a preference key with its current value and a link to edit it across sites.

Name	Value	Default Value
Subpro Credit Card Requirement Message	You need credit card to pay for SubPro subscripti on Error message noting that a credit card is required for...	You need a credit card to pa... Edit Across Sites
Subpro Checkout Login Message	You need to be logged in as a registered customer Message informing customer that he/she needs to be ...	You need to be logged in as... Edit Across Sites
Subscribe Pro Order Processing Error Mail	omahdybor@fluid.com An email address that order subscriptions processing ...	Edit Across Sites
Subpro Apple Pay Login Message	 Message informing customer that he/she needs to be ...	You need to be logged in as... Edit Across Sites
Subscribe Pro Widget Script Url	 	https://widgets.subscribepr... Edit Across Sites
Subscribe Pro API Base Url	 	https://api.subscribe.com Edit Across Sites
Suffix to append to the Subscribe Pro Web Service Credentials	 	siteName Edit Across Sites

5.3 Storefront Functionality

Product Detail Page

When the product page template has been updated, you should be able to see subscription options for product: one time delivery or regular delivery and drop down with delivery intervals if regular delivery is available. This information will be shown only for products for which subscription is configured in SFCC Business Manager and in Subscribe Pro Business Manager.

Womens / Clothing / Tops / Scoop Neck Tee With Applique

The screenshot shows a product detail page for a "Scoop Neck Tee With Applique". The product is a blue t-shirt. The page includes a large main image of a woman wearing the t-shirt, two smaller thumbnail images below it, and a "SELECT COLOR" section with "BLUE" highlighted. Below that is a "SELECT SIZE" section with sizes S, M, L, XL, and S selected. A "Size Chart" link is also present. Under "AVAILABILITY", it says "In Stock". A red box highlights the "Subscription Options" section, which contains two radio buttons: "ONE-TIME DELIVERY" (unchecked) and "REGULAR DELIVERY - 0% DISCOUNT" (checked). Below this is a "DELIVERY EVERY" dropdown menu set to "Monthly". At the bottom, there is a "QTY" input set to "1" and a green "ADD TO CART" button. Social sharing icons for Facebook, Twitter, Google+, Pinterest, and Email are at the bottom, along with "Add to Wishlist" and "Add to gift registry" buttons.

Cart Page

If any product with subscription was added to basket, on cart page you will see the same subscription options as on PDP for such product line item. At this step subscription options can be changed. To apply changes you'll need to click on 'Update Cart' button.

« Continue Shopping CHECKOUT

PRODUCT	DELIVERY OPTIONS	QTY	PRICE	TOTAL
 Scoop Neck Tee With Applique - Product Price Adjustment Item No.: 701644059262 Color Blue Size S Edit Details	Home Delivery	<input type="text" value="1"/>	In Stock Remove Add to Wishlist	\$39.00 List Price \$39.00 Item Total \$39.00

