Signifyd INTEGRATION

Version 18.1.0

Table of Contents

Intended Audience

This document is intended for the technical audience that will be directly involved in the setup and or integration of this Salesforce Commerce Cloud cartridge.

1. Summary

Signifyd is a fraud solution that for this cartridge implementation will be integrated into Salesforce Commerce Cloud using the following two primary API integration points. Signifyd (Create Case) does not immediately (asynchronous) return a fraud approval or disapproval but rather relies on the callback response to provide this answer.

Action 1. The first is calling the Signifyd Create Case rest API after the Salesforce Commerce Cloud order has gone through the authentication process against the payment provider right before displaying the order summary page. Because it is only called during order creation this will ensure that create case is never called again for that same order.
The use of DWRE Service Framework is in use by all service calls.

Action 2. The second integration point is a publicly accessible URL that will be used as the callback/web-hook endpoint. This endpoint will be called when certain actions within Signifyd happen such as when an updated to the Case is made or the final fraud decision is made. This triggers and update to the order in SFCC and could also indicate that the order is ready to export(depending on settings).

This document primarily serves as the LINK implementation guide for setting up Signifyd on standard SiteGenesis.

The Set Up and Custom Code Configuration described in this document assume the use of SiteGenesis 103.1.11 release of app_storefront_core. Custom coding might be required if adapting the cartridge to work with other SiteGenesis releases, pre-2.0 releases, and versions of SiteGenesis that do not include the RequireJS framework.

2. Components

Cartridge Name
Int_signifyd

New Signifyd Controller
controllers/Signifyd.js

Modified System Controller
COSummary

Modified System Pipeline
COSummary

Modified Core Template
htmlhead.isml

Scripts
service/signifydInit.ds
service/signifyd.ds
service/pp_signifyd.ds
job/CreateMissingOrders.js

Templates
default/signifyd_device_fingerprint.isml

Cartridge Path

Pipeline based approach:
int_signifyd:app_storefront_pipelines:app_storefront_core...

Controllers based approach:
int_signifyd:app_storefront_controllers:app_storefront_core...

MetaData

- metadata.xml
- services.xml
- jobs.xml

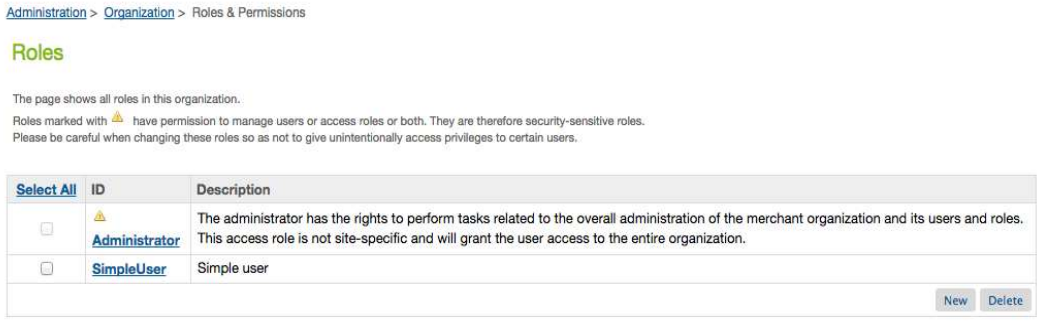
3. Component Overview

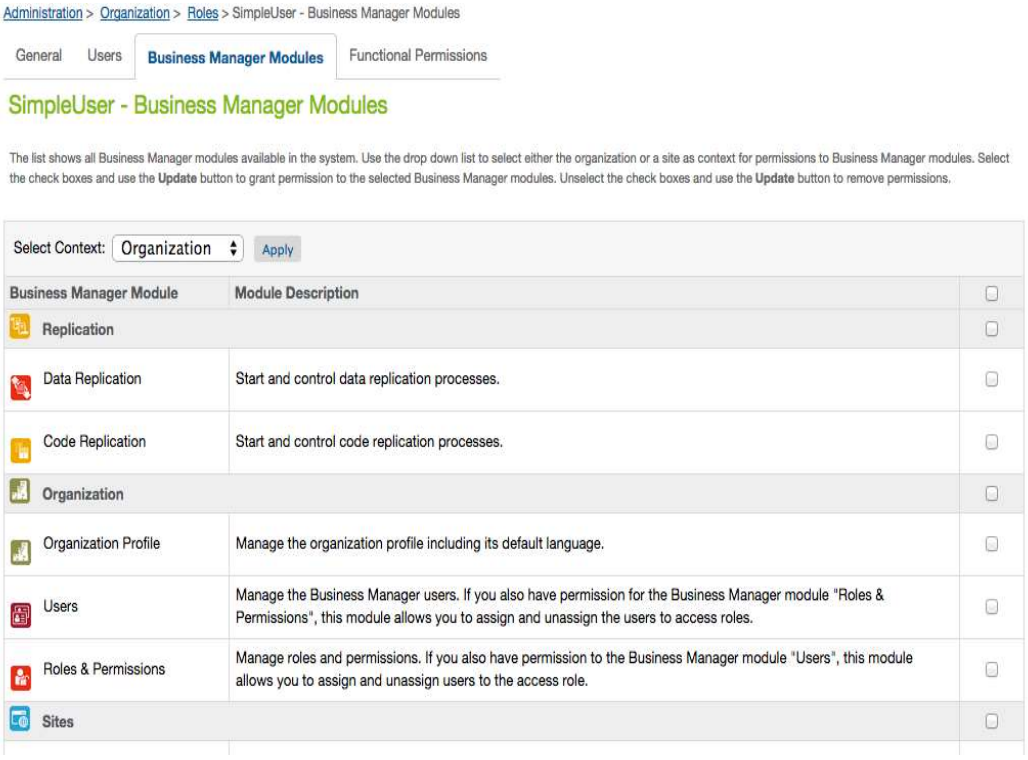
3.1 Functional Overview & Integration Guide

3.1.1 *Setup access to the Site Preference*

All permissions for customers can be set in Administration →Organization →Role & Permissions. You can allow admin level users to edit Site settings and disallow non-admin users.

You may need to make changes to this in order to enable or disable access to the required Signifyd site preferences.





3.1.2 *Setup Eclipse*

Steps for Loading the Cartridge in Eclipse

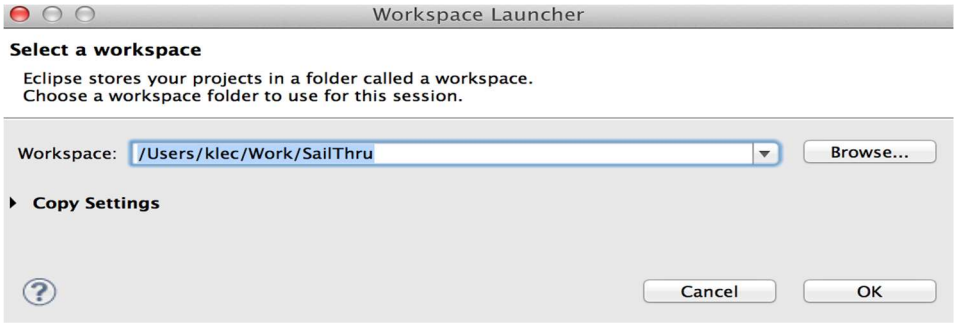
A workspace is an Eclipse-specific local directory that contains Eclipse projects. Normally Eclipse projects are connected to Java source directories (packages). In Demandware Studio projects are different: they either define a connection to a Salesforce Commerce Cloud instance or they point to a Salesforce Commerce Cloud cartridge. They are never used to compile Java projects since Java code is not used in Salesforce Commerce Cloud application programming.

