**Cloudshelf Cartridge**

**for**

**Salesforce B2C Commerce Cloud**

**Integration Documentation**

[Versions 2](#_heading=h.30j0zll)

[Abbreviations 2](#_heading=h.1fob9te)

[Summary 2](#_heading=h.3znysh7)

[Overview 2](#_heading=h.2et92p0)

[Compatibility 2](#_heading=h.tyjcwt)

[Implementation Guide 2](#_heading=h.3dy6vkm)

[Prerequisites 2](#_heading=h.1t3h5sf)

[Installing cartridge 2](#_heading=h.4d34og8)

[Initial Install 3](#_heading=h.2s8eyo1)

[Registering the Cartridge with Individual SFCC Sites 3](#_heading=h.17dp8vu)

[Registering the Cartridge Globally 3](#_heading=h.3rdcrjn)

[Importing Metadata, Services and Jobs 4](#_heading=h.26in1rg)

[Cartridge metadata 5](#_heading=h.lnxbz9)

[Setup 6](#_heading=h.35nkun2)

[Configuring Custom Site Preferences for API key 6](#_heading=h.1ksv4uv)

[Custom Site Preferences Configurations 6](#_heading=h.44sinio)

[Jobs & Schedule 7](#_heading=h.2jxsxqh)

[Order management 8](#_heading=h.z337ya)

[Components Overview 10](#_heading=h.3j2qqm3)

[Products Export 10](#_heading=h.1y810tw)

[High level tech approach 10](#_heading=h.4i7ojhp)

[ProductInput mapping 11](#_heading=h.2xcytpi)

[UpsertVariantsInput mapping 12](#_heading=h.1ci93xb)

[ProductGroups Export Overview 13](#_heading=h.3whwml4)

[ProductGroupInput mapping 13](#_heading=h.2bn6wsx)

[Locations Export 13](#_heading=h.qsh70q)

[High level tech approach 13](#_heading=h.3as4poj)

[LocationInput mapping 13](#_heading=h.1pxezwc)

[Cloudshelf to SFCC basket transfer 14](#_heading=h.49x2ik5)

[Purpose 14](#_heading=h.2p2csry)

[Prerequisites 14](#_heading=h.147n2zr)

[Overview of expectations 14](#_heading=h.3o7alnk)

[Overview of technical approaches 14](#_heading=h.23ckvvd)

[OCAPI flow 14](#_heading=h.ihv636)

[OCAPI configuration 16](#_heading=h.32hioqz)

[Simple Redirect Flow 16](#_heading=h.1hmsyys)

[Cloudshelf and Theme creation 17](#_heading=h.41mghml)

# 

# Versions

| version | date | comment |
| --- | --- | --- |
| 1.0.0 | 24 Jan 2024 | Initial implementation includes products export, stores export; creating of default theme and cloudshelf; transfer basket from cloudshelf to SFCC |
|  |  |  |

# Abbreviations

For this document the following abbreviations are in place:

* Salesforce B2C commerce cloud: SFCC
* Open commerce API: OCAPI
* StoreFront Reference Architecture: SFRA
* Business Manager: BM

# Summary

This document provides general overview and technical instruction of installing of Salesforce Commerce Cloud (SFCC) LINK cartridge that lets you integrate Cloudhshelf with SFCC.

Also, it provides overview of components and functionality implemented in scope of the cartridge.

# Overview

Cloudshelf cartridge (int\_cloudshelf) for Salesforce B2C Commerce Cloud.

Cartridge allows to connect SFCC instance with Cloudshelf and includes the next functionalities:

* Export products and catalogs from SFCC to Cloudshelf
* Export stores from SFCC to Cloudshelf
* Order management (aka transfer of basket from Cloudshelf to SFCC)
* Update order status on Cloudshelf

The cartridge can be used by any new or existing SFCC retailer to be able to connect to Cloudshelf after simply installing the cartridge.

# Compatibility

Cartridge is compatible with 6.3.0 and 7.0.0 SFRA version and Salesforce Commerce Cloud API version 24.1 (Compatibility Mode: 22.7)

# Implementation Guide

## Prerequisites

Before installing cartridge on SFCC side make sure you have:

* Account and API key created on Cloudshelf side - <https://documentation.cloudshelf.ai/getting-started/welcome>
* SFCC instance with setup site and product catalog
* SFCC business user account

## Installing cartridge

Complete the following steps to install the cartridge.

### Initial Install

See the SFCC documentation for adding a cartridge to your storefront:

* <https://developer.salesforce.com/docs/commerce/sfra/guide/b2c-build-sfra.html#upload-code-for-sfra>

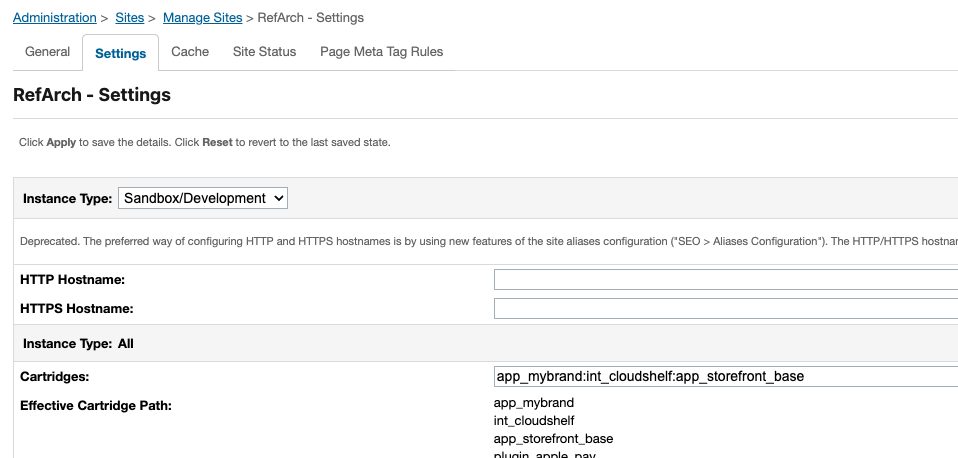
Cartridge does not contain any additional npm dependencies or client js script changes and can be simply uploaded to SFCC instance after receiving a copy of it.

