**Signifyd INTEGRATION**

Version 3.4.0



Table of Contents

[1. Intended Audience 3](#_Toc203403263)

[2. Summary 3](#_Toc203403264)

[3. Components 4](#_Toc203403265)

[3.1 int\_signifyd 4](#_Toc203403266)

[3.2 int\_signifyd\_sfra 5](#_Toc203403267)

[3.3 signifyd\_sfra\_changes 6](#_Toc203403268)

[3.4 Metadata 6](#_Toc203403269)

[3.4.1 System Object Types 6](#_Toc203403270)

[4. Functional Overview 8](#_Toc203403271)

[4.1 Features 8](#_Toc203403272)

[4.1.1 Post-authorization 8](#_Toc203403273)

[4.1.2 Pre-authorization 8](#_Toc203403274)

[4.1.3 Passive Mode 9](#_Toc203403275)

[4.1.4 Post-Auth Fallback 9](#_Toc203403276)

[4.1.5 Order Fulfillment 9](#_Toc203403277)

[4.1.6 Reroute 10](#_Toc203403278)

[4.1.6 Custom Preferences 10](#_Toc203403279)

[4.1.7 Manual Review 10](#_Toc203403280)

[4.1.8 Webhooks 12](#_Toc203403281)

[4.1.9 Payment Exclusion 12](#_Toc203403282)

[4.1.10 Create Missing Orders Job 12](#_Toc203403283)

[5. Integration 13](#_Toc203403284)

[5.1 Metadata Import 13](#_Toc203403285)

[5.1.1 System Object Type Extensions 13](#_Toc203403286)

[5.1.2 Services 15](#_Toc203403287)

[5.1.3 Jobs ………………………………………………………………………………………………………..16](#_Toc203403288)

[5.2 SFRA Integration 17](#_Toc203403289)

[5.2.1 Script changes 17](#_Toc203403290)

[5.2.2 Adding the necessary changes 17](#_Toc203403291)

[5.3 SiteGenesis Integration – Controllers 18](#_Toc203403292)

[5.3.1 Modifications in System Controller 18](#_Toc203403293)

[5.3.2 Modifications to Core Template – Device Fingerprinting 19](#_Toc203403294)

[5.4 SiteGenesis Integration – Pipelines 20](#_Toc203403295)

[5.5 Limitations and Constraints 21](#_Toc203403296)

[5.5.1 AVS and CVV Response Codes 21](#_Toc203403297)

[5.5.2 Order Channel 22](#_Toc203403298)

[5.5.3 Fulfillment API 23](#_Toc203403299)

[5.5.4 Pre-auth default error message 23](#_Toc203403300)

[5.5.6 Digital items 24](#_Toc203403301)

[5.5.7 Vanity hostnames 24](#_Toc203403302)

[5.6 Updating from V2 to V3 25](#_Toc203403303)

[5.7 Logging 26](#_Toc203403304)

[5.8 Configurations on Signifyd Console 26](#_Toc203403305)

[5.9 Testing 26](#_Toc203403306)

[5.9.1 Pre-auth flow 27](#_Toc203403307)

[5.9.2 Post-auth flow 28](#_Toc203403308)

[5.9.3 Automated Testing 28](#_Toc203403309)

[5.10 Troubleshoot 29](#_Toc203403310)

[5.10.1 Missing API Key 29](#_Toc203403311)

[5.10.2 Wrong API Key 29](#_Toc203403312)

[5.10.3 Service not enabled 29](#_Toc203403313)

[5.10.4 Order fields not being updated 29](#_Toc203403314)

[6 Operations, Maintenance 30](#_Toc203403315)

[6.1 Availability 30](#_Toc203403316)

[6.2 Support 30](#_Toc203403317)

[6.3 Intended Locales 30](#_Toc203403318)

[7 Release History 30](#_Toc203403319)

[8 Process Flow Diagrams 31](#_Toc203403320)

# 1. 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.

# 2. Summary

Signifyd is a fraud solution that provides a financial guarantee, allowing businesses to increase sales while reducing fraud losses. The Signifyd cartridge will be integrated into Salesforce Commerce Cloud using three primary API integration points:

1. Pre-payment authorization (Checkout API, Transaction API)
2. Post-payment authorization (Sale API, Webhooks)
3. Post order fulfilment (Fulfilment API)

Sale API is used if merchants are executing a "Post-Auth" Flow where they call into Signifyd after the order has been placed and the payment has been authorized with the Payment Gateway. Checkout API is used if merchants are executing a "Pre-Auth" Flow where they call into Signifyd before authorizing a payment with the Payment Gateway. In "Pre-Auth" flow, for each order, there are at least 2 API calls, a Checkout API call prior to payment authorization and a Transaction API after authorization

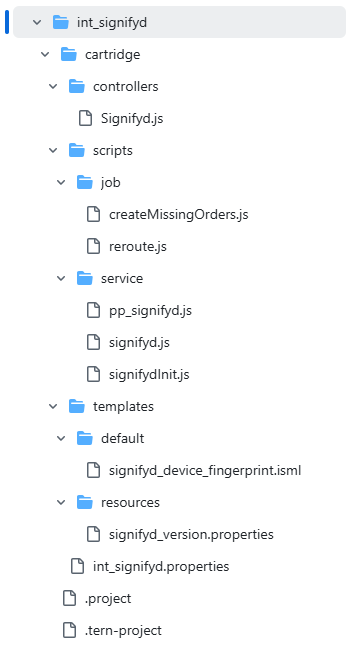
This document primarily serves as the LINK implementation guide for setting up Signifyd on SFRA and SiteGenesis.