Each workspace should have only 1 Salesforce Commerce Cloud server connection (covered later in this module). For example, if you are a developer working on numerous client projects, you will want to create a separate workspace for each client. Each client workspace will then have only 1 specific server connection.

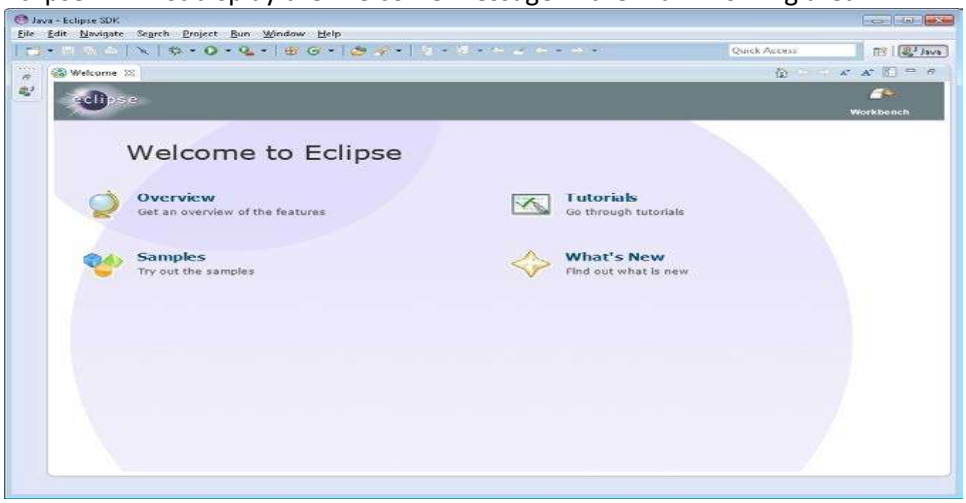
Run the Create a Workspace activity.

To install the UX Studio plugin into Eclipse and to create a new workspace (when using UX Studio for the first time), follow these steps:

1. The first time you use the application, you will be prompted to create a new workspace name. Give your workspace a name that references the client you are working with.



2. Eclipse will first display the Welcome message in the main working area.



Creating a Server Connection

In order to upload your code to a Salesforce Commerce Cloud server, you will need to create a server connection in UX Studio. A server connection allows you to push your code to the server instance but you will not be able to pull the code onto your personal computer from the Salesforce Commerce Cloud server. The connection is a 1-way push only.

Create a new server Connection

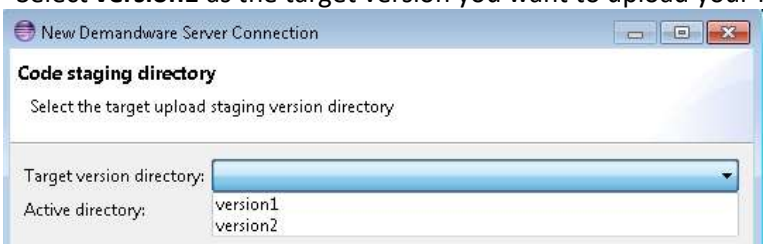
1. From UX Studio, click **Fil**e->**Ne**w->**Digital Server Server Connection**. The new server connection box opens.

2. Complete it as follows.

   In the **Project name** and **Host name** fields, use the host name provided by your client:
   e. g. **https://signifyd01-tech-prtnr-na05-dw.demandware.net/**

   Enter your password. Check the **Remember Password** flag.

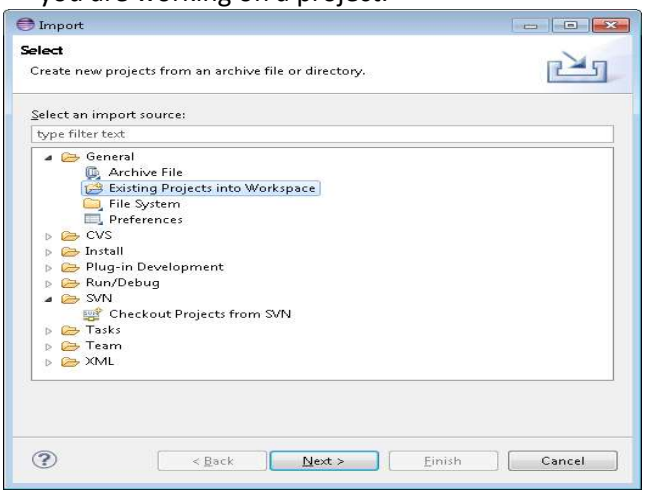
3. Click **Next**.
4. A security warning regarding an invalid certificate for your sandbox shows up. Click **Yes** to continue.
   Select **version1** as the target version you want to upload your files to:



5. Click **Finish**.
   Your connection project is now connected to your sandbox and will be used to upload any cartridge projects to that sandbox, as seen later in this module.

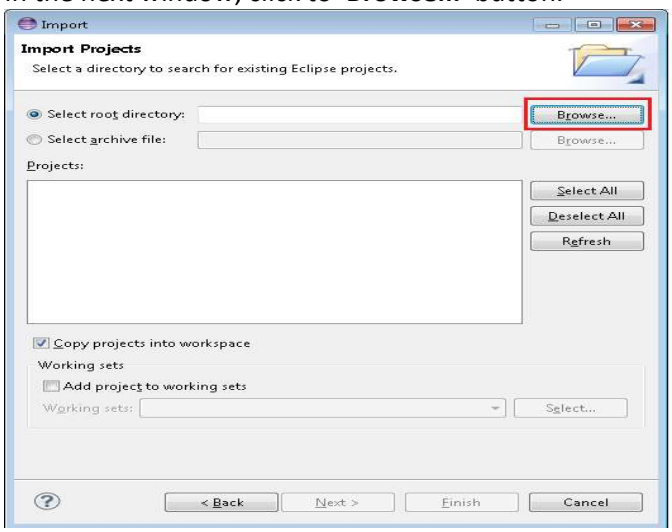
Import a project in Studio

1. From within UX Studio, click on **Fil**e->**Import...** an import window will open.
2. From the Import window, click to expand the **General** menu.
3. Click the **Existing Projects into Workspace** option. If you have an SVN server, you could projects directly from a repository, which is the most common way to obtain cartridges when you are working on a project.



