Cloudinary SiteGenesis Integration Guide

**Version 20.1.0 for year 2020**

Icon

Description automatically generated

****

**Table of Contents**

[1. Summary 3](#_Toc53745452)

[2. Component Overview 4](#_Toc53745453)

[2.1 Functional Overview 4](#_Toc53745454)

[2.2 Limitations, Constraints 4](#_Toc53745455)

[2.3 Compatibility 4](#_Toc53745456)

[2.4 Privacy, Payment 4](#_Toc53745457)

[3. Implementation Guide 5](#_Toc53745458)

[3.1 General Implementation 5](#_Toc53745459)

[3.1.1 Installing the Cartridges on the sandbox 5](#_Toc53745460)

[3.1.2 Setting Up Business Manager 5](#_Toc53745461)

[3.2 Configuring Cloudinary 9](#_Toc53745462)

[3.2.1 Adding Structured Metadata in Cloudinary 9](#_Toc53745463)

[3.2.2 Enabling Resource Listing 10](#_Toc53745464)

[3.3 Configuring the Cartridge Site Preferences: 10](#_Toc53745465)

[3.4 System Object Types: 19](#_Toc53745466)

[3.5 Jobs Details: 23](#_Toc53745467)

[3.6 Services Details: 27](#_Toc53745468)

[3.7 Cartridge Operation Modes: 28](#_Toc53745469)

[3.8 Code Changes 29](#_Toc53745470)

[3.9 Managing Videos 54](#_Toc53745471)

[4. Operations, Maintenance 56](#_Toc53745472)

[4.1 Data Storage 56](#_Toc53745473)

[4.2 Availability 56](#_Toc53745474)

[4.3 Support 56](#_Toc53745475)

[4.4 Cloudinary Workflow 56](#_Toc53745476)

[5. User Guide 57](#_Toc53745477)

[5.1 Roles, Responsibilities 57](#_Toc53745478)

[5.2 Business Manager 57](#_Toc53745479)

[5.3 Storefront Functionality 58](#_Toc53745480)

[6 Known Issues 59](#_Toc53745481)

[7 Release History 60](#_Toc53745482)

# Summary

This document provides technical instructions for using the Cloudinary SalesForce Commerce Cloud Cartridge to integrate Cloudinary images and videos into an SFCC storefront.

**Glossary**

|  |  |
| --- | --- |
| **Term** | **Description** |
| SFCC | SalesForce Commerce Cloud |
| Business Manager (BM) | The primary tool used to configure the SFCC platform and customer storefront |
| SiteGenesis | The SFCC default ecommerce storefront |
| Cloudinary Transformation | A set of parameters specifying how to change an image or video in terms of size, visual style, format, quality and more. For details, see the transformation guides for images (<https://cloudinary.com/documentation/image_transformations>) and videos (<https://cloudinary.com/documentation/video_manipulation_and_delivery>). |

# Component Overview

## Functional Overview

The Cloudinary cartridge enables images and videos available in Cloudinary to be used on the merchant Storefront.

## Limitations, Constraints

Use of the Cloudinary SFCC Cartridge requires credentials, keys and URLs from Cloudinary. Please go to the Cloudinary website to sign up for the services (<https://cloudinary.com/signup>).

## Compatibility

The cartridge is designed for Salesforce Commerce Cloud(SG) version 105.1.0. Compatibility Mode: 19.10.

The Cloudinary cartridge works with all locales.

## Privacy, Payment

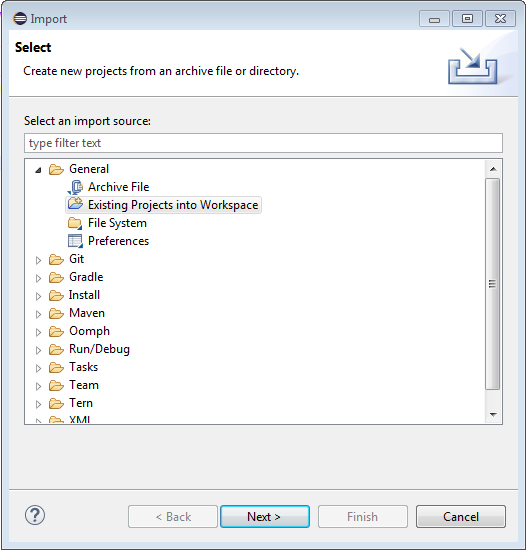
The Cloudinary cartridge does not collect and process any user profile information or billing information.