The setup and custom code configuration described in this document assumes the use of SFRA version >= 3.3.0 and 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.

# 3. Components

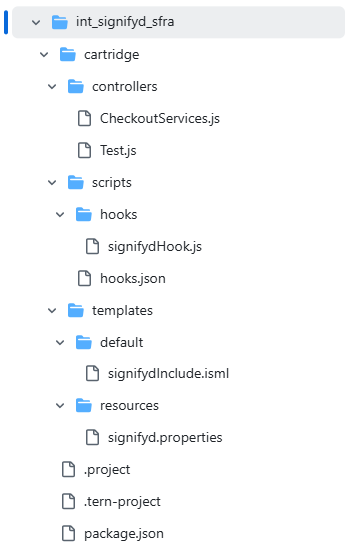
## 3.1 int\_signifyd

The **int\_signifyd** cartridge contains the base changes for the cartridge that are common for both SFRA and SiteGenesis.



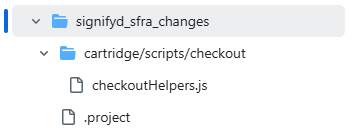
## 3.2 int\_signifyd\_sfra

The **int\_signifyd\_sfra** cartridge contains files specific to the SFRA integration.



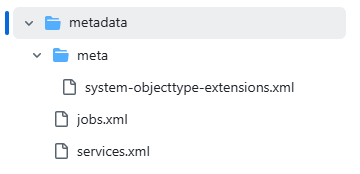
## 3.3 signifyd\_sfra\_changes

The **signifyd\_sfra\_changes** cartridgecontains examples of files that should be changed on the SFRA base file or customized file used by the merchant integration.



## 3.4 Metadata

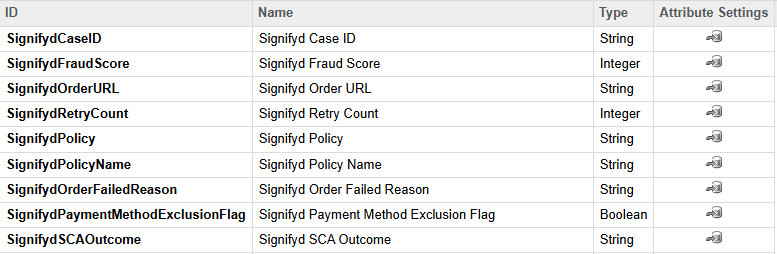
The cartridge metadata includes System Objects, such as Order and Site Preferences, along with the necessary jobs and services essential for the cartridge's functionality.



### 3.4.1 System Object Types

#### Order

The Order system object is updated with Signifyd related custom attributes, visible in the Signifyd custom attribute group.



#### Site Preferences

The Signifyd Custom Site Preferences are inside the group Signifyd Settings. The preferences are used to control the cartridge behavior.

Interface gráfica do usuário, Texto

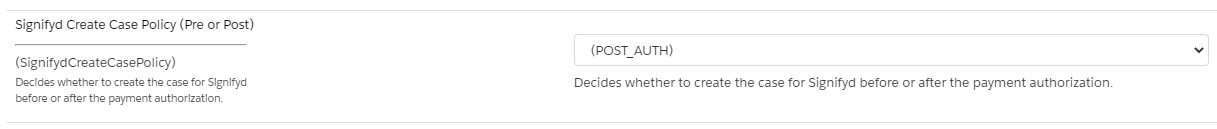
O conteúdo gerado por IA pode estar incorreto.

# 4. Functional Overview

## 4.1 Features

### 4.1.1 Post-authorization

If the custom site preference ***SignifydCreateCasePolicy*** is set to “POST\_AUTH”, Signifyd decisions are returned asynchronously, so an HTTP callback (webhook) is used to return their guarantee decision. Refer to section 7 - Process Flow Diagrams for post-auth process flow.



**Action 1.** The Signifyd Sale API is called after the Salesforce Commerce Cloud place order event, which happens after the order has gone through the authorization process against the payment provider right before displaying the order summary page. Because it is only called during the order creation, this will ensure that Create Case is never called again for that same order. The Sale API is called when all below actions are completed:

1. Payment is authorized without errors.
2. The order is successfully placed.
3. Confirmation of the order is sent to the customer.

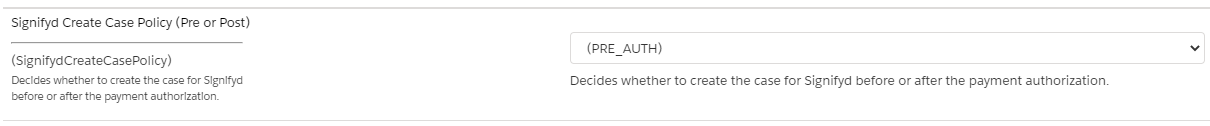
**Note**: Any customizations done by the merchant should guarantee that the Sale API is called after the above steps are completed.