4. Click '**Next**'.

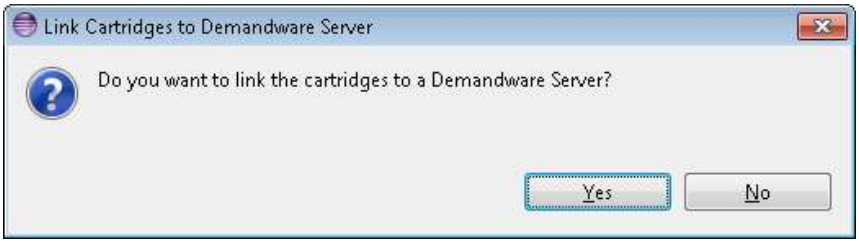
5. In the next window, click to '**Browse...**' button.



6. Locate the folder on your hard drive where cartridges are located. Your instructor will provide a zip file with all solution cartridges for you to install locally. Click **OK**.

7. Any cartridges in the folder structure (including subfolders) will be displayed in the **Projects** box. Click **Select All:**

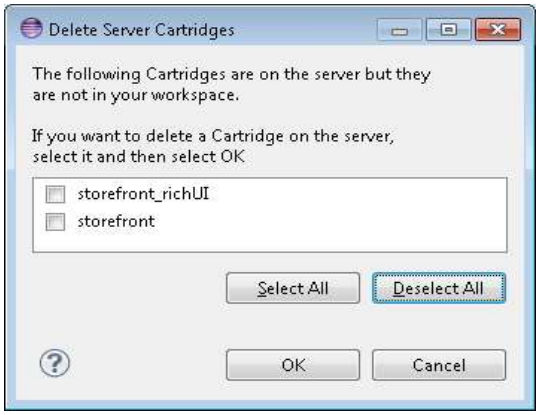
8. Click '**Finish**'.



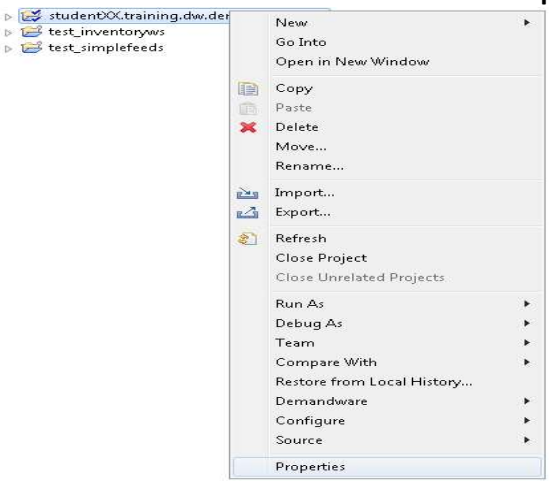
9. The next dialog allows you to select the specific cartridges you want uploaded to your server connection. Click **Select All.**

10. Click **OK** to upload the cartridges and finish the import process.

11. You might receive a dialog stating to delete projects on the server not matching the ones in your workspace. If you're the only one working on that instance e.g. it's your personal sandbox you might recognize the projects there.



12. If you import cartridges before you have an active server connection or somehow forgotten to link a cartridge to the server, do the following to ensure that cartridges will get uploaded correctly: right-click the server connection and select **Properties:**

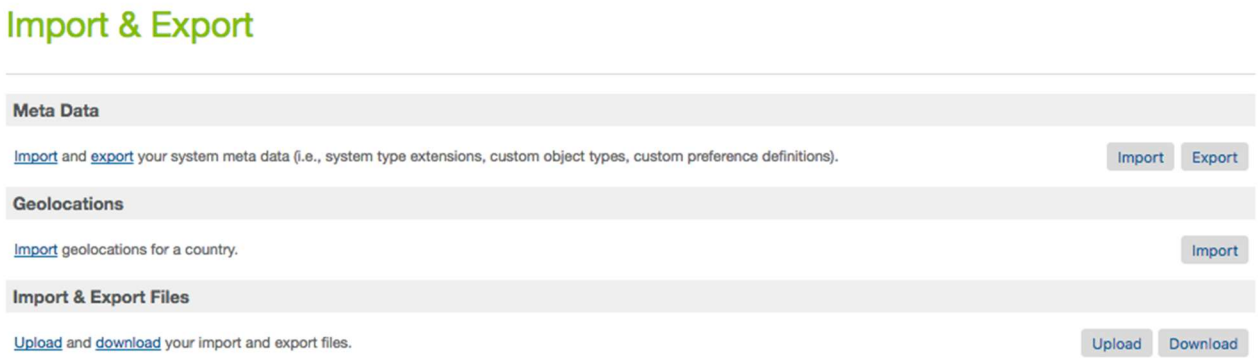


13. Select **Project References** and then select every cartridge that you want uploaded to the server connection and Click ok.

Configuration - Metadata import

First step is to import system object definitions for the Signifyd attributes for Order and Site Preferences. These are provided with cartridge in metadata.xml file

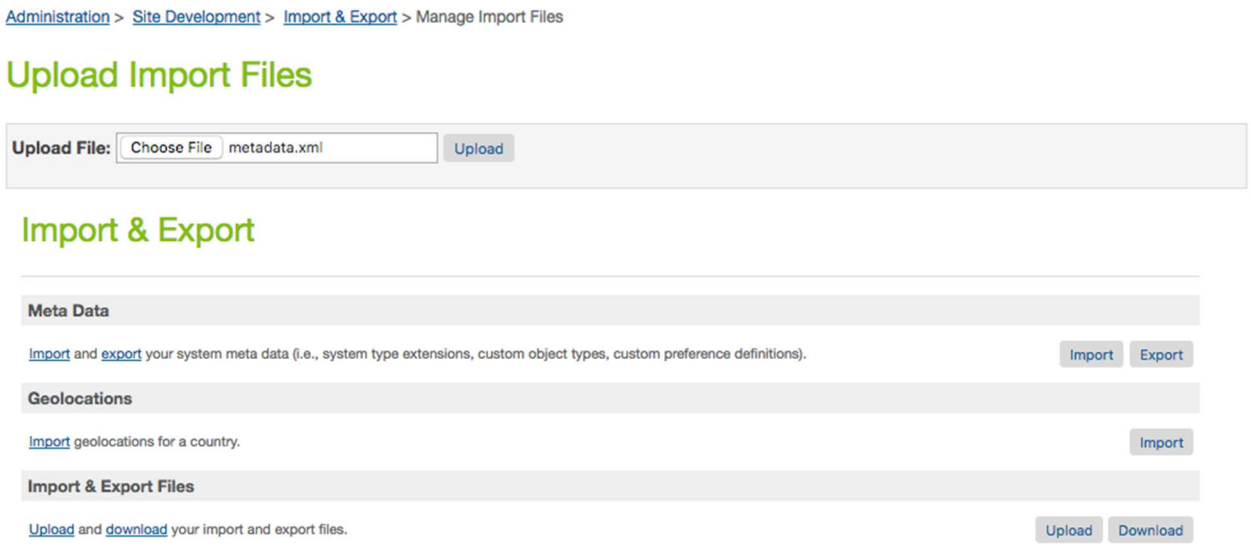
Upload this file via Business manager into your site:□1. Click on button "Upload" in Administration > Site Development > Import & Export