Note: You must also register the cartridge with the Business Manager site in order to run the Cloudshelf jobs.

Navigate to **Administration > Sites > Manage Sites > Business Manager > Settings tab**.

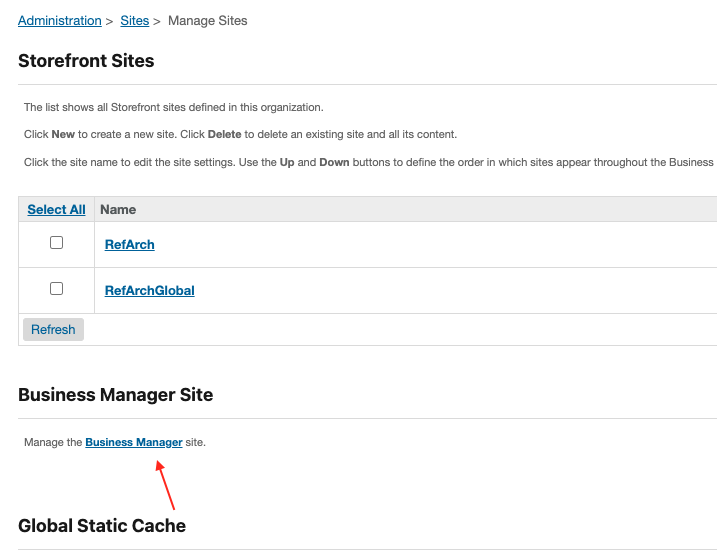
### Registering the Cartridge with Individual SFCC Sites

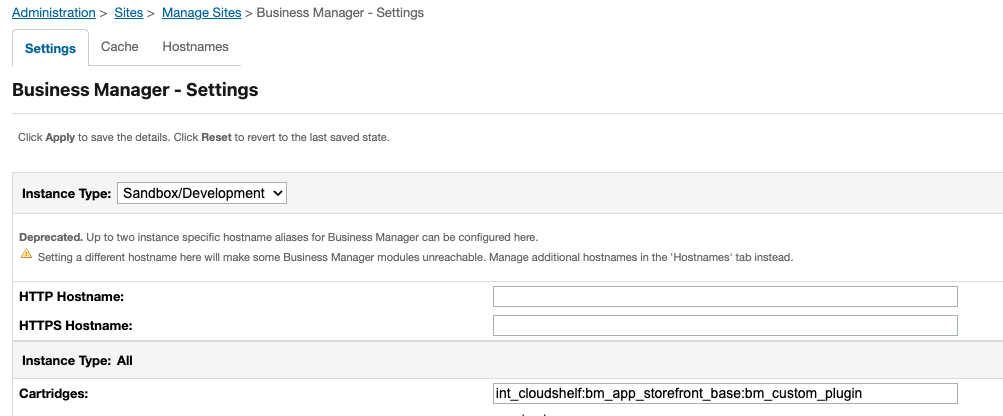
1. Navigate to Administration > Sites > Manage Sites > [SiteName] > Settings tab.
2. Add “:int\_cloudshelf” to the effective cartridge path.
3. Click “Apply”.



### Registering the Cartridge Globally

1. Navigate to Administration > Sites > Manage Sites
2. Click on Manage the Business Manager site link
3. Add “:int\_cloudshelf”” to the effective cartridge path.
4. Click “Apply”.

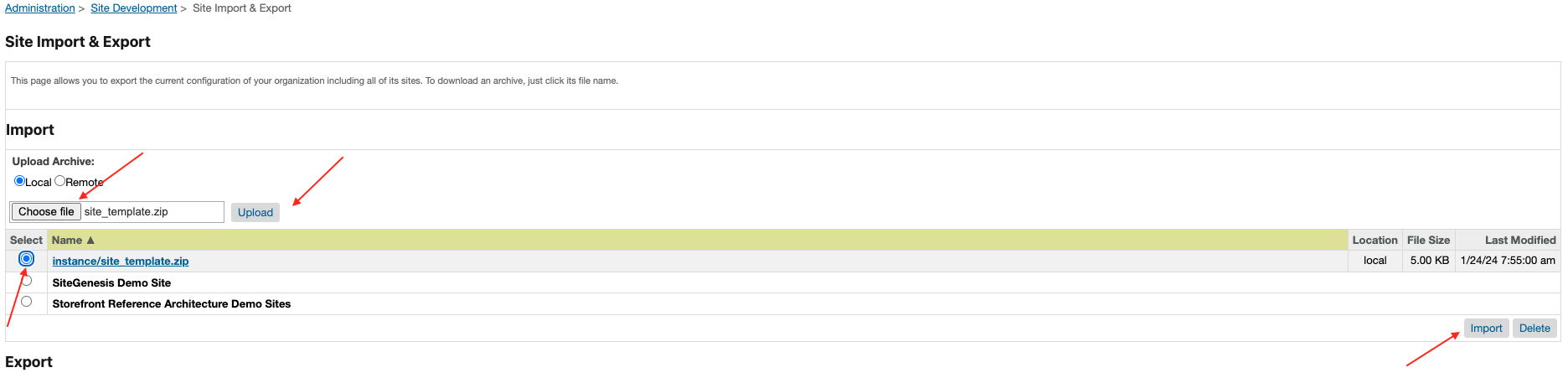




### Importing Metadata, Services and Jobs

Complete the following steps to import the Cloudshelf metadata, jobs and services.