**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 Signifyd has completed its fraud assessment, and a decision is made to either approve or decline the order for financial guarantee. This triggers an update to the order in SFCC and could also indicate that the order is ready to export (depending on settings).

Note: In the case that Signifyd returns a decline decision, the merchant should decide how to proceed with that order. If the order is manually cancelled in SFCC and/or with the payment processor, the communication to the client should also be done by the merchant, since the checkout experience on the storefront itself is complete at this stage.

### 4.1.2 Pre-authorization

If the custom site preference ***SignifydCreateCasePolicy*** is set to “PRE\_AUTH”, Signifyd decisions are returned synchronously after the relevant Signifyd API is called. Refer to section 7 - Process Flow Diagrams for pre-auth process flow.



**Action 1.** The Signifyd Checkout API is called before the Salesforce Commerce Cloud order has gone through the authorization process against the payment provider. If Signifyd returns a declined/rejected response, the authorization call will not happen, and the order will be failed. The failed order will be in “failed” status in Business Manager, where the merchant can also check the Signifyd order attributes, like the score received and the reason it failed. The storefront will display a default error message to the customer. If the merchant wants to customize this message, it can be changed directly in the code. If Signifyd returns an accept/approved response, Salesforce Commerce Cloud proceeds with the authorization process against the payment provider.

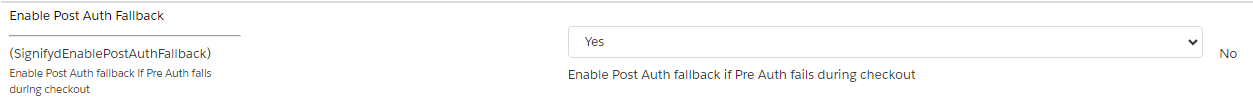
**Action 2.** For Signifyd "Approved" orders, the Transaction API is called to send payment status after the authorization result. Some of the fields should be manually added by the merchant for the Transaction API. More details in the section 3.1.11 API Integration – Limitations and Constraints.

### 4.1.3 Passive Mode

Passive mode can be used by setting the custom site preference ***SignifydPassiveMode*** to “Yes”. If passive mode is enabled, Signifyd decision will be visible on Business Manager through the order custom attributes, but it won’t impact the order status regardless of Signifyd accept/reject decisions. When the passive mode is switched off, Signifyd decision will be updated on the order and the order status will be impacted according to the accept/reject decisions.

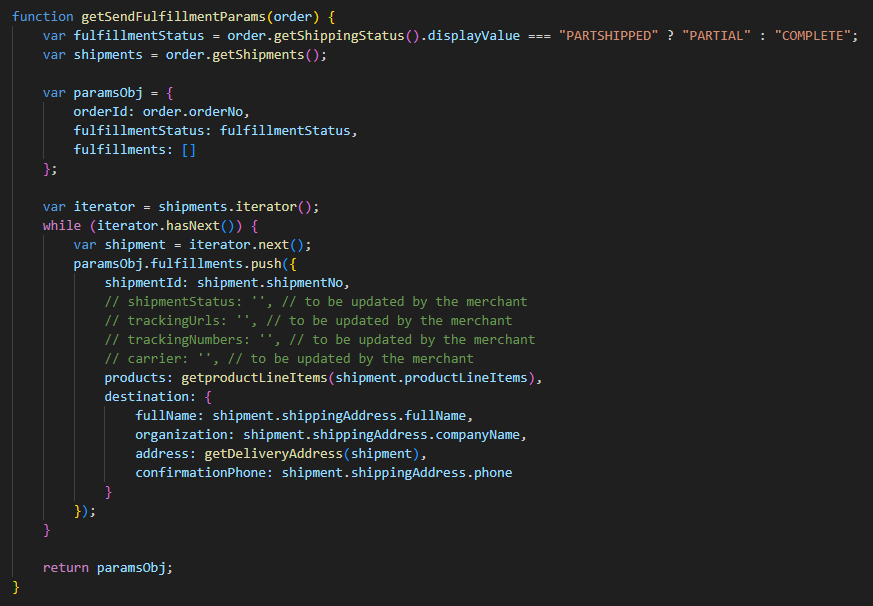
### 4.1.4 Post-Auth Fallback

When in Pre-Auth mode, the merchant can set the ***SignifydEnablePostAuthFallback*** preference to “Yes” to have a fallback Post-Auth call in case the original Pre-Auth call fails during checkout. By default, this preference is set to “No”.



### 4.1.5 Order Fulfillment

Fulfillment represents a shipment of one or more items in an order. The merchant can submit fulfillments details for orders that were shipped, even if the order is partially shipped. The function **sendFulfillment** from the file **signifyd.js** is available to be called at the time when the order is fulfilled on the merchant order flow. Some of the fields for the Fulfillment API need to be added manually by the merchant. More details in the section **5.5 Limitations and Constraints**.



### 4.1.6 Reroute

The **sendReroute** function is available to be called whenever the delivery address on an order needs to be changed. The change will be linked to the original order with the orderId parameter in the body. The **getSendRerouteParams** auxiliary function creates the request with the necessary information for the API call.



### 4.1.6 Custom Preferences

The Signifyd cartridge has a configuration setting to hold the order or immediately export depending on Signifyd’s guarantee decision. 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 indicates that the order is approved. This is accomplished by marking the order as 'Not Exported' to prevent it from exporting until the webhook listener updates the order to 'Ready for Export'.