Subscription Options

One-time delivery

Regular delivery - 0% Discount

Delivery every

Monthly

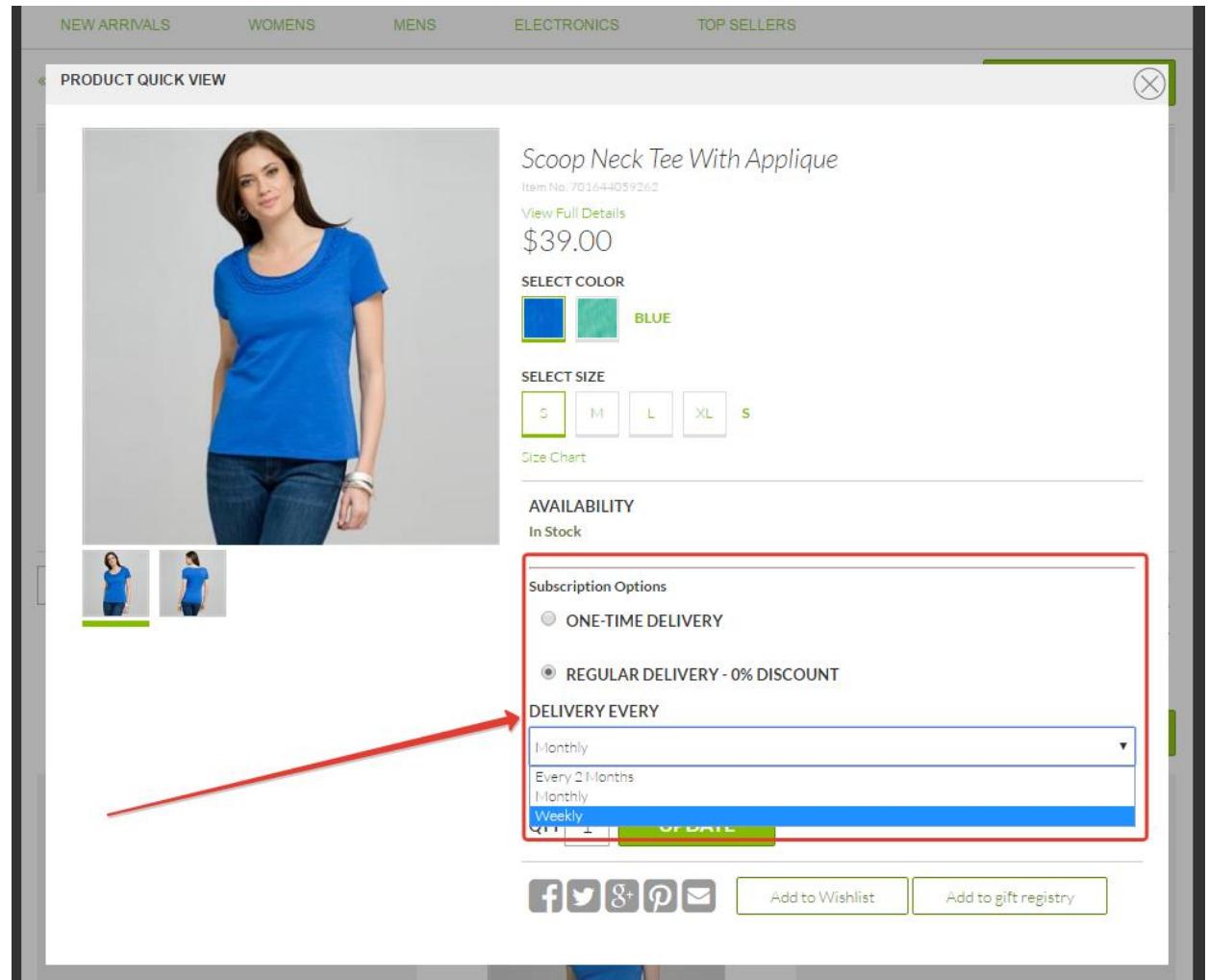
Apply Update Cart

Subtotal	\$39.00
Shipping	-
Sales Tax	-
Estimated Total	\$39.00

[« Continue Shopping](#) CHECKOUT

Product Page (Edit Details Quickview)

While on cart page you can click on the 'Edit Details' link. It will show Quick View containing PDP, where you'll see subscription options. They can be changed. To apply changes click on 'Update' button.



Login Page

If customer has any product with regular subscription option selected, he/she must be logged in as a registered customer to place order with subscriptions. In other case customer will see error message informing that he/she should log in to proceed. 'Checkout as Guest' option is excluded.

The screenshot shows a commerce website's login page. At the top, there is a header with the 'salesforce commerce cloud' logo, a search bar, and user account icons. Below the header, there are navigation links for 'NEW ARRIVALS', 'WOMENS', 'MENS', 'ELECTRONICS', and 'TOP SELLERS'. The main content area is divided into two sections: 'GUESTS OR NEW CUSTOMERS' on the left and 'RETURNING CUSTOMERS • REQUIRED' on the right. The 'GUESTS OR NEW CUSTOMERS' section contains a button labeled 'Create Account Now' and a link 'What are the benefits?'. The 'RETURNING CUSTOMERS' section contains fields for 'Email Address' and 'Password', both marked as required. There is also a 'Login' button, a 'Remember Me' checkbox, and links for 'Forgot Password?' and 'OR'. Below the 'OR' link, there are social media login icons for Google, LinkedIn, Microsoft, Facebook, Twitter, and VK.

Log in to place order with subscription

GUESTS OR NEW CUSTOMERS

Create Account Now

What are the benefits?

RETURNING CUSTOMERS • REQUIRED

Email Address

Password

Login

Forgot Password?