1. Navigate to Administration > Site Development >Site Import & Export
2. In the Upload Archive section
3. Archive data/site\_template folder from the repository and use it for upload
4. Select Upload.
5. On page reload select the site\_template radio button
6. Press Import Button



Depending on the SFCC environment you use, optionally it is possible to import separately metadata services and jobs.

* Metadata files (files in *data/site\_template/meta* folder) can be imported in BM > Administration > Site Development > Import & Export (first upload xml files then import Meta Data).
* *data/site\_template/jobs.xml* file can be uploaded and imported in BM > Administration > Operations > Import & Export (first upload xml files then import Jobs).
* *data/site\_template/services.xml* file can be uploaded and imported in BM > Administration > Operations > Import & Export (first upload xml files then import Services).

#### Cartridge metadata

After import, these system definition attributes and custom definition attributes, jobs and services should appear in the system.

**System object definitions**

* Order & Basket:
  + isCloudshelf
  + cloudshelfData
  + cloudshelfStatus
* Store:
  + IsCloudshelf
* Site preferences
  + Cloudshelf group
  + cloudshelfAPIKey
  + CloudshelfProductMetadataMapping
  + CloudshelfCategoryMetadataMapping
  + CloudshelfProductMetadataMapping
* Product
  + CloudshelfTags (set-of-string)

**Custom object definitions**

* JobsData

**Service**

* **cloudshelf.http.graphql** - SFCC connects with cloudshelf with [GraphQL API](https://documentation.cloudshelf.ai/background-information/about-graphql)

**Jobs**

* CloudshelfDataExportFull
* CloudshelfDataExportDelta
* CloudshelfOrderStatusUpdate

## Setup

Configuring custom Site Preferences for Cloudshelf API key value is required. All other settings are optional.

### Configuring Custom Site Preferences for API key

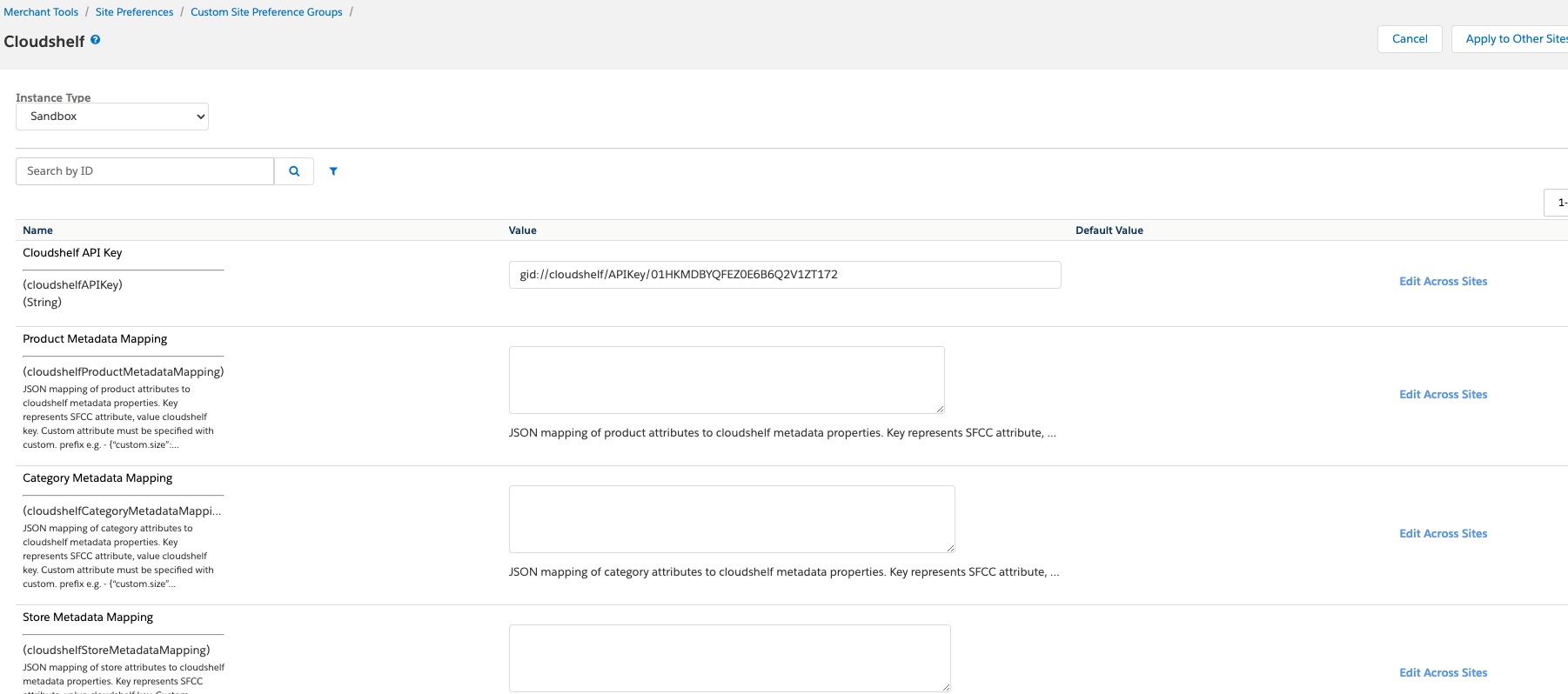
Complete the following steps.

1. Navigate to Site > (select desired SFCC Site) > Site Preferences > Custom Preferences > Cloudshelf
2. Select the appropriate Instance (Sandbox/Development, Staging, Production), and click Apply.
3. Set the following using the value provided by your Cloudshelf representative: Cloudshelf API Key
4. Click Apply.