The site preference **SignifydCreateCasePolicy** decides whether the create case policy will be pre-authorization (synchronous) or post-authorization (asynchronous).

Use the site preference **SignifydCoverageRequest** to setup the Coverage Request as FRAUD (default), NONE, INR or ALL.

### 4.1.7 Manual Review

To use Signifyd’s manual review feature for pre-auth, you would need to configure the ORDER\_CHECKPOINT\_ACTION\_UPDATE webhook. In this scenario, Signifyd decisions (for manual review orders) will be returned asynchronously, so an HTTP callback (webhook) is used to return the guarantee decision.

The expected order flow process for manual review orders in pre-auth setting is mentioned below:

1. Order is sent to Signifyd for pre-auth fraud check
2. Signifyd deems the order to be a manual review order and returns a PENDING decision response back to SFCC (instead of ACCEPT or REJECT)
3. SFCC sends the order for payment authorization
4. Once the payment authorization process is complete, SFCC sends the payment authorization details to Signifyd via the transaction API endpoint
5. Based on the payment authorization details sent, Signifyd makes a manual decision on the order (order moved from Pending to Accept/Reject)
6. ORDER\_CHECKPOINT\_ACTION\_UPDATE webhook is sent to SFCC for this asynchronous decision by Signifyd
7. SFCC consumes Signifyd’s ORDER\_CHECKPOINT\_ACTION\_UPDATE webhook and processes the order as per the defined merchant settings. Example:
8. For Signifyd approved orders, the order may be exported to merchant’s OMS
9. For Signifyd declined/rejected orders, the order may not be exported to merchant’s OMS

The table below lists out the different types of scenarios related to Manual Review orders that a merchant may want to consider.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scenario | | Signifyd decision for a Manual Review Order | Merchant Scenario | Action needed from Merchant | Who takes liability? |
| 1 | Accept | | Merchant decides to progress order to fulfillment | No action needed as this should happen automatically via ORDER\_CHECKPOINT\_ACTION\_UPDATE webhooks | Signifyd |
| 2 | Reject | | Merchant decides to fulfill Signifyd declined order | 1. Merchant user would need to mark the order as Good within Signifyd console 2. Manually progress order to OMS for order fulfillment | Merchant |
| 3 | Reject | | Merchant decides to not fulfill Signifyd declined order | - Merchant would need to reverse payment authorization for this order since the authorization was already done.  - In case the payment was captured prior to receiving Signifyd's decision, the payment would need to be refunded | N/A |
| 4 | Reject | | Merchant decides to resubmit order to Signifyd | - Merchant users would need to resubmit order from the console and add additional details related to why this declined order is being resubmitted.  - If Signifyd subsequently approves this order post resubmit, then order should be progressed to the OMS for fulfillment (this would happen automatically via webhooks)  - If Signifyd declines this order again post resubmit, then merchant can decide if they still want to progress the order (scenario 2) or not (scenario 3) |  |

### 4.1.8 Webhooks

Webhooks are sent by Signifyd when Signifyd approves/declines an order for guarantee and will be indicated in the checkpointAction string as “ACCEPT”, “REJECT”, or “HOLD". Typically, for pre-auth integrations, there is no need to configure a webhook as Signifyd’s decision will be synchronous.

However, if a merchant wants to use Signifyd’s manual review feature for pre-auth, then you would need to configure the ORDER\_CHECKPOINT\_ACTION\_UPDATE webhook. In this scenario, Signifyd decisions (for manual review orders) will be returned asynchronously, so an HTTP callback (webhook) is used to return the guarantee decision.

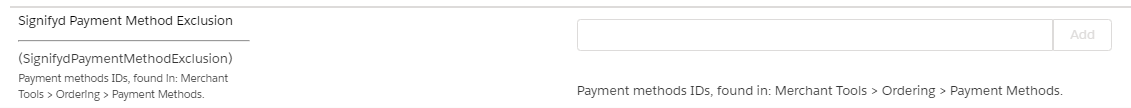
**Please note**: For all Signifyd APIs, “caseId”= “signifydId”, which is Signifyd's unique identifier for the order in our systems.

Instructions to setup V3 Webhooks are available on this page: <https://docs.signifyd.com/#operation/CreateTeamWebhook>

Once the Signifyd decision has been made, it will be listed under order attributes.

### 4.1.9 Payment Exclusion

Merchants can add payment methods to an exclusion list by adding the payment method ids in the custom preference **SignifydPaymentMethodExclusion.** The payment method id informed in the preference should be the same id visible in the Business Manager menu **Merchant Tools > Ordering > Payment Methods**.



Orders with an excluded payment method from the preference will be ignored during checkout and will not trigger a request to Signifyd. These orders will instead be directly sent for payment authorization. The boolean attribute **SignifydPaymentMethodExclusionFlag** in the order will indicate that this order was ignored by Signifyd.

### 4.1.10 Create Missing Orders Job

The **Signifyd-CreateMissingOrders** job can be scheduled by the merchant to run periodically and catch orders that do not have a case ID (SignifydCaseID attribute = null), have a retry count less than the maximum retry count configured in the custom preference **SignifydMaxRetryCount** (default is 3), and do not have the status FAILED or CANCELLED. Additionally, orders that were ignored due to the payment method being listed in the payment exclusion custom preference and have the attribute **SignifydPaymentMethodExclusionFlag** set as true, will not be processed by the job.