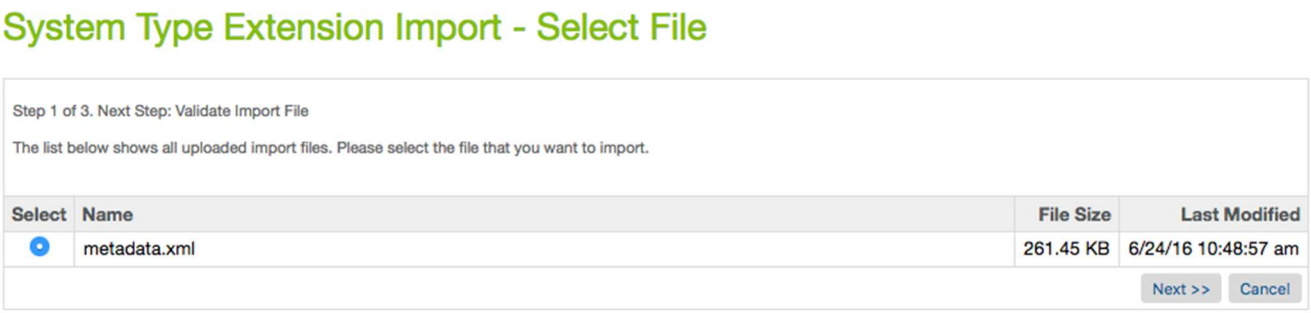
2. Choose your local file and again click "Upload"

3. And Click back button to return to Import page.

4. On the Meta Data section click on the 'Import' button

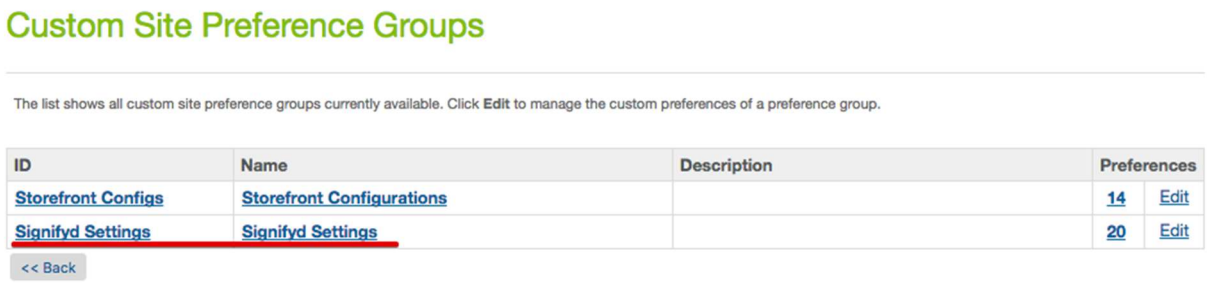


5. Select the metadata.xml file that you just uploaded and click 'next' to go through import process.

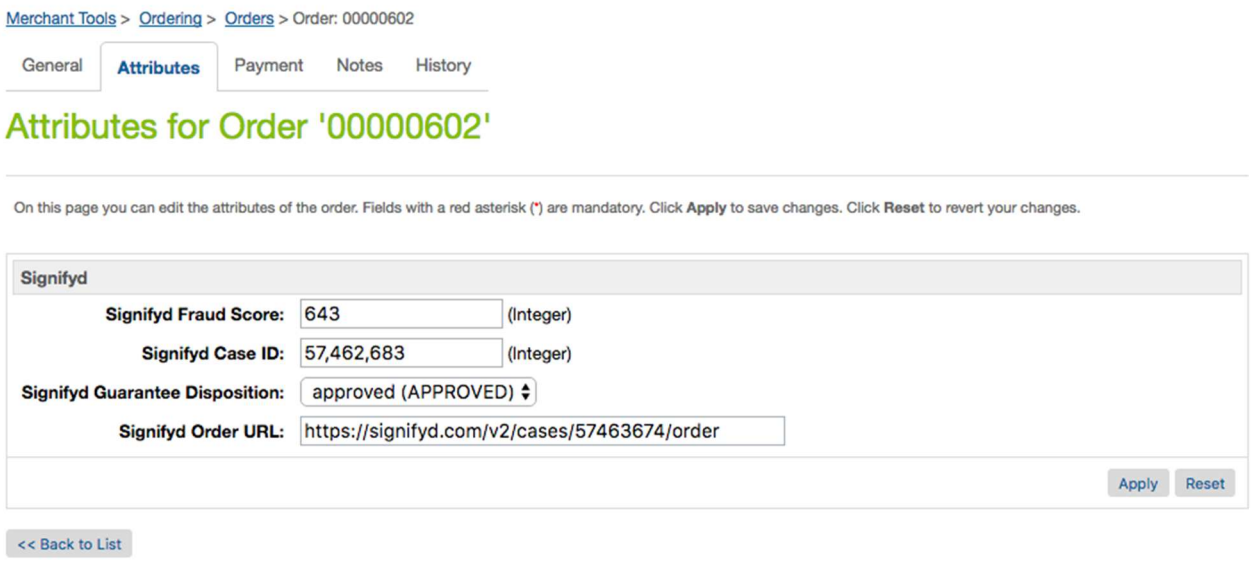


After import – Preference Entry

You will now see a 'Signifyd Settings' attribute group in the site preference section. Merchant Tools > Site Preferences > Custom Site Preferences:



You will now also be able to see the Signifyd attribute under an order by clicking on the 'Attributes' tab



Configuration – Service Framework Setup Import

Import the base Signifyd Service Framework configuration required by the Signifyd cartridge.

Step 1

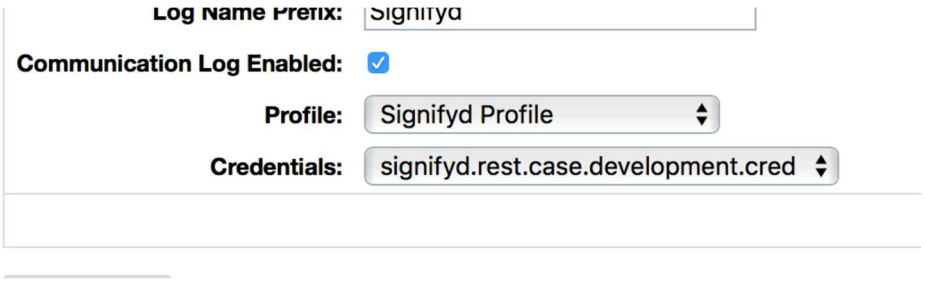
In Business Manager go to Administration->Operations->Import Export and select 'upload' then browse to the *services.xml* file that is located in the meta folder included with the Signifyd cartridge.

Step 2

Once the services file is uploaded click Import and chose *merge* to import the default Service Framework configuration.