### Custom Site Preferences Configurations

Other setting available in **Navigate to Site > (select desired SFCC Site) > Site Preferences > Custom Preferences > Cloudshelf**

| **id** | **name** | **description** |
| --- | --- | --- |
| cloudshelfProductMetadataMapping | Product Metadata Mapping | JSON mapping of product attributes to [cloudshelf product metadata properties](https://documentation.cloudshelf.ai/types/Product). Key represents SFCC attribute, value cloudshelf key. Custom attribute must be specified with custom. prefix e.g. - {“custom.size”: “size”, “custom.badge”: “symbol”} |
| cloudshelfCategoryMetadataMapping | Category Metadata Mapping | JSON mapping of category attributes to [cloudshelf product group metadata properties](https://documentation.cloudshelf.ai/types/ProductGroup). Key represents SFCC attribute, value cloudshelf key. Custom attribute must be specified with custom. prefix e.g. - {“custom.size”: “size”, “custom.badge”: “symbol”} |
| cloudshelfStoreMetadataMapping | Store Metadata Mapping | JSON mapping of store attributes to [cloudshelf location metadata properties](https://documentation.cloudshelf.ai/types/Location). Key represents SFCC attribute, value cloudshelf key. Custom attribute must be specified with custom. prefix e.g. - {“phone”: “mobile”, “custom.badge”: “symbol”} |



## Jobs & Schedule

In BM go to Administration > Operations > Jobs.

Two jobs are part of the initial cartridge setup

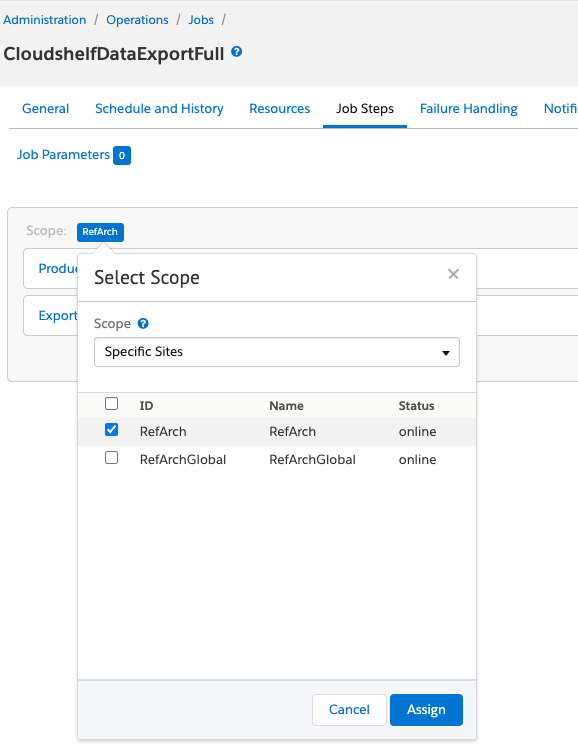
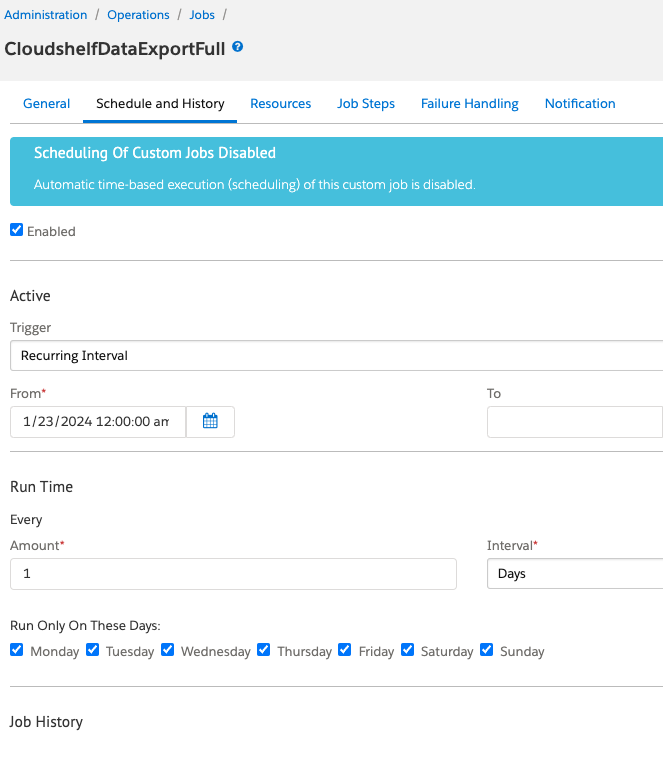
* CloudshelfDataExportFull – Full export of SFCC products and categories to cloudshelf. Also, exports SFCC stores marked with isCloudshelf flag to Cloudshelf system as locations
* CloudshelfDataExportDelta – Delta export of products data to Cloudshelf
* CloudshelfOrderStatusUpdate - Update order status on Cloudshelf side, when it’s changed on SFCC