# 5. Integration

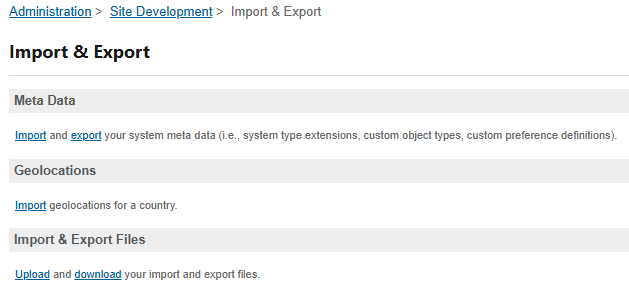
## 5.1 Metadata Import

### 5.1.1 System Object Type Extensions

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

Upload this file via Business Manager into your site:

1. Click on the Upload" button in Administration > Site Development > Import & Export



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

A screenshot of a computer

Description automatically generated

3. Click on the Back button to return to the Import page.

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

Interface gráfica do usuário, Aplicativo, Teams

O conteúdo gerado por IA pode estar incorreto.

5. Select the **system-objecttype-extensions.xml** file that you just uploaded and click 'next' to go through the import process.

A screenshot of a computer

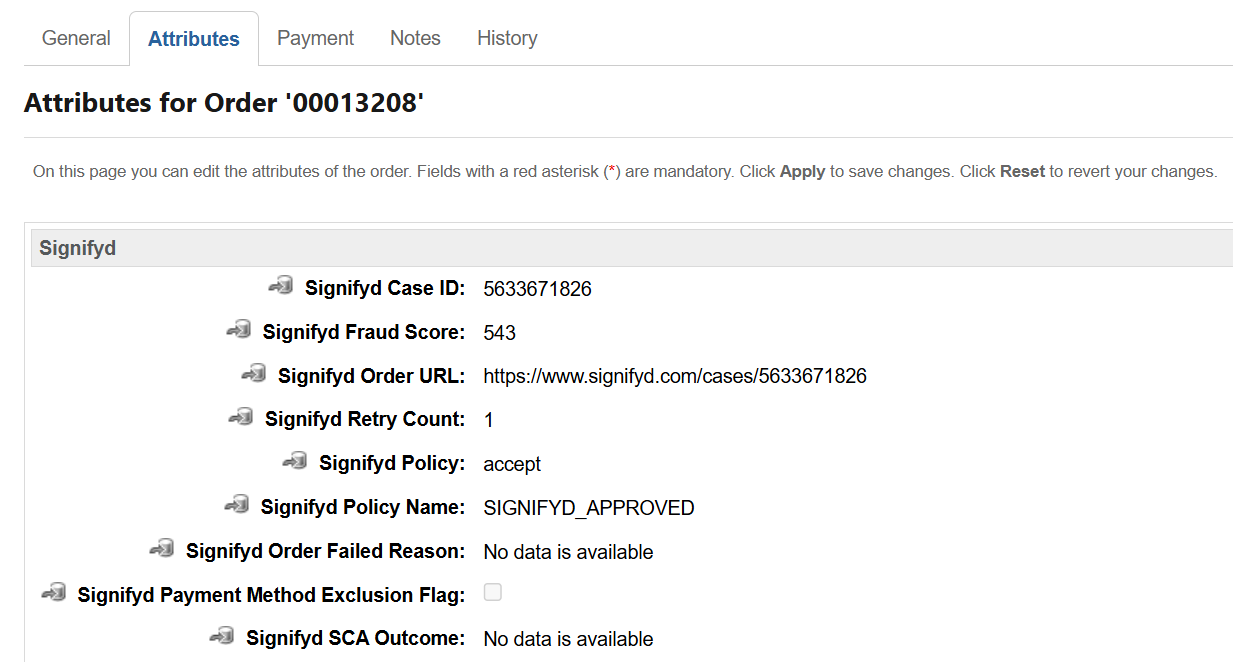
Description automatically generated

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

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

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

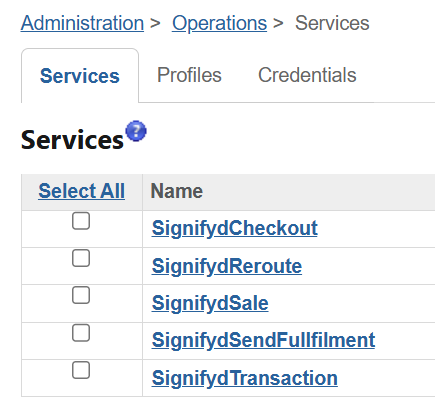


### 5.1.2 Services

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



Once the services file is uploaded click Import and choose **Merge** to import the default Service Framework configuration. Navigate to the Signifyd services configuration and make sure they are enabled.