OR

Log in using your account from

g in Microsoft f tw vk

Order Summary Right Rail and Minicart

For products with subscription selected options will be shown on Order Summary Right Rail block and on Minicart.

NEW ARRIVALS WOMENS MENS ELECTRONICS TOP SELLERS

STEP 1: Shipping > STEP 2: Billing > STEP 3: Place Order

Help? 800-555-0199

SELECT OR ENTER SHIPPING ADDRESS • REQUIRED

Select an Address Select from Saved Addresses ▾

• First Name Alexander

• Last Name Mahdybor

• Address 1 Myru ave. 75B 61

Address 2 APO/FPO

• City Chernihiv

• ZIP Code 14005

• Country United States ▾

• State American Samoa ▾

• Phone 333-333-3333

Why is this required?
Example: 333-333-3333
 Add to Address Book
 Use this address for Billing

Is this a gift? Yes No

ORDER SUMMARY Edit


Scoop Neck Tee With Applique
Color Blue
Size S
Qty: 1 \$39.00
Subscription info
Delivery mode
regular
Delivery interval
Monthly

Subtotal \$39.00
Edit Shipping Store \$0.00
Pickup
Sales Tax \$1.95
Order Total: \$40.95

Checkout Billing Page

To place order with subscription Credit Card or Apple Pay payment method is required, even if customer has completely paid for order with alternative payment methods, without a credit card, using a gift card for example. The user must be required to enter a credit card number or Apple Pay for the subscription. The message informing customer about it will be shown.

The screenshot shows a payment method selection interface. At the top, a green box displays the message: "USD 58.79 has been redeemed from gift certificate *****GGVC Remove". Below this, a red box highlights the message: "You need credit card to pay for SubPro subscription!". A red arrow points from this message to the "Credit Card" radio button, which is selected. Other options shown are "Apple Pay", "Pay Pal", and "Bill Me Later". Further down, there are fields for "Select Credit Card" (dropdown menu "Select from saved cards"), "Name on Card", "Type" (dropdown menu "Visa"), "Number" (with placeholder "Example: 4111111111111111"), "Expiration Date" (dropdown menus for month "January" and year "2016"), and "Security Code" (input field with "What is this?" link). A checkbox "Save this card" is checked, with a note below stating: "Your credit card information will automatically be saved to your account for your subscriptions". A large grey button at the bottom right says "CONTINUE TO PLACE ORDER >".

USD 58.79 has been redeemed from gift certificate *****GGVC Remove

SELECT PAYMENT METHOD • REQUIRED

You need credit card to pay for SubPro subscription!

Apple Pay Credit Card Pay Pal

Bill Me Later

Select Credit Card Select from saved cards

• Name on Card

• Type Visa

• Number

Example: 4111111111111111

• Expiration Date: January 2016

• Security Code What is this?

Save this card
Your credit card information will automatically be saved to your account for your subscriptions

CONTINUE TO PLACE ORDER >

Also if credit card was selected its information will be force saved to customer's account to pay for recurring subscriptions. Message, informing customer about it, will be shown.

SELECT PAYMENT METHOD • REQUIRED

Apple Pay Credit Card Pay Pal

Bill Me Later

Select Credit Card (Visa) ****1111 - Expiration 01.2022 ▾

• Name on Card Alexander Mahdybor

• Type Visa ▾

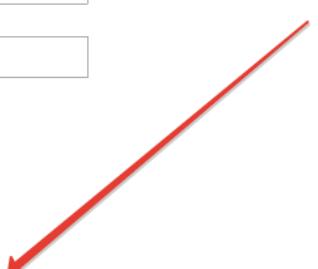
• Number ****1111
Example: 4111111111111111

• Expiration Date: January ▾ 2022 ▾

• Security Code 123 [What is this?](#)

Save this card
Your credit card information will automatically be saved to your account for your subscriptions

CONTINUE TO PLACE ORDER >



Order Summary Page

When the order summary page template has been updated, you should be able to see selected subscription options for product in order.