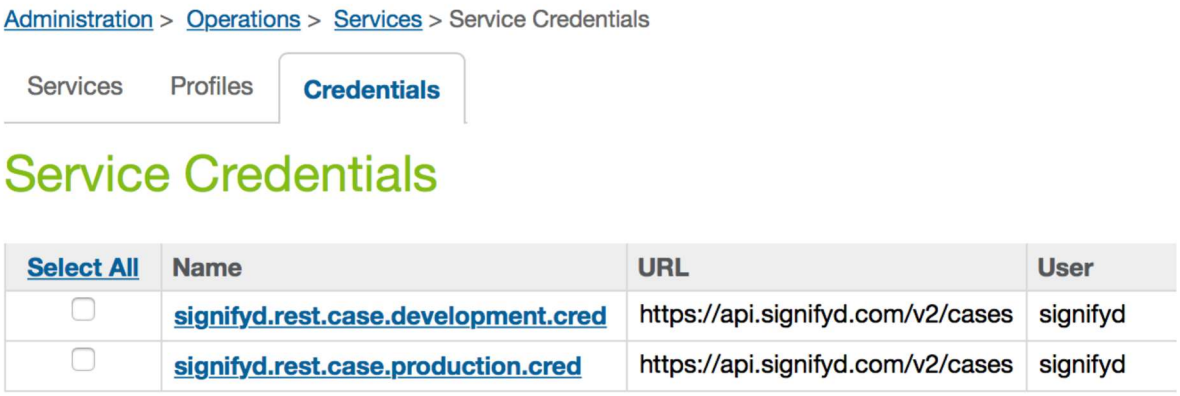
Step 3

Once imported you will need to navigate to the Signifyd service configuration and make sure the credential set that is being used aligns with the correct SFCC instance (i.e. development credentials on sandboxes).



Step 4

Navigate to the *credentials* tab and click to edit the credential sets. Making sure to enter the information provided by Signifyd for your particular implementation.



3.1.5 *Setup Job Schedules Configuration*

Configuration – Job Schedules Setup Import

Import the base Signifyd Job Schedules configuration required by the Signifyd cartridge.

Step 1

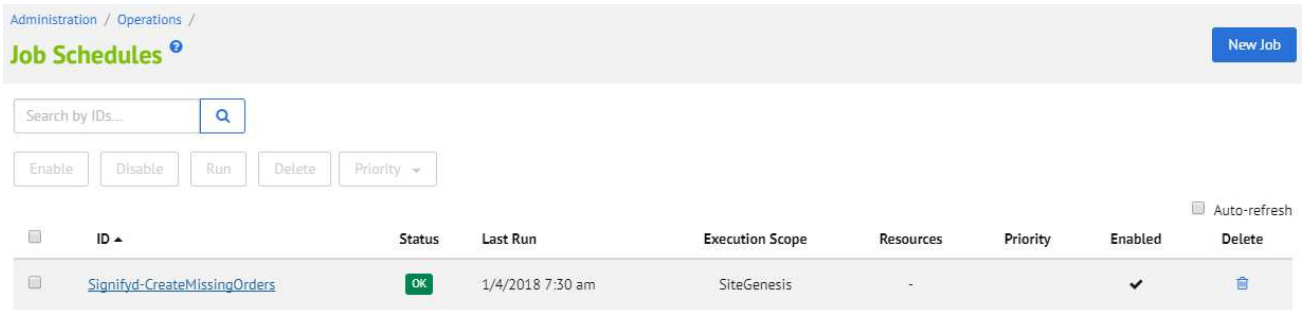
In Business Manager go to Administration->Operations->Import Export and select ‘upload’ then browse to the *jobs.xml* file that is located in the meta folder included with the Signifyd cartridge.

Step 2

Once the jobs file is uploaded click Import and chose *replace* to import the default Job Schedules configuration.

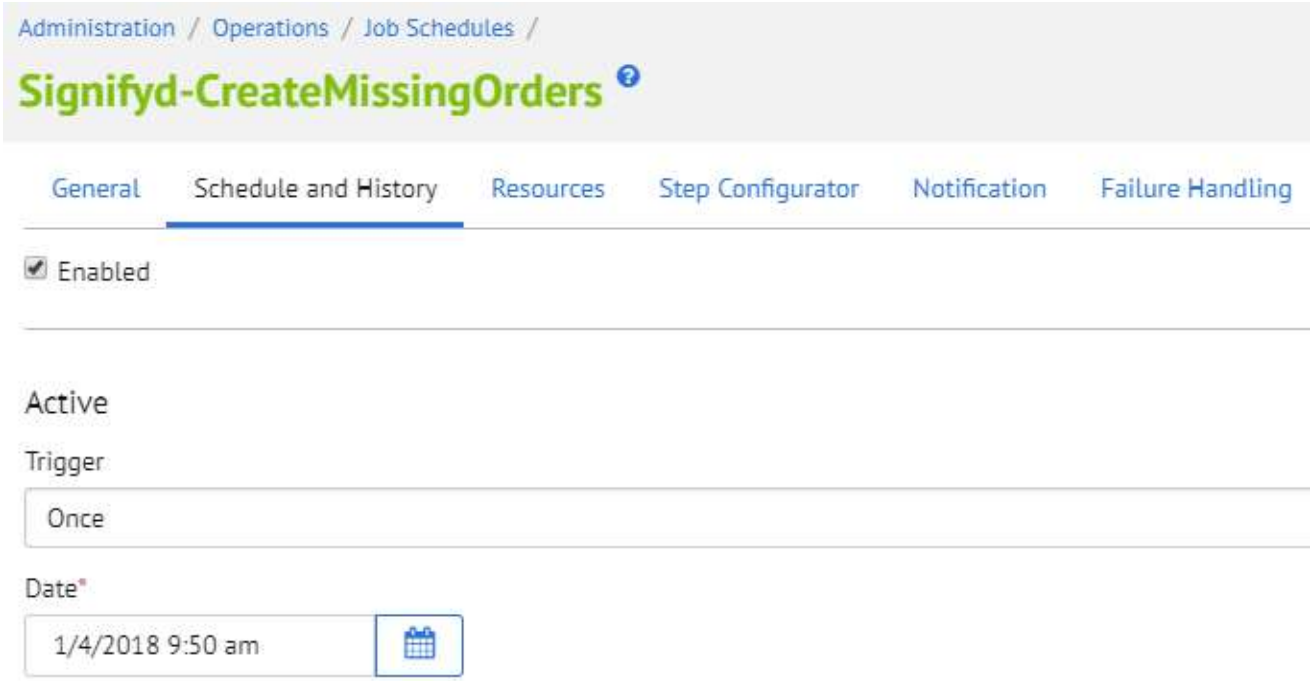
Step 3

In Business Manager go to Administration->Operations->Job Schedules. The Signifyd-CreateMissingOrders job will be displayed:



Step 4

Select Signifyd-CreateMissingOrders to enter the Job Schedule configuration. Configure your Job Schedule to run once, daily, or on any desired schedule:



3.1.6 *API Integration - Controllers*

Modifications in System Controller

Enabling SFCC to send requests to Signifyd requires a modification to the system controller file: *controllers/COSummary.js*