### 5.1.3 Jobs

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

Interface gráfica do usuário, Aplicativo

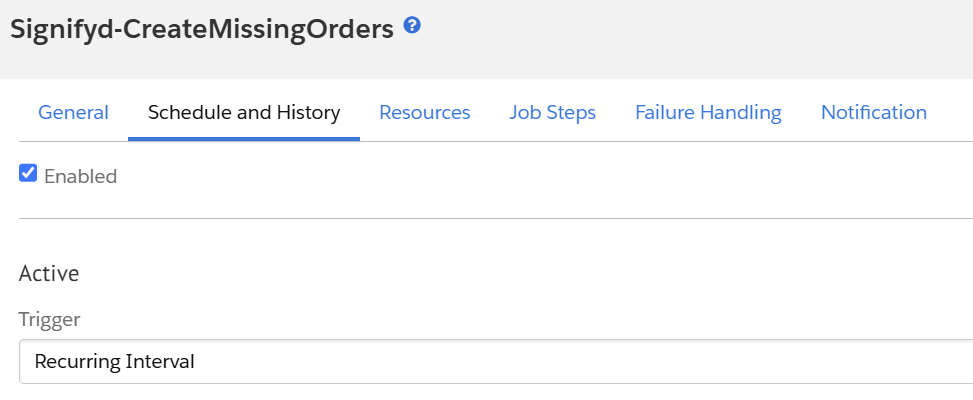
O conteúdo gerado por IA pode estar incorreto.

Once the jobs file is uploaded, click on Import and choose **Replace**to import the default Job Schedules configuration.

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



Select Signifyd-CreateMissingOrders to enter the Job Schedule configuration. Configure your Job Schedule to run once, daily, or on any desired schedule. We recommend that you schedule your job to run at least once a day.



## 5.2 SFRA Integration

### 5.2.1 Script changes

**Script:** checkoutHelpers.js

**Path**: signifyd\_sfra\_changes/cartridge/scripts/checkout/checkoutHelpers.js

Function **placeOrder()** on **checkoutHelpers.js** was modified to add the following logic:

If the cartridge is enabled and the **SignifydHoldOrderEnable** is set to **Yes**, then the order export status will be set to Not Exported and will later be updated based on Signifyd’s webhook decision.

If the cartridge is enabled and the **SignifydHoldOrderEnable** is set to **No**, then the order export status will be set to Ready For Export as in the default **placeOrder()** function.

Different logic can be added if the merchant wants to have customization for specific payments.

### 5.2.2 Adding the necessary changes

**Case 1 - Merchant didn’t customize the base SFRA checkoutHelpers.js file**

The file **checkoutHelpers.js** contains code to override the SFRA default function **placeOrder()** and add custom logic to set the export status according to the **SignifydHoldOrderEnable** custom preference.

The file should be placed on the following path, to extend the existing SFRA file:

**{merchant\_customized\_cartridge}/cartridge/scripts/checkout/checkoutHelpers.js**

**Case 2 - Merchant already customized the base SFRA checkoutHelpers.js file**

If the merchant already has custom logic added on his own checkoutHelpers.js file, only the necessary changes can be added (example file available on signifyd\_sfra\_changes/cartridge/scripts/checkout/checkoutHelpers.js):

Text

Description automatically generated

## 5.3 SiteGenesis Integration – Controllers

### 5.3.1 Modifications in System Controller

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

Add these two rows in the beginning of the ***submit***method:

var Signifyd = require('int\_signifyd/cartridge/scripts/service/signifyd');  
var orderSessionID = Signifyd.getOrderSessionId();

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 the last “else if” statement.

*else if (placeOrderResult.order\_created) {*   
**….<Insert code here>…..**  
}

The final code should look like this:

Texto

O conteúdo gerado por IA pode estar incorreto.

### 5.3.2 Modifications to Core Template – Device Fingerprinting

To insert the fingerprint JavaScript snippet in the HTML <head> element, modify the template **default/components/header/htmlhead.isml**.

Add the following lines at the end of the file:

<isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('SignifydEnableCartridge')}">  
 <isinclude template="signifyd\_device\_fingerprint" />  
</isif>

The result should look like the following:

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

## 5.4 SiteGenesis Integration – Pipelines

Another way to make SFCC send requests to Signifyd is the modification of the default 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 instead.

The Pipeline name is**COSummary***.*

Add the script **pp\_signifyd.ds**to the end of the **Submit** pipeline as shown below:

Diagrama

O conteúdo gerado por IA pode estar incorreto.

The input parameter for this script must be a current Order.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

## 5.5 Limitations and Constraints

### 5.5.1 AVS and CVV Response Codes

The Signifyd fraud service relies on transaction data 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.js* file as shown below:

The values for AVS and CVV Response Code fields MUST map to standard response codes. See [this document](http://www.emsecommerce.net/avs_cvv2_response_codes.htm) for valid response codes. AVS and CVV values should be updated by the merchant for getSendTransactionParams() (in case of Pre-auth enabled) and getParams() (in case Post-auth enabled) functions on signifyd.js file.

**getSendTransactionParams()**

Text, application

Description automatically generated

**getParams()**

Interface gráfica do usuário, Aplicativo, Word

O conteúdo gerado por IA pode estar incorreto.