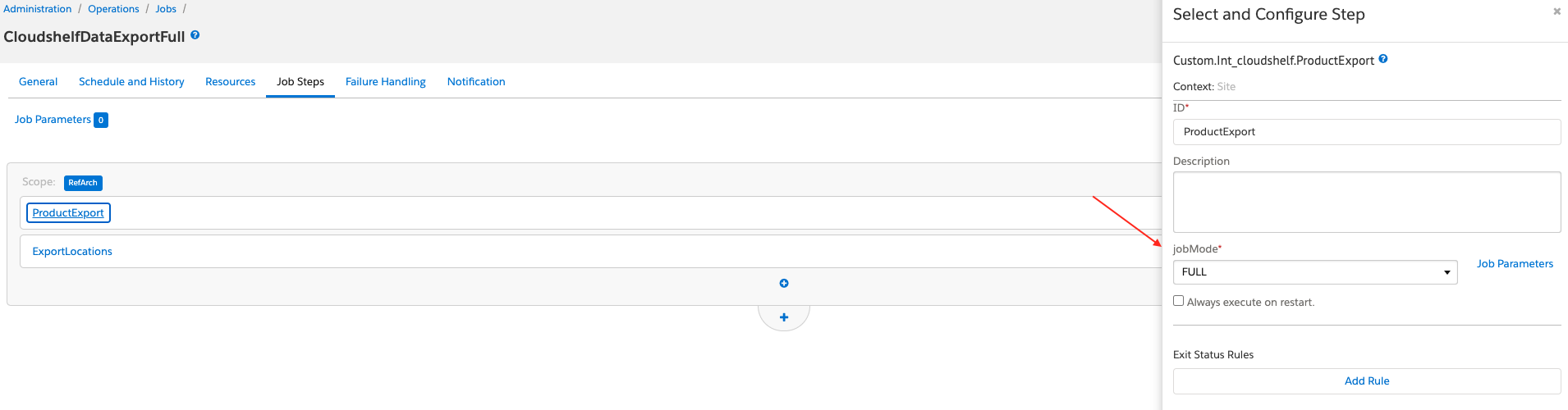
For initial full products export simply click “Run” button of CloudshelfDataExportFull job. It will exports all sites products and categories to Cloudshelf and create default theme and cloudshelf entities in Cloudshelf system.

To export stores from SFCC to Cloudshelf as location go to BM > Merchant Tools > Online Marketing > Stores > [select store] > CLOUDSHELF Tab, set isCloudShelf to be "Yes" and click "Save"

To change jobs configs in BM go to Administration > Operations > Jobs > select a specific job

* In Schedule and History tab you can configure triggering interval for the job
* In Job Steps tab you can select specific sites for which export should work
* In Job Steps tab by selecting ProductExport step you can configure export mode (FULL or Delta)

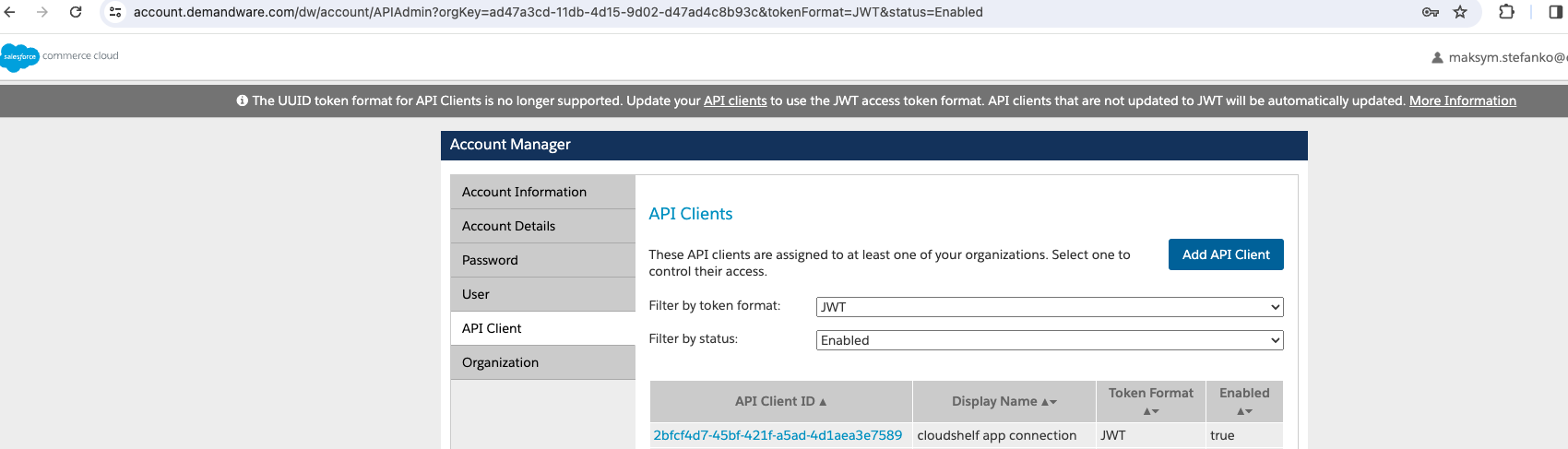




## Order management

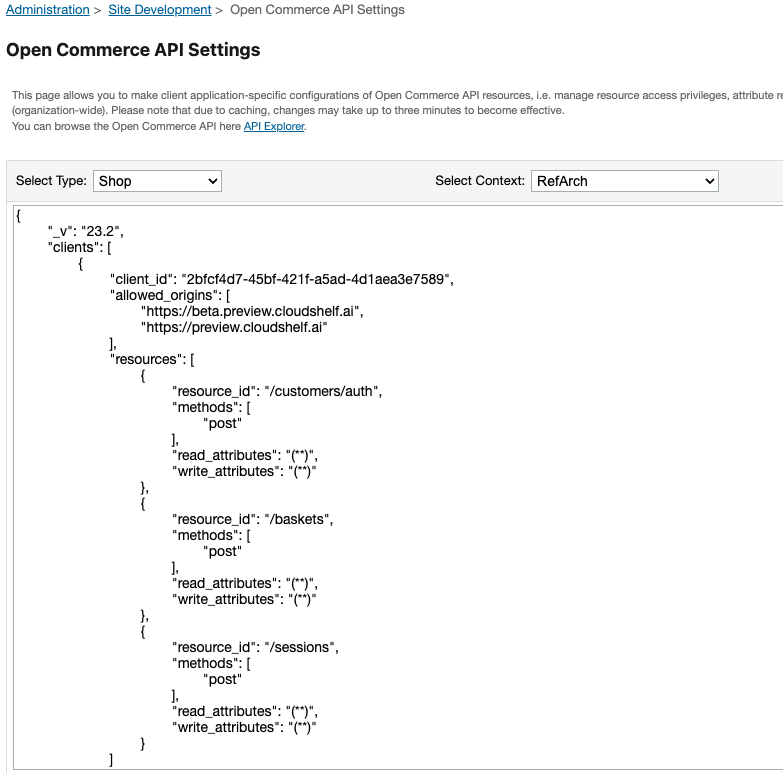
Transferring basket from Cloudshelf to SFCC functionality works using OCAPI as result the next are required: creating OCAPI client id and setting up OCAPI configs in business manager.