NEW ARRIVALS WOMENS MENS ELECTRONICS TOP SELLERS

STEP 1: Shipping > STEP 2: Billing > **STEP 3: Place Order**

Help? 800-555-0199

PRODUCT	QTY	TOTAL
 Scoop Neck Tee With Applique - Product Price Adjustment Item No.: 701644059262 Color Blue Size S Subscription info Delivery mode regular Delivery interval Monthly	1 In Stock	\$39.00
	Subtotal	\$39.00
	Shipping Express	\$16.99
	Sales Tax	\$2.80
	Order Total	\$58.79

« Edit Cart **PLACE ORDER**

ORDER SUMMARY [Edit](#)

Subtotal	\$39.00
Edit Shipping Express	\$16.99
Sales Tax	\$2.80
Order Total:	\$58.79

SHIPPING ADDRESS [Edit](#)

United States
Method: Express

BILLING ADDRESS [Edit](#)

United States

PAYMENT METHOD [Edit](#)

Credit Card
Visa ****1111
Exp. 01.2022
Amount: \$58.79

Order Confirmation Page

When the order confirmation page template has been updated, you should be able to see selected subscription options for product in order.

Thank you for your order.

If you have questions about your order, we're happy to take your call (800-555-0199) Monday - Friday, 8AM - 8PM

 Order Number: **00002001**

Order Placed: **Apr 26, 2017**

PAYMENT METHOD

Credit Card
Alexander Mahdybor
Visa
*****1111
Amount: \$58.79

PAYMENT TOTAL

Subtotal	\$39.00
Shipping Express	\$16.99
Sales Tax	\$2.80
Order Total:	\$58.79

BILLING ADDRESS

Alexander Mahdybor
Alexander Mahdybor
United States
Phone: 333-333-3333

SHIPMENT NO. 1

SHIPPING STATUS:

Not Shipped

METHOD:

Express

SHIPPING TO

Alexander Mahdybor
Alexander Mahdybor
United States
333-333-3333

ITEM

Scoop Neck Tee With Applique

- Product Price Adjustment

ITEM NO.: 701644059262

COLOR Blue

SIZE S

Subscription info

Delivery mode

regular

Delivery interval

Monthly

QTY

1

PRICE

\$39.00

Order Confirmation Email

When the order confirmation email template has been updated, you should be able to see selected subscription options for product in order in Order Confirmation email sent to customer.

The screenshot shows an order confirmation email from Salesforce Commerce Cloud. At the top left is the Salesforce commerce cloud logo. To its right is the company address: Salesforce Commerce Cloud SiteGenesis, 5 Wall Street, Burlington, MA 01803, salesforce.com, Phone: +1.800.555.0199. Below this is a grey header bar with the text "Thank you for your order." In the main body, there's a message: "If you have questions about your order, we're happy to take your call" followed by a phone number "1 (800) 555-0199, Monday - Friday 8AM - 8PM". The email continues with order details: "Order Placed: Apr 26, 2017" and "Order Number: 00002001". A "Payment Information" section shows a redacted billing address, payment method (Credit Card, Visa, ending in 1111, amount \$58.79), and a payment total of \$58.79. The "Shipment#1" section lists an item: "Scoop Neck Tee With Applique" (Quantity 1, Price \$39.00). It also shows shipping information: "Shipping To" (redacted address, United States, Method: Express, Shipping Status: Not Shipped). A red box highlights the "Subscription info" section under the item details, which includes "Delivery mode regular" and "Delivery interval Monthly". A red arrow points from this highlighted area to the "Subscription info" text in the "Shipping To" section.

Salesforce Commerce Cloud SiteGenesis
5 Wall Street
Burlington, MA 01803
salesforce.com
Phone: +1.800.555.0199

Thank you for your order.

If you have questions about your order, we're happy to take your call
[1 \(800\) 555-0199](tel:18005550199), Monday - Friday 8AM - 8PM

Order Placed: Apr 26, 2017
Order Number: 00002001

Payment Information

Billing Address	Payment Method	Payment Total
[Redacted]	Credit Card Visa *****1111 Amount: \$58.79	Subtotal \$39.00 Shipping Express \$16.99 Sales Tax \$2.80 Order Total: \$58.79

Shipment#1

Item	Quantity	Price	Shipping To
Scoop Neck Tee With Applique - Product Price Adjustment Item No.: 701644059262 Color Blue Size S Subscription info Delivery mode regular Delivery interval Monthly	1	\$39.00	[Redacted] United States Method: Express Shipping Status: Not Shipped

6. Known Issues

Talk about the subscription

7. Release History

Version	Date	Changes
17.1.0	<DATE>	Initial release