### 5.5.2 Order Channel

Order channel information should be updated by the merchant in the **getParams()** function:

**Graphical user interface, text, application

Description automatically generated**

### 5.5.3 Fulfillment API

For the Fulfillment API, the following request fields should be updated by the merchant on the function **getSendFulfillmentParams()**, according to the merchant’s shipping carrier and following the format specified on the [API documentation](https://docs.signifyd.com/#operation/Fulfillments):

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

### 5.5.4 Pre-auth default error message

When an order is failed during Pre-auth flow, the default SFCC technical error message is displayed. The merchant can customize the code to display a different error message on the code below:

**Graphical user interface, text, application

Description automatically generated**

### 5.5.6 Digital items

If the merchant is selling digital items, the **itemIsDigital** field inside the purchase.products object should be updated to true. By default, the value is set to false.

Texto

O conteúdo gerado por IA pode estar incorreto.

### 5.5.7 Vanity hostnames

Salesforce will block any access to the hyphenated demandware.net (e.g. development-xxx.demandware.net) that does not originate from the platform itself, which means that any attempt from third-party integrations that access the Storefront through a link will not be able unless passed through a vanity hostname, such as yourbrand.com, www.yourbrand.com, etc. This should be taken into consideration while configuring the URLs for the webhooks in the Signifyd console.

For the Development and Production instances, it is mandatory to use the vanity hostname, so you will need to add an URL Rule like below, for the Staging and sandbox instances this way can be used as well, but it is not mandatory.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

And add e.g. https://yourStoreURL.com/s/SiteGenesis/signifydcallback to the Webhook Addresses in <https://app.signifyd.com/settings/notifications>

If the custom site preference ***SignifydCreateCasePolicy*** is set to ‘PRE\_AUTH’, there is no need to configure a webhook to receive the decision back, unless the merchant needs aforementioned Signifyd's manual review services.

## 5.6 Updating from V2 to V3

Merchants who are upgrading from the V2 API to the V3 API should install the V3 compatible cartridge on top of their existing V2 cartridge. This installation process will overwrite the existing cartridge files. It is important to import all files in the metadata folder to ensure that the respective metadata, jobs, and services are updated to be compatible with the V3 API.

Please keep in mind that any customizations made by the merchant will need to be reapplied on top of the V3 cartridge files after the installation is complete.

Any outdated services and custom preferences from the V2 cartridge that will not be used in the V3 version can be safely removed once the successful installation of the V3 cartridge has been confirmed.

The outdated V2 services that can be removed are shown on the image below:

A screenshot of a computer

Description automatically generated

In V3, those services were replaced with new services that reflect the V3 API structure:

* SignifydCheckout
* SignifydSale
* SignifydSendFullfilment
* SignifydTransaction

## 5.7 Logging

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

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Note:** DEBUG log level is only available in sandboxes, Development and Staging instances. For Production instances, any log level above DEBUG can be used.

## 5.8 Configurations on Signifyd Console

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>.

The ORDER\_CHECKPOINT\_ACTION\_UPDATE webhook should be setup following the instructions on [https://docs.signifyd.com/#operation/CreateTeamWebhook](https://docs.signifyd.com/%23operation/CreateTeamWebhook%20) in order to update SFCC with the latest status from Signifyd. Webhooks can be set up directly in the console by going to <https://app.signifyd.com/settings> >> Integrations >> and configuring the v3 ORDER\_CHECKPOINT\_ACTION\_UPDATE webhook for the given environments.

## 5.9 Testing

### 5.9.1 Pre-auth flow

**Pre-auth Test Case 1:** Signifyd response: Accept

**Expected Result**:

1. Before the payment processor authorization, a synchronous call to Checkout API is made and a case is created on Signifyd.
2. Signifyd sends the checkpointAction field as “Accept” in the response. The attributes **SignifydCaseID**, **SignifydOrderURL**, **SignifydFraudScore**, **SignifydPolicy** and **SignifydPolicyName** are saved on the order and the order proceeds to authorization.
3. Depending on the payment processor response:
   1. Authorized: order proceeds to be placed. A Transaction API call is made after the order is placed to update the order details in Signifyd.
   2. Not authorized or error: the order fails. A Transaction API call is made to update the order details in Signifyd.

**Pre-auth Test Case 2:** Signifyd response: Reject and Passive Mode: No.

**Expected Result**:

1. Before the payment processor authorization, a synchronous call to Checkout API is made and a case is created on Signifyd.
2. Signifyd sends the checkpointAction field as “Reject” in the response. The attributes **SignifydCaseID**, **SignifydOrderURL**, **SignifydFraudScore**, **SignifydPolicy** and **SignifydPolicyName** are saved on the order.
3. The order fails and the attribute **SignifydOrderFailedReason** is updated on the order.
4. The default error message is shown to the user on the checkout page

**Pre-auth Test Case 3:** Signifyd response: Reject and Passive Mode: Yes.

**Expected Result**:

1. Before the payment processor authorization, a synchronous call to Checkout API is made and a case is created on Signifyd.
2. Signifyd sends the checkpointAction field as “Reject” in the response. The attributes **SignifydCaseID**, **SignifydOrderURL**, **SignifydFraudScore**, **SignifydPolicy** and **SignifydPolicyName** are saved on the order.
3. The attribute **SignifydOrderFailedReason** is updated on the order and the order proceeds to be authorized.
4. Depending on the payment processor response:
   1. Authorized: order proceeds to be placed. A Transaction API call is made after the order is placed to update the order details in Signifyd.
   2. Not authorized or error: the order fails. A Transaction API call is made to update the order details in Signifyd.