[OCAPI client and appropriate client id](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/b2c-commerce-ocapi/clientapplicationidentification.html) can be created in [SFCC account manager](https://account.demandware.com/dw/account/Home#!/) by account manager administrator.



OCAPI client permission and allowed origins for specific SFCC organization or site can be setup in the SFCC business manager using [OCAPI setting](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/b2c-commerce-ocapi/ocapisettings.html).

In BM go to Administration > Site Development > Open Commerce API Settings > select required site context



Example of OCAPI config:

{

"client\_id":"2bfcf4d7-45bf-421f-a5ad-4d1aea3e7589",

"allowed\_origins":[

"https://beta.preview.cloudshelf.ai",

"https://preview.cloudshelf.ai"

],

"resources":[

{

"resource\_id":"/customers/auth",

"methods":[

"post"

],

"read\_attributes":"(\*\*)",

"write\_attributes":"(\*\*)"

},

{

"resource\_id":"/baskets",

"methods":[

"post"

],

"read\_attributes":"(\*\*)",

"write\_attributes":"(\*\*)"

},

{

"resource\_id":"/sessions",

"methods":[

"post"

],

"read\_attributes":"(\*\*)",

"write\_attributes":"(\*\*)"

}

]

}

More details about Shop OCAPI can be found by link - <https://developer.salesforce.com/docs/commerce/b2c-commerce/references/ocapi-shop-api?meta=Summary>

General OCAPI documentation link - <https://developer.salesforce.com/docs/commerce/b2c-commerce/references/b2c-commerce-ocapi/get-started-with-ocapi.html>

# Components Overview

## Products Export

Products data will be exported from SFCC to Cloudshelf using GraphQL API provided by Cloudshelf.

<https://documentation.cloudshelf.ai/guides/product-ingestion>

To add product to Cloudshelf system the next API call required:

* <https://documentation.cloudshelf.ai/mutations/upsertProducts>
* <https://documentation.cloudshelf.ai/mutations/upsertProductVariants>

Currently on Cloudshelf supported product types are master and variation products. In case of simple product on SFCC side we will have to create both base (master) product and variant product on Cloudshelf side. Product bundles and product set currently will not be exported from SFCC to Cloudshelf.

On SFCC side there will be a job that trigger full or batch products export from SFCC to Cloudshelf.

### High level tech approach

There is a job step that pickups all site products and send them to Cloudshelf in chunks of 150.

* Created and implemented chunk-oriented custom job step for exporting products from SFCC to Cloudshelf (custom.int\_cloudshelf.ProductExport)
  + Chunk size = 150 products
  + Gets all site products and process them by chunks of 150
    - uses search model to get all site products (search by root category)
  + For each chunk generates array of cloudshelf product models and send all information in scope of one API call for upsertProducts and another API call for upsertProductVariants
  + Step parameters:
    - Mode (FULL or DELTA)
* Created and implemented cloudshelf models for products
  + base cloudshelf product model
    - <https://documentation.cloudshelf.ai/types/ProductInput>
    - Model is build based on SFCC Product system object and ProductInput mapping provided below
  + variation cloudshelf product model
    - <https://documentation.cloudshelf.ai/types/UpsertVariantsInput>
    - Model is build based on SFCC Product system object and UpsertVariantsInput mapping provided below
* Created site preferences in Cloudshelf preferences group
  + CloudhselfAPIKey (string)
  + CloudhselfProductMetadataMapping (text) - that will be JSON config object
* Delta export approach
  + Datetime value of last successful job execution is saved on custom object level and then compared with product.getLastModified() time. Exports only products that were updated after last successful job execution time. Updates successful job execution datetime after job is finished with OK status
* CloudhselfProductMetadataMapping approach
  + The idea is to have a json config mapping of product attributes to cloudshelf metadata properties:
    - Example of site prefs config: {“size”: “size”, “badge”: “symbol”}
    - SFCC Product’s size attribute should be sent as {"key": "size ",”data”: “...value of product size attribute” }
    - SFCC Product’s badge attribute should be sent as {"key": "symbol",”data”: “...value of product badge attribute” }

### ProductInput mapping

<https://documentation.cloudshelf.ai/types/ProductInput>

| Cloudshelf value | SFCC value | Comment |
| --- | --- | --- |
| description | short description |  |
| displayName | product name |  |
| id | value generated based on id value | gid://external/salesforceProduct/:id |
| metadata | any string product attribute value can be passed here based on custom preferences mapping | by default, it will be empty.  each retailer will have an ability to configure attributes those values should be sent here |
| productType | Primary category name | By default, it will be the product's primary category. Optionally there will be ability to specify product attribute that should be used for this value |
| tags | new custom attribute (set of string) will be created |  |
| vendor | brand |  |