# Implementation Guide

## General Implementation

### 3.1.1 Installing the Cartridges on the sandbox

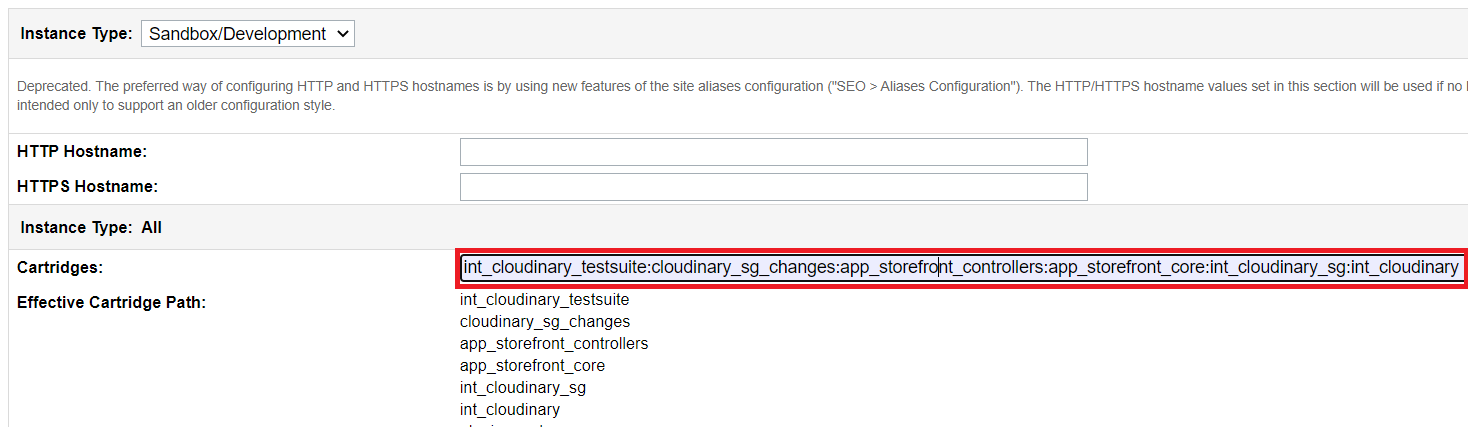
1. Download the int\_cloudinary int\_cloudinary\_sg bm\_cloudinary cloudinary\_sg\_changes and int\_cloudinary\_testsuite cartridges.
2. Open Demandware Studio.
3. Import the downloaded cartridges from the Import dialog:



1. Link the cartridges to the sandbox. Select the sandbox connection and then select Properties. Select Project Reference and check in except cloudinary\_sg\_changes because it’s only a reference to see how to make changes in base cartridge.

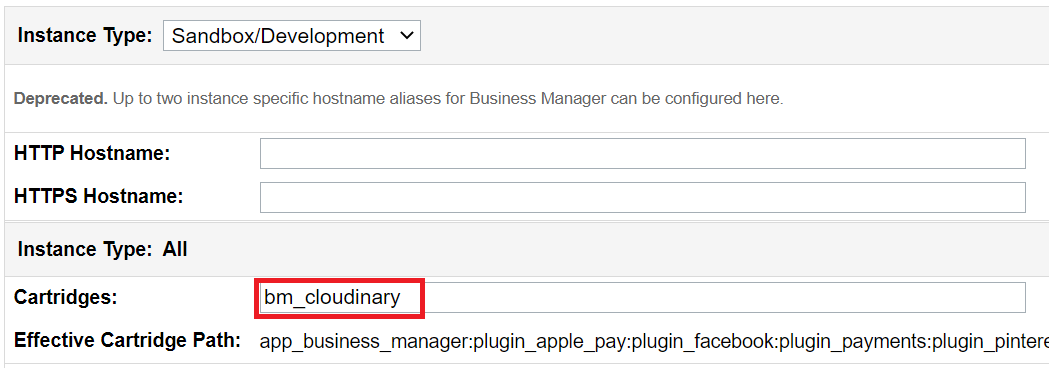
### 3.1.2 Setting Up Business Manager

1.  Go to Business Manager > Site > Manage Sites. Select the correct site, then select the  **Settings** tab. In the cartridge path add the int\_cloudinary int\_cloudinary\_sg and int\_cloudinary\_testsuite (optional) cartridges:

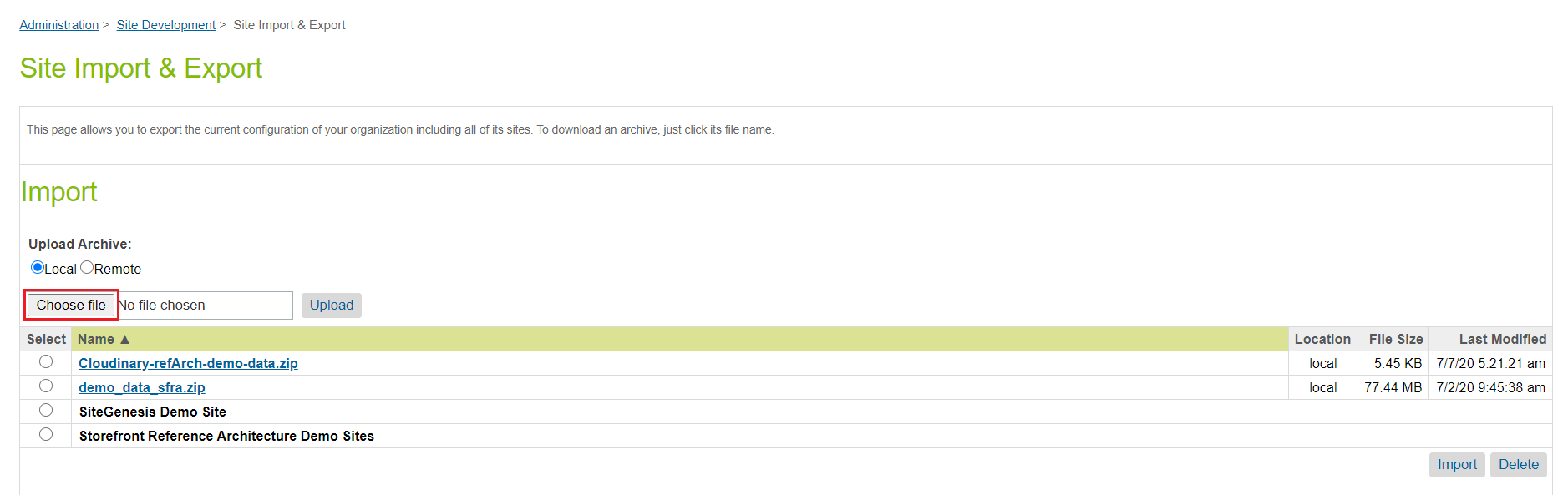


**Note:** Theint\_cloudinary\_testsuite cartridge is optional; it includes test harnesses for all script APIs. You may also include the cloudinary\_sg\_changes cartridge if you want to see how everything is integrated on different pages, but it’s not recommended to use it on a production environment.