**Pre-auth Test Case 4:** Signifyd response: Hold

**Expected Result**:

1. Before the payment processor authorization, a synchronous call to Checkout API is made and a case is created on Signifyd.
2. Signifyd sends the checkpointAction field as “Hold” in the response. The attributes **SignifydCaseID**, **SignifydOrderURL**, **SignifydFraudScore**, **SignifydPolicy** and **SignifydPolicyName** are saved on the order.
3. Depending on the payment processor response:
   1. Authorized: order proceeds to be placed. A Transaction API call is made after the order is placed to update the order details in Signifyd.
   2. Not authorized or error: the order fails. A Transaction API call is made to update the order details in Signifyd.
4. ORDER\_CHECKPOINT\_ACTION\_UPDATE Webhook will be triggered to send a new checkpointAction and update the order **SignifydPolicy** and **SignifydPolicyName** attributes.

### 5.9.2 Post-auth flow

**Post-auth Test Case 1:** Signifyd response: Accept

1. After the order is successfully placed, a call to Sale API is made.
2. The attributes **SignifydCaseID** and **SignifydOrderURL** are saved on the order.
3. The ORDER\_CHECKPOINT\_ACTION\_UPDATE Webhook is triggered, a callback is made to SFCC to update the order attribute **SignifydPolicy** with the “Accept” checkpointAction value from the response.
4. If **SignifydHoldOrderEnable** custom preference is set to Yes, the order export status is updated to Ready to Export.

**Post-auth Test Case 2:** Signifyd response: Reject

1. After the order is successfully placed, a call to Sale API is made.
2. The attributes **SignifydCaseID** and **SignifydOrderURL** are saved on the order.
3. The ORDER\_CHECKPOINT\_ACTION\_UPDATE Webhook is triggered, a callback is made to SFCC to update the order attribute **SignifydPolicy** with the with the “Reject” checkpointAction value from the response.
4. If **SignifydHoldOrderEnable** custom preference is set to Yes, the order export status is updated to Not Exported.

### 5.9.3 Automated Testing

On this implementation we have provided unit and integration testing.

#### 5.9.3.1 Unit testing

To do the automated testing open your command prompt (Windows) or terminal (MacOS/Linux/Unix) navigate to the folder above the cartridges and tun the command “npm run test”, this will test the functionalities within the cartridge source code.

#### 5.9.3.2 Integration testing

To run the automated integration testing create a dw.json file on the navigate to the folder above the cartridges

{  
 "hostname" : "somesb.demandware.net",  
 "username" : "someUser",  
 "password" : "somePassoword",  
 "version" : "someversion"  
}

Then run the command npm run test:integration

## 5.10 Troubleshoot

### 5.10.1 Missing API Key

If the API key provided by Signifyd is not correctly informed in the site custom preferences, the following error will be seen in the logs for any of the services:

Null Arguments in The given string to encode was null

**Solution**: The merchant should verify if the custom preference **SignifydApiKey** is correctly setup with the API key.

### 5.10.2 Wrong API Key

If the API key is wrong, the following error will be seen in the logs for any of the services:

Provided API Key not found.

**Solution**: The merchant should double check if the custom preference **SignifydApiKey** has the correct API key.

### 

### 5.10.3 Service not enabled

If a service is not correctly enabled in Business Manager, the following error will be triggered:

Service={name\_of\_the\_service} status=SERVICE\_UNAVAILABLE errorCode=0 errorMessage=The service is not enabled

**Solution**: The merchant should double check if the service informed in the error message is enabled.

### 5.10.4 Order fields not being updated

If the Signifyd related fields in the orders are not being updated, the possible reasons are:

* **Incorrect setup of the code**. The merchant should check if all steps to integrate the cartridge were correctly followed.
* **Incorrect setup of webhooks**. The merchant should check if only the ORDER\_CHECKPOINT\_ACTION\_UPDATE webhook is being used and correctly setup in the Signifyd console.
* **Code customizations**. The merchant should check if any customizations applied are not interfering in the cartridge functionality.
* **Incorrect data being passed in the service requests**. Some of the fields for Transaction and Fulfillment API need to be manually mapped by the merchant directly in the code during the integration. If a field is not correctly mapped or has a wrong value, this might cause the API to not work correctly, thus not updating the order.
* **Service timeouts**. The service timeout can be increased in the service configurations if needed, following SFCC default steps to do it. If the timeouts persist, Signifyd should be contacted.

# 6 Operations, Maintenance

## 6.1 Availability

Availability/Uptime is 24/7 is the intended access. But in case of service failure the order will get placed. And a script step job can be configurated to create the missing orders not sent to Signifyd.

## 6.2 Support

For implementation questions or issues please contact your assigned Customer Success Manager. For general support questions or issues [contact Support](https://www.signifyd.com/contact/).

## 6.3 Intended Locales

Out of the box the cartridge supports “en\_US” locale, but other locales may be added according to Signifyd’s service availability, Signifyd has support for multiple countries and locales.

# 7 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 |
| 19.1.0 | 02/28/2019 | Added modifications to support SFRA |
| 20.1.0 | 03/24/2020 | Added Unit and Integration testing |
| 22.1.0 | 05/31/2022 | V3 API release |

# 8 Process Flow Diagrams

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.