### UpsertVariantsInput mapping

<https://documentation.cloudshelf.ai/types/UpsertVariantsInput>

| Cloudshelf value | SFCC value | Comment |
| --- | --- | --- |
| productId | Master product ID | id://external/salesforceProduct/:id |
| variants  [ProductVariantInput!]! |  |  |
| attributes | Variation attributes (e.g. color, size, etc) |  |
| displayName | Product name |  |
| id | value generated based on id value | gid://external/salesforceVariant/:id |
| isInStock | True/false based on availability | True if product is in stock, false otherwise |
| availableToPurchase | True/false based on availability | True if product is in stock or available for pre-order, false otherwise |
| metadata | any string product attribute value can be passed here based on custom preferences mapping | By default, it will be empty.  each retailer will have an ability to configure attributes those values should be sent here |
| metaimages | Images absolute URLs | First image will be marked as preferredImage |
| originalPrice | List price |  |
| currentPrice | Sale price |  |
| sku | manufacturerSKU |  |

### ProductGroups Export Overview

ProdutGroups on Cloudshelf side are represented by storefront categories on SFCC side.

ProdutGroups (categories) data will be exported from SFCC to Cloudshelf using GraphQL API provided by Cloudshelf.

<https://documentation.cloudshelf.ai/guides/product-group-ingestion>

### ProductGroupInput mapping

<https://documentation.cloudshelf.ai/types/ProductGroupInput>

| Cloudshelf value | SFCC value | Comment |
| --- | --- | --- |
| displayName | Category name |  |
| id | value generated based on id value | gid://external/salesforceCategory/:id |
| metadata | any string category attribute value can be passed here based on custom preferences mapping | by default, it will be empty.  each retailer will have an ability to configure attributes those values should be sent here |
| featuredImage | Category image attribute | Absolute url value |

## Locations Export

Cloudshelf locations are physical stores where cloudshelf app is running.

Cloudshelf locations will be represented by SFCC stores.

Stores data will be exported from SFCC to Cloudshelf using GraphQL API provided by Cloudshelf.

<https://documentation.cloudshelf.ai/mutations/upsertLocations>

On SFCC side there will be a separate job step that exports stores from SFCC to Cloudshelf.

### High level tech approach

* Created a new boolean custom attribute for store object that indicates if store should be exported as location to cloudshelf
* Created a custom job step for exporting location to cloudshelf (custom.int\_cloudshelf.exportLocationsToCloudShelf)
* Used SystemObjectMgr.querySystemObject method to search for stores that marked with export to Cloudshelf flag
* Implemented Cloudshelf location model that build object for export based on SFCC system store object. Use mapping provided below in LocationInput mapping section
* Used cloudshelf service to call graphQL API for export

### LocationInput mapping

<https://documentation.cloudshelf.ai/types/LocationInput>

| Cloudshelf value | SFCC value | Comment |
| --- | --- | --- |
| id | value generated based on id value | gid://external/ SalesforceLocation/:id |
| address | address1 + postalCode + city | Concatenation of address related fields in SFCC separated by comma |
| countryCode | Country code |  |
| displayName | Store name |  |
| metadata | any string store attribute value can be passed here based on custom preferences mapping | by default, it will be empty.  each retailer will have an ability to configure attributes those values should be sent here |

## Cloudshelf order status update

### Scope

CloudshelfOrderStatusUpdate job and order custom attribute cloudshelfStatus

Implementation overview  
CloudshelfOrderStatusUpdate job updates order status on Cloudshelf and order custom attribute: cloudshelfStatus , when it’s changed on SFCC. Supported only two statuses update:   
- PAID - when SFCC order payment status = Paid  
- VOIDED - when SFCC order status Cancelled

## Cloudshelf to SFCC basket transfer

### Purpose

The purpose of this section of document is to provide an explanation of how to redirect to the SFCC storefront with a shopping cart created.

### Prerequisites

int\_cloudshelf cartridge must be set up on the SFCC side. Products data must be exported from SFCC to CLoudshelf

### Overview of expectations

After exporting products from SFCC to Cloudshelf, you can create a basket on the cloudshelf side. Customer must be redirected from cloudshelf system to SFCC checkout to complete payment and finalize order creation. A basket with the selected items on cloudshelf side should be created on SFCC side upon redirect.

### Overview of technical approaches

There are two options for moving the basket from Cloudshelf to SFCC.

The first option is that the basket must be created on the Cloudshelf side using the SFCC Open commerce API (OCAPI), then Cloudshelf includes the session ID parameter in the redirect link to the SFCC.

The second option is that the basket is created on the SFCC side based on the request parameter passed within the redirect link to the SFCC.

The first approach is recommended and preferable. The second should be used only as a fallback if for some reason it is not possible to make API calls from cloudshelf application installed in store.

#### OCAPI flow

Before redirecting to the SFCC storefront, the basket must be created through Open commerce API calls from Cloudshelf to SFCC. Three OCAPI calls are required. Details and link to documentation are provided below:

* **/customers/auth** ("type":"guest") - to get jwt
  + [Get or refresh customer JWT (JSON Web Token)](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/ocapi-shop-api?meta=Get%2Bor%2Brefresh%2Bcustomer%2BJWT%2B(JSON%2BWeb%2BToken))
* **/baskets** - create basket and populate with product items information
  + [Create basket](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/ocapi-shop-api?meta=Create%2Bbasket)
* **/sessions** - to get session id cookie value based on provided jwt
  + [Exchange JWT](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/ocapi-shop-api?meta=Exchange%2BJWT)
  + [OCAPI Session Bridge - Obtain Session](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/b2c-commerce-ocapi/sessionbridge.html#obtain-session)

Session id value from /sessions response (dwsid cookie value) must be provided as parameter in redirect to SFCC storefront link.