1. Add these two rows in the beginning of the *submit()* method:

```
var Signifyd = require('int_signifyd/cartridge/scripts/service/signifyd');
var orderSessionID = Signifyd.getOrderSessionId();
```

2. Add those two rows near the end of the same function method:

```
Signifyd.setOrderSessionId(placeOrderResult.Order, orderSessionID);
Signifyd.Call(placeOrderResult.Order);
```

Those two last lines (number 2) should go within last 'else if' statement.

```
else if (placeOrderResult.order_created) {
….<Insert code here>…..
}
```

The final code should look something like this:

```
function submit() {
    // Calls the COPlaceOrder controller that does the place order action and
    // COPlaceOrder returns a JSON object with an order_created key and a bool
    // If the order creation failed, it returns a JSON object with an error ke

    /* Signifyd Modification Start */
    var Signifyd = require('int_signifyd/cartridge/scripts/service/signifyd');
    var orderSessionID = Signifyd.getOrderSessionId();
    /* Signifyd Modification End */

    var placeOrderResult = app.getController('COPlaceOrder').Start();
    if (placeOrderResult.error) {
        start({
            PlaceOrderError: placeOrderResult.PlaceOrderError
        });
    } else if (placeOrderResult.order_created) {
        /* Signifyd Modification Start */
        Signifyd.setOrderSessionId(placeOrderResult.Order, orderSessionID);
        Signifyd.Call(placeOrderResult.Order);
        /* Signifyd Modification End */

        showConfirmation(placeOrderResult.Order);
    }
}
```

3.1.7 *API Integration - Templates*

Modifications to Core Template

In order to insert the fingerprint javascript snippet in the HTML <head> element, we have to modify the template *default/components/header/htmlhead.isml*.

Add the following lines in the end of the file (around line 78):

```
<isif
condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('Signify
dEnableCartridge')}">
  <isinclude template="signifyd_device_fingerprint" />
</isif>
```

The result should look like the following:

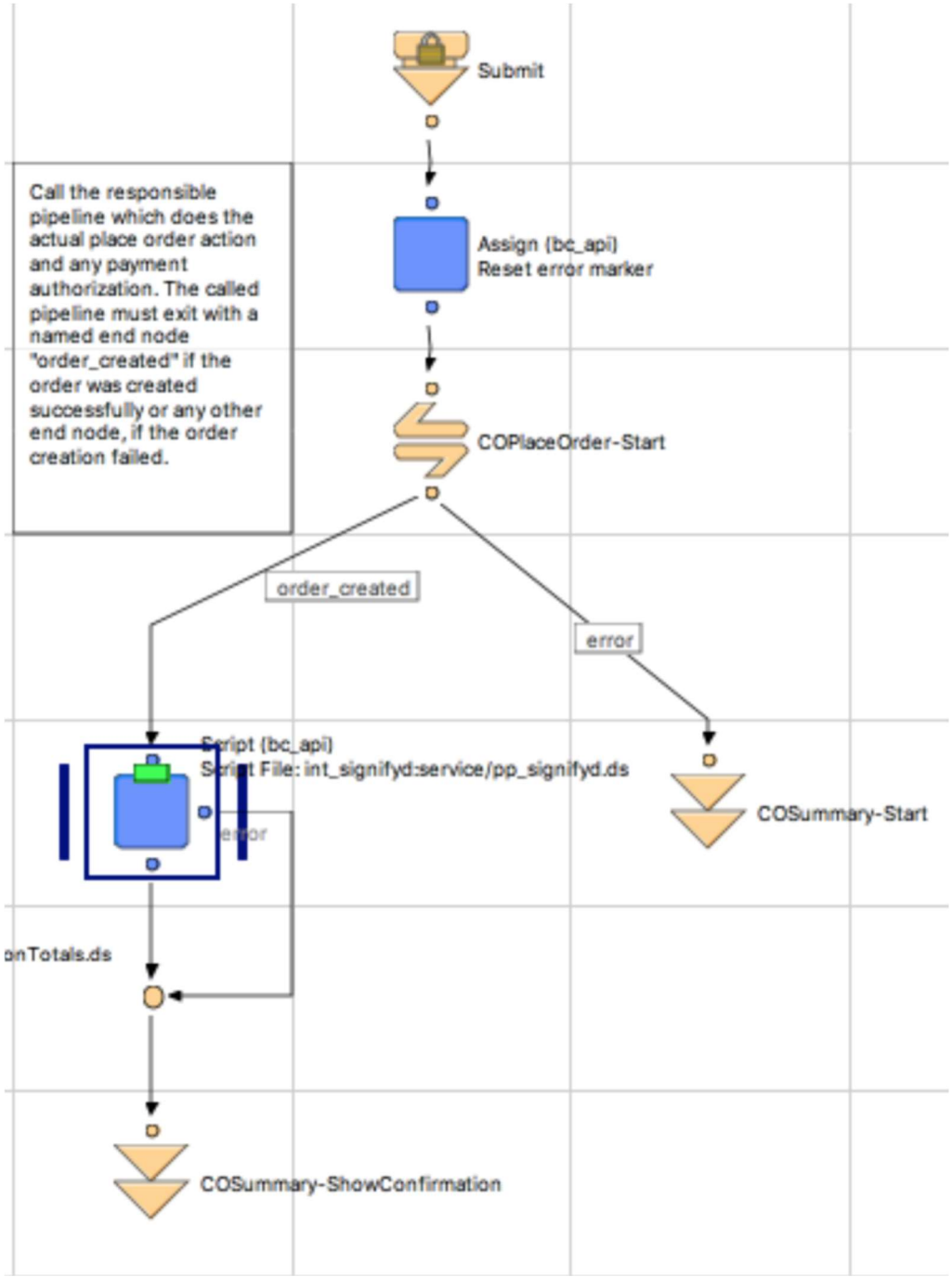
```
74
75  <iscomment>Gather device-aware scripts</iscomment>
76  <isinclude url="${URLUtils.url('Home-SetLayout')}"/>
77
78  <!-- Signifyd Modification Start -->
79  <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('SignifydEnableCartridge')}">
80    <isinclude template="signifyd_device_fingerprint" />
81  </isif>
82  <!-- Signifyd Modification End -->
83
```