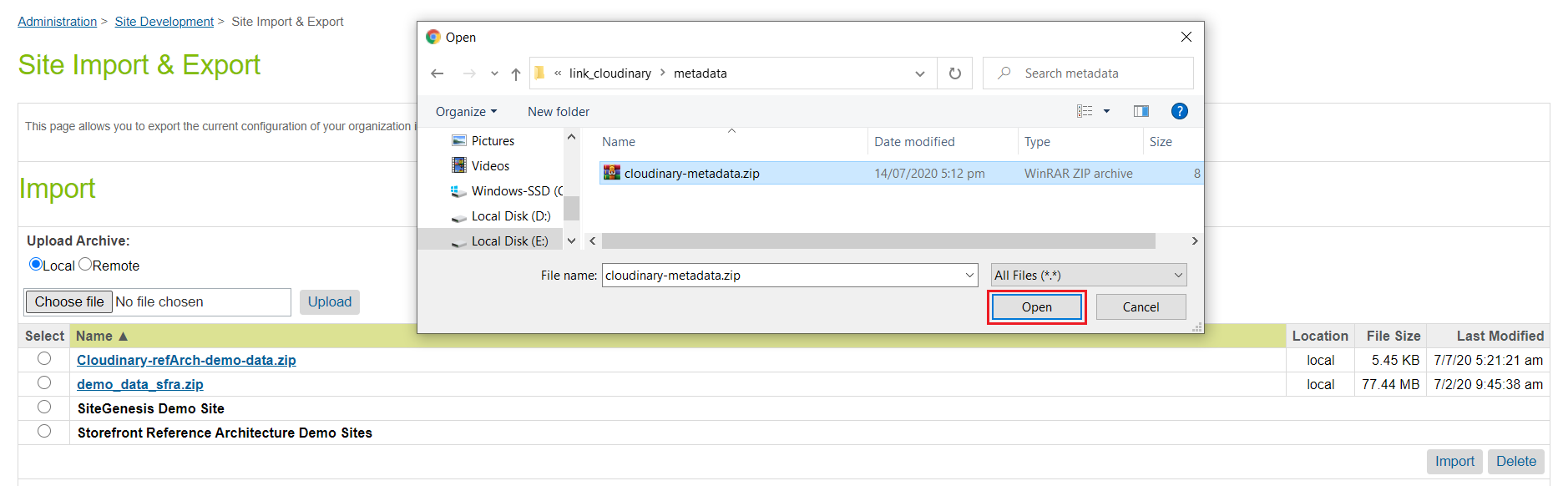
Add bm\_Cloudinary cartridge to the cartridge path in Administration > Sites > Manage Sites > Business Manager - Settings:



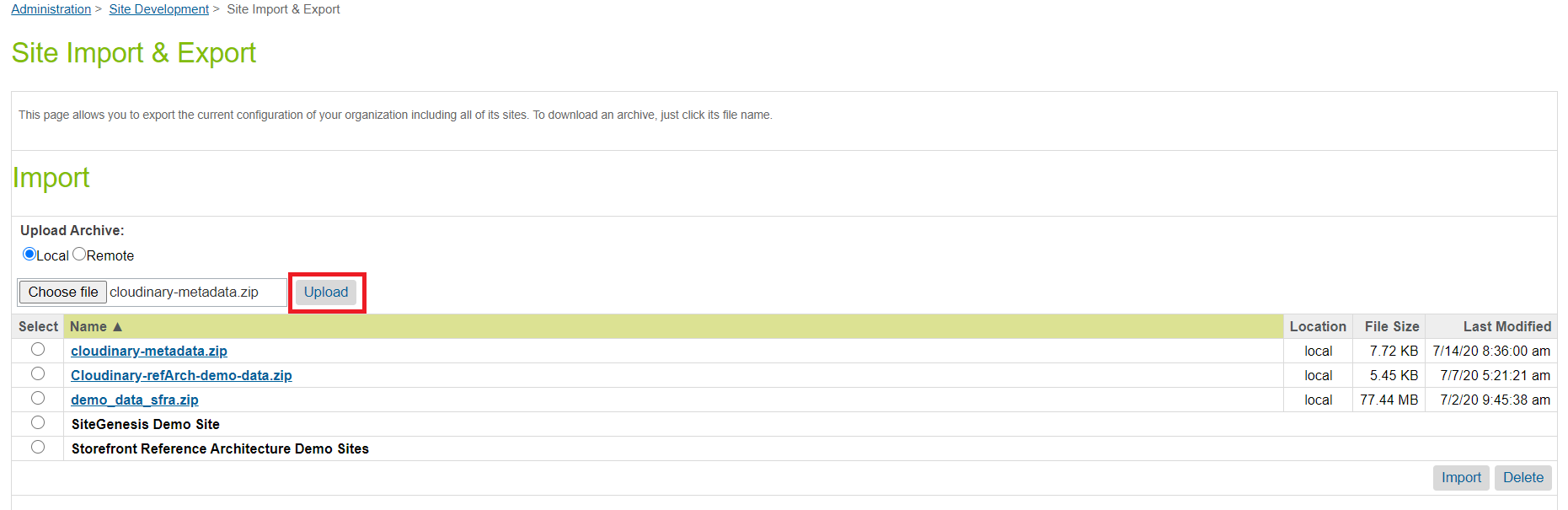
2.  Go to Administration > Site Development > Site Import & Export. In the Import section click on Choose File:



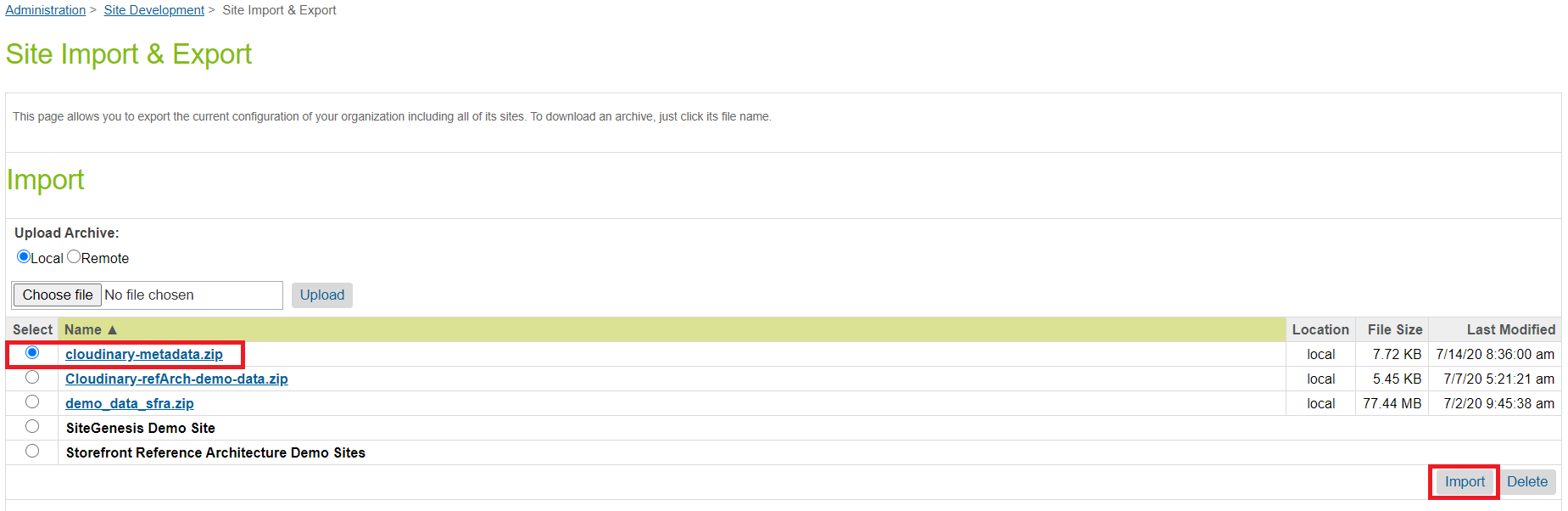
3.  Select cloudinary-metadata.zip in the metadata folder and clickOpen.