It is important to provide the next data in scope of basket creation API call:

* product\_items - array of product items with product ids and quantities
* c\_isCloudshelf - always true
* c\_cloudshelfData - stringified JSON object of all related to cloushelf details

Curl examples of OCAPI requests:

1. curl --location 'https://{{hostnmame}}/s/{{site}}/dw/shop/v23\_1/customers/auth?client\_id={{clinet\_id}} \

--header 'x-dw-client-id: {{clinet\_id}} \

--header 'Content-Type: application/json' \

--data '{

"type":"guest"

}'

1. curl --location 'https://{{hostnmame}}/s/{{site}}/dw/shop/v23\_1/baskets?client\_id={{clinet\_id}} \

--header 'Authorization: Bearer eyJfdiI6IjEiLCJhb.....' \

--header 'x-dw-client-id: {{clinet\_id}} \

--header 'Content-Type: application/json' \

--data '{

"product\_items": [

{

"product\_id": "78916783M-2",

"quantity": 1

},

{

"product\_id": "883360544212M",

"quantity": 2

}

],

"c\_isCloudshelf": true,

"c\_cloudshelfData": "{\"deviceId\":\"1234567890\",\"cloudshelfId\":\"1234567890\",\"salesAssistantId\":\"1234567890\",\"productItems\":[{\"productId\":\"78916783M-2\",\"quantity\":1}, {\"productId\":\"883360544212M\",\"quantity\":2}]}"

}'

1. curl --location --request POST 'https://{{hostnmame}}/s/{{site}}/dw/shop/v23\_1/sessions?client\_id={{clinet\_id}} \

--header 'Authorization: Bearer eyJfdiI6IjEiLCJh....' \

--header 'x-dw-client-id: {{clinet\_id}}

Example of redirect URL:

*https://{{hostnmame}}/on/demandware.store/Sites-{{site}}-Site/en\_US/Cloudshelf-SetSession?sid=wEOgyL-fLVxq-xfQTdV5YBNCNrNQEr7Ej-Pydz2bj1GyJhqaOvXZaJ5SwOEjh2TBdytJz-dvt5-22fZiduN54Q%3D%3D*

#### OCAPI configuration

Before OCAPI can be used there are some configurations required to be done on the SFCC side. Configuration might vary depending on if OCAPI is already used by specific retailers but in general it includes two main steps: creating OCAPI client id and setting up OCAPI configs in business manager.

[OCAPI client and appropriate client id](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/b2c-commerce-ocapi/clientapplicationidentification.html) can be created in [SFCC account manager](https://account.demandware.com/dw/account/Home#!/) by account manager administrator.

OCAPI client permission and allowed origins for specific SFCC organization or site can be setup in the SFCC business manager using [OCAPI setting](https://developer.salesforce.com/docs/commerce/b2c-commerce/references/b2c-commerce-ocapi/ocapisettings.html).

More details about Shop OCAPI can be found by link - <https://developer.salesforce.com/docs/commerce/b2c-commerce/references/ocapi-shop-api?meta=Summary>

General OCAPI documentation link - <https://developer.salesforce.com/docs/commerce/b2c-commerce/references/b2c-commerce-ocapi/get-started-with-ocapi.html>

#### Simple Redirect Flow

Only redirect link creation is required on cloudshelf side. All basket information should be provided to SFCC in the scope of generated link query parameter.

data parameter must be provided as URL encoded json string. As an example if data object is {"deviceId":"1234567890","cloudshelfId":"1234567890","salesAssistantId":"1234567890","productItems":[{"productId":"78916783M-2","quantity":1}]} then it must be URL encoded like %7B%22deviceId%22%3A%221234567890%22%2C%22cloudshelfId%22%3A%221234567890%22%2C%22salesAssistantId%22%3A%221234567890%22%2C%22productItems%22%3A%5B%7B%22productId%22%3A%2278916783M-2%22%2C%22quantity%22%3A1%7D%5D%7D

productItems property is always required for basket creation on the SFCC side. All other information is optional. Any information provided in scope of the link will be stored on SFCC basket and order level.

Example of redirect:

https://{{hostnmame}}/on/demandware.store/Sites-{{site}}-Site/en\_US/Cloudshelf-CreateBasket?data=%7B%22deviceId%22%3A%221234567890%22%2C%22cloudshelfId%22%3A%221234567890%22%2C%22salesAssistantId%22%3A%221234567890%22%2C%22productItems%22%3A%5B%7B%22productId%22%3A%2278916783M-2%22%2C%22quantity%22%3A1%7D%5D%7D

## Cloudshelf and Theme creation

Default theme and cloudshelf creation logic is part of the cartridge

* <https://documentation.cloudshelf.ai/mutations/upsertTheme>
* <https://documentation.cloudshelf.ai/mutations/upsertCloudshelves>

ThemeInput mapping

<https://documentation.cloudshelf.ai/types/ThemeInput>

| Cloudshelf value | SFCC value | Comment |
| --- | --- | --- |
| id | value generated based on site id value | gid://external/SalesforceBrand/[unique-store-id-here] |
| displayName | "Default Theme" constant |  |
| logoUrl | absolute url to site logo | /images/logo.svg |

CloudshelfInput mapping

<https://documentation.cloudshelf.ai/types/CloudshelfInput>

| Cloudshelf value | SFCC value | Comment |
| --- | --- | --- |
| id | value generated based on site id value | gid://external/SalesforceConnectorGeneratedCloudshelf/[unique-store-id-here] |
| randomContent | true |  |
| displayName | "First Cloudshelf" by default constant if not provided specific value |  |
| homeFrameCallToAction | "Touch to discover and buy" by default constant if not provided specific value |  |