Modifications to System Pipeline

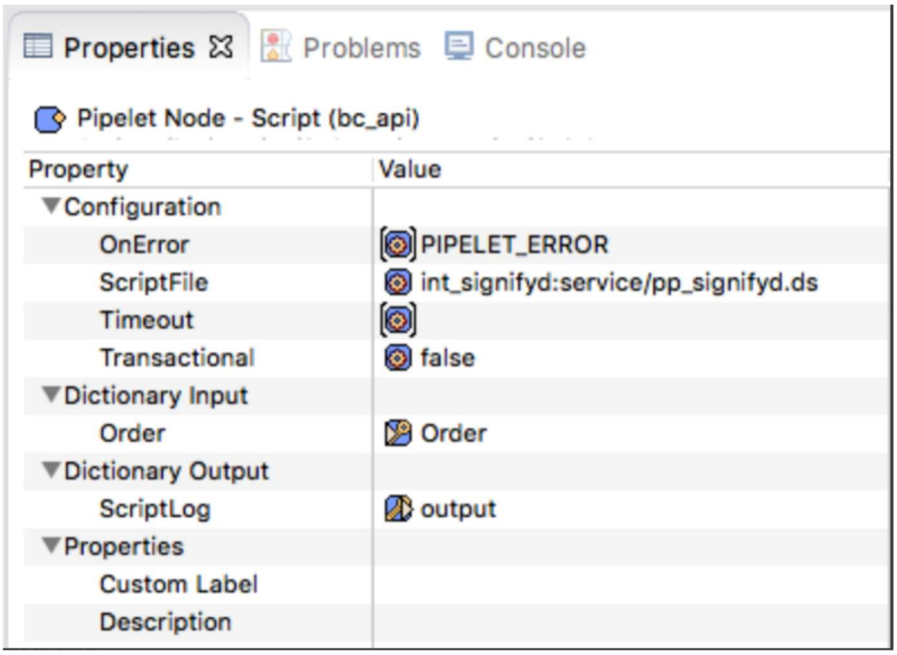
Another way to make SFCC send requests to Signifyd is modification to system pipeline for pipeline based site implementations. Since the pipeline based approach could eventually be deprecated by Salesforce Commerce Cloud, the controller based approach is recommended

Pipeline name is*: pipelines/ COSummary.xml*

Add a script *pp_signifyd.ds* to the end of pipeline Submit as this shown on below image:



Input parameter for this script must be a current Order.


Properties
Problems
Console
Pipelet Node - Script (bc_api)
Property
Value
Configuration
OnError
PIPELET_ERROR
ScriptFile
int_signifyd:service/pp_signifyd.ds
Timeout
Transactional
false
Dictionary Input
Order
Order
Dictionary Output
ScriptLog
output
Properties
Custom Label
Description

3.1.9 *API Integration - Customized order information*

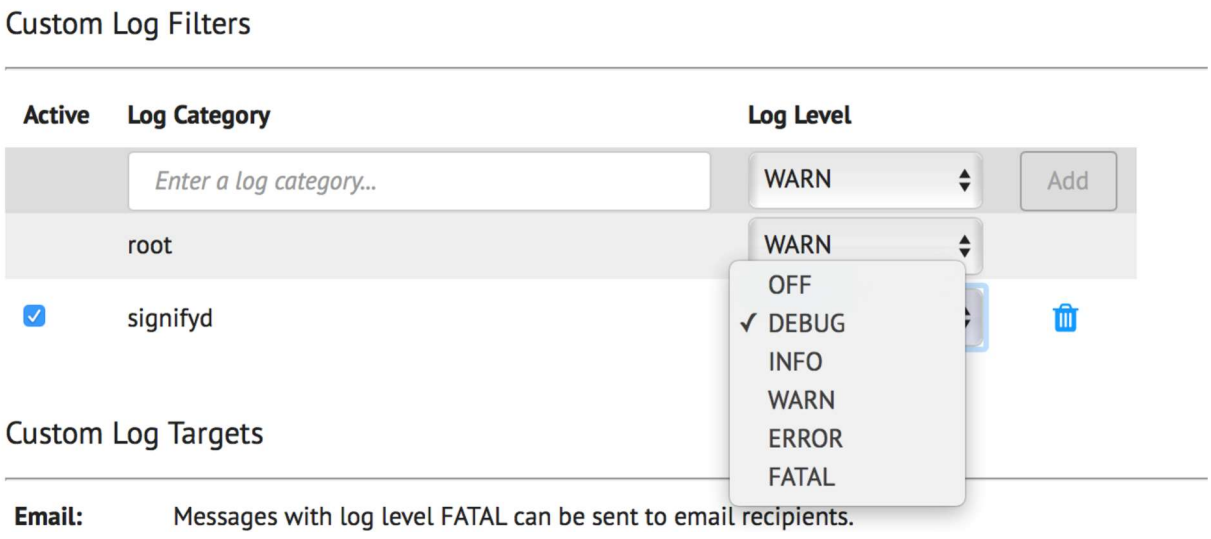
The Signifyd fraud service relies on information that is in part passed back from the payment gateway.
If a custom payment gateway is implemented, make sure to pass required information to Signifyd by modifying the *signifyd.ds* file as shown below:

```
135 * Converts an Order into JSON format
136 * acceptable on Stringifyd side
137 *
138 * @param {order} - Order that just have been placed.
139 * @return {result} - json objects describes Order.
140 */
141 function getParams(order: Order) {
142     var cal : Calendar = new Calendar(order.creationDate);
143     return {
144           purchase: {
145             "browserIpAddress": order.remoteHost,
146             "orderId": order.currentOrderNo,
147             "createdAt": StringUtils.formatCalendar(cal, "yyyy-MM-dd'T'HH:mm:ssZ"),
148             "paymentGateway": order.paymentTransaction.paymentProcessor.ID,
149             "paymentMethod": order.paymentTransaction.paymentInstrument.paymentMethod,
150             "transactionId": order.paymentTransaction.transactionID,
151             "currency": order.paymentTransaction.amount.currencyCode,
152             "avsResponseCode": "Y",
153             "cvvResponseCode": "M",
154             "orderChannel": "WEB",
155             "totalPrice": order.getTotalGrossPrice().value,
156             "products": getProducts(order.productLineItems),
157             "shipments": getShipments(order.shipments),
158           },
159           recipient: getRecipient(order.shipments[0], order.customerEmail),
160           card: {
161             "cardHolderName": order.paymentTransaction.paymentInstrument.creditCardHolder,
162             "bin": "",
163             "last4": order.paymentTransaction.paymentInstrument.creditCardNumberLastDigits,
164             "expiryMonth": order.paymentTransaction.paymentInstrument.creditCardExpirationMonth,
165             "expiryYear": order.paymentTransaction.paymentInstrument.creditCardExpirationYear,
166             "billingAddress": {
167               "streetAddress": order.billingAddress.address1,
168               "unit": order.billingAddress.address2,
169               "city": order.billingAddress.city,
```

3.2 Other Non-Transactional Operations

You can enable logs for all operations with the Signifyd API for debugging. But don't forget to disable it after debugging to prevent uncontrolled growth of log files.