4. Click Upload.



5. Select cloudinary-metadata.zip in the Import section and click Import.

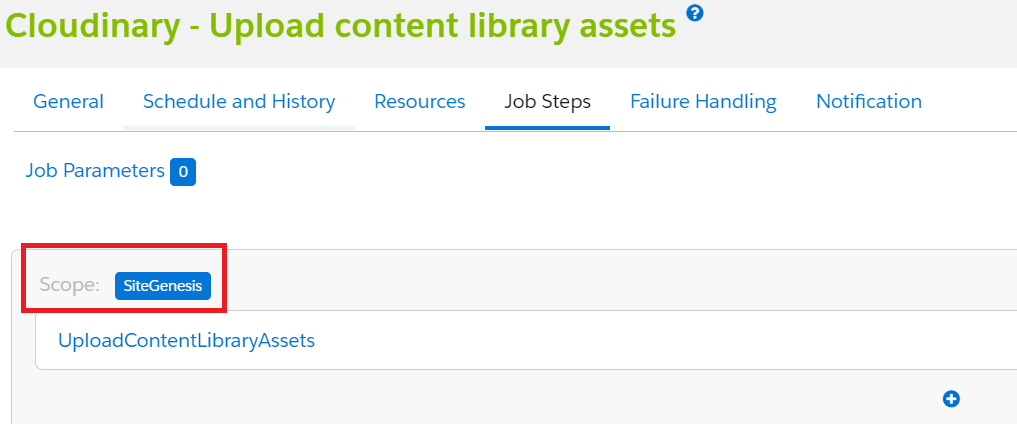


6. Click on OK.



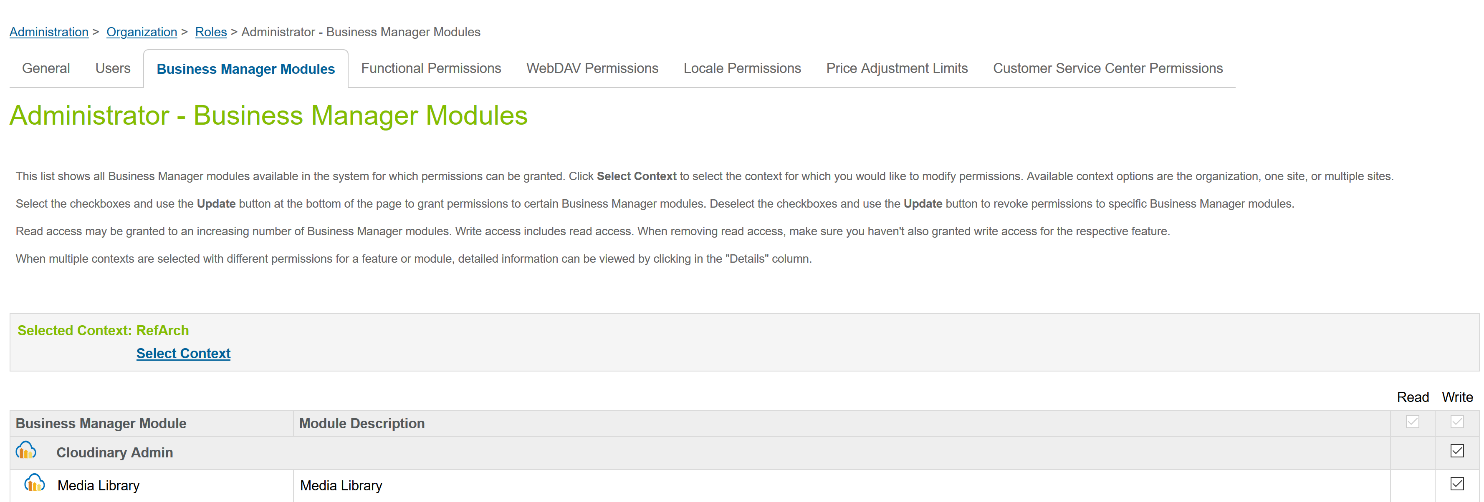
**Note**: There is a folder called “Sample content” inside “metadata” which contains sample data for specific sites. If you want to import sample data for SiteGenesis then import “SiteGenesis.zip” file by following the above process.

7. In Administration > Operations > Job Schedules select any Cloudinary job e.g. “Cloudinary - Upload content library assets”. Go into the Job Steps tab and update the scope to the site name. Repeat this process for all other Cloudinary jobs listed in section [3.5](#_heading=h.4f1mdlm).



**Note:** For the “Cloudinary - Upload content library assets” job, we recommend assigning only one site if the library is shared across multiple sites; however, if it *is* shared across multiple sites the job will process that library content multiple times but not duplicate content in the Cloudinary DAM.

9. Go to Administration > Organization > Roles & Permissions and select the specific user role (Administrator). Select the “Business Manager Modules” tab, then from “Select Context” choose your site and click Apply. In the list of “Business Manager Modules” find “Cloudinary Admin” and click to update the permissions for Media Library and click “Update”.