Go to Administration –>Operation –>Custom Log Settings. You can enable specific levels of logging for Signifyd. Each level brings a different or higher level of detail in the logs

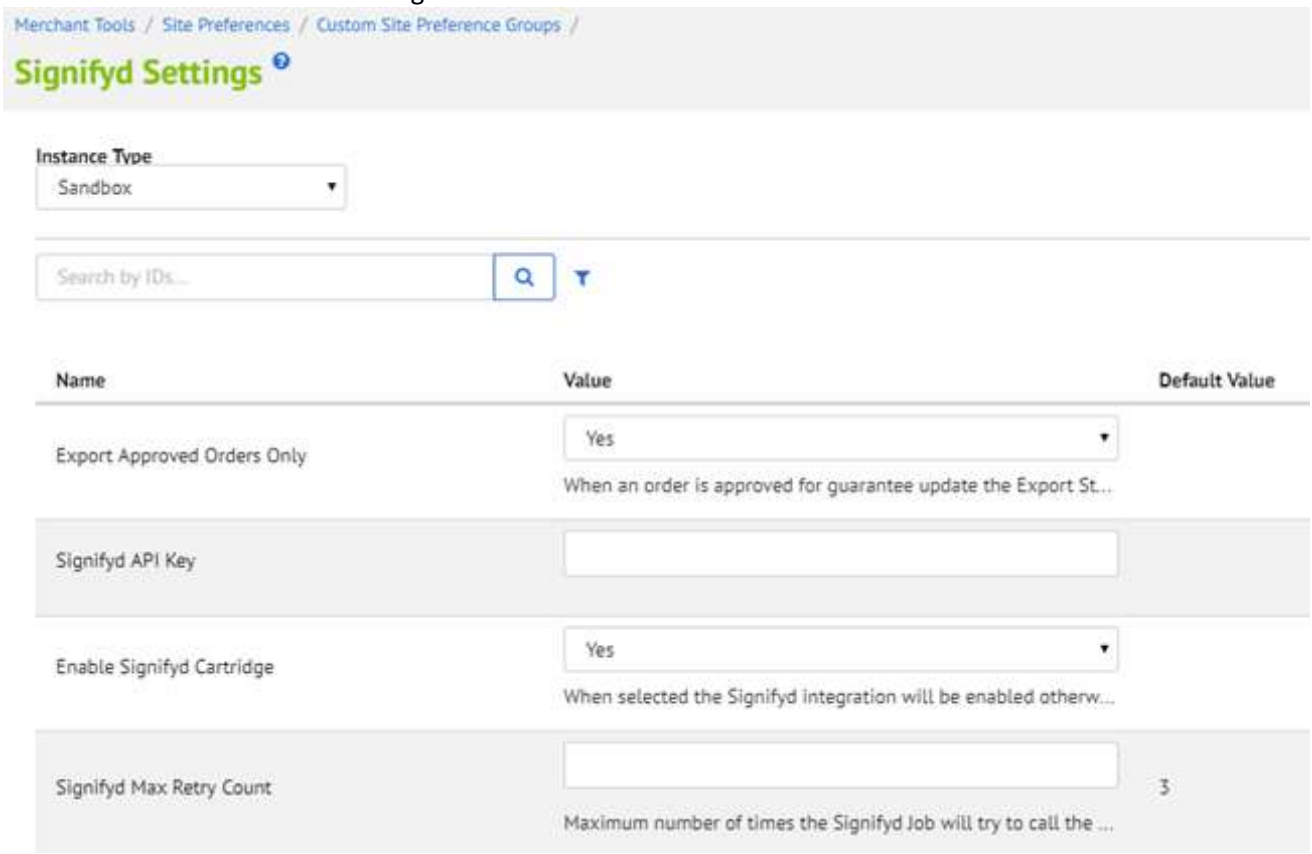


4. Configuration Guide

4.1 Setup

The Signifyd cartridge has a configuration setting to hold the order or immediately export depending on the fraud status.
The site preference setting is called ***SignifydHoldOrderEnable*** and if set to true this indicates that the Salesforce Commerce Cloud order is held until the webhook listener is called and indicate that the order is approved. This is accomplished by marking order as 'Not Exported' to prevent it from exporting until the webhook listener updates the order to 'Ready for Export'.

See below for screenshot of setting:



4.1.1 *Configuration on Signifyd side*

All Signifyd cases created during SFCC order creation can be viewed here:
https://app.signifyd.com/cases

API key, Profile and all other settings can be set up on this page:
https://app.signifyd.com/settings

4.2 External Interfaces List

>

Action	API Call	Method
Case Creation	https://www.signifyd.com/docs/api/#/reference/cases/create-a-case	POST
Guarantee submission	https://www.signifyd.com/docs/api/#/reference/guarantees	POST
Webhook interface	https://www.signifyd.com/docs/api/#/reference/webhooks	POST

4.3 Testing

Test Case: Order Placement with Site Preference 'Signifyd Hold Order' Set to "Yes"

Status: Order Approved

Expected Result:

1. A case will be created on Signifyd site at https://app.signifyd.com/cases. The case ID will be saved on the order.
2. The order details will be sent from SFCC to Signifyd and should be visible in the Signifyd Admin Panel.
3. Order Status will be 'Open' in SFCC
4. Once webhook marks the order data 'guaranteeDisposition: "APPROVED"' the order will be updated to 'ready for export' in SFCC.
5. The following order attributes will be updated:
   'signifydFraudScore'
   'signifydGaurenteeDisposition'
   'signifydOrderURL'

Status: Order Declined

Expected Result:

1. A case will be created on Signifyd site at https://app.signifyd.com/cases. The case ID will be saved on the order.
2. The order details will be sent from SFCC to Signifyd and should be visible in the Signifyd Admin Panel.
3. Order Status will be 'Open' in SFCC
4. Once webhook marks the order data 'guaranteeDisposition: "DECLINED"' the order will stay on 'Open' status in SFCC.
5. The following order attributes will be updated:

'signifydFraudScore'
'signifydGaurenteeDisposition'
'signifydOrderURL'

Test Case: Order Placement with Site Preference 'Signifyd Hold Order' Set to "No"

Status: Order Approved/Order Declined

Expected Result:

1. A case will be created on Signifyd site at https://app.signifyd.com/cases. The case ID will be saved on the order.
2. The order details will be sent from SFCC to Signifyd and should be visible in the Signifyd Admin Panel.
3. Order Status will be 'Ready To Export' in SFCC
4. The following order attributes will be updated:
   'signifydFraudScore'
   'signifydGaurenteeDisposition'
   'signifydOrderURL'

Test Case: Retry job

#:	Step actions:	Expected Results:
1	BM>Merchant Tools>Site Preferences>Custom Site Preference Groups>Signifyd Settings - Set "Enable Signifyd Cartridge:" to **Yes**. - Enter **invalid** API key	The configuration should be saved.
2	Place an order.	The order should be placed.
3	BM>Merchant Tools>Ordering>Orders Check order attributes	Signifyd Case ID for this order is empty
4	BM>Merchant Tools>Site Preferences>Custom Site Preference Groups > Signifyd Settings - Enter **valid** API key	The configuration should be saved.
5	BM>Administration>Operations>JobSchedules Run Signifyd-CreateMissingOrders.	The job should run and a Case ID should be generated for the order.
6	BM > Merchant Tools > Ordering > Orders Check order attributes	The Signifyd Case ID was set

5. Operations, Maintenance

5.1 Availability

Availability/Uptime is 24/7

5.2 Support

Contact Signifyd support at https://www.signifyd.com/resources/faq/

6. Release History

Version	Date	Changes
16.1.0	08/15/2015	Initial release
18.1.0	04/01/2018	Added Job Schedule configuration and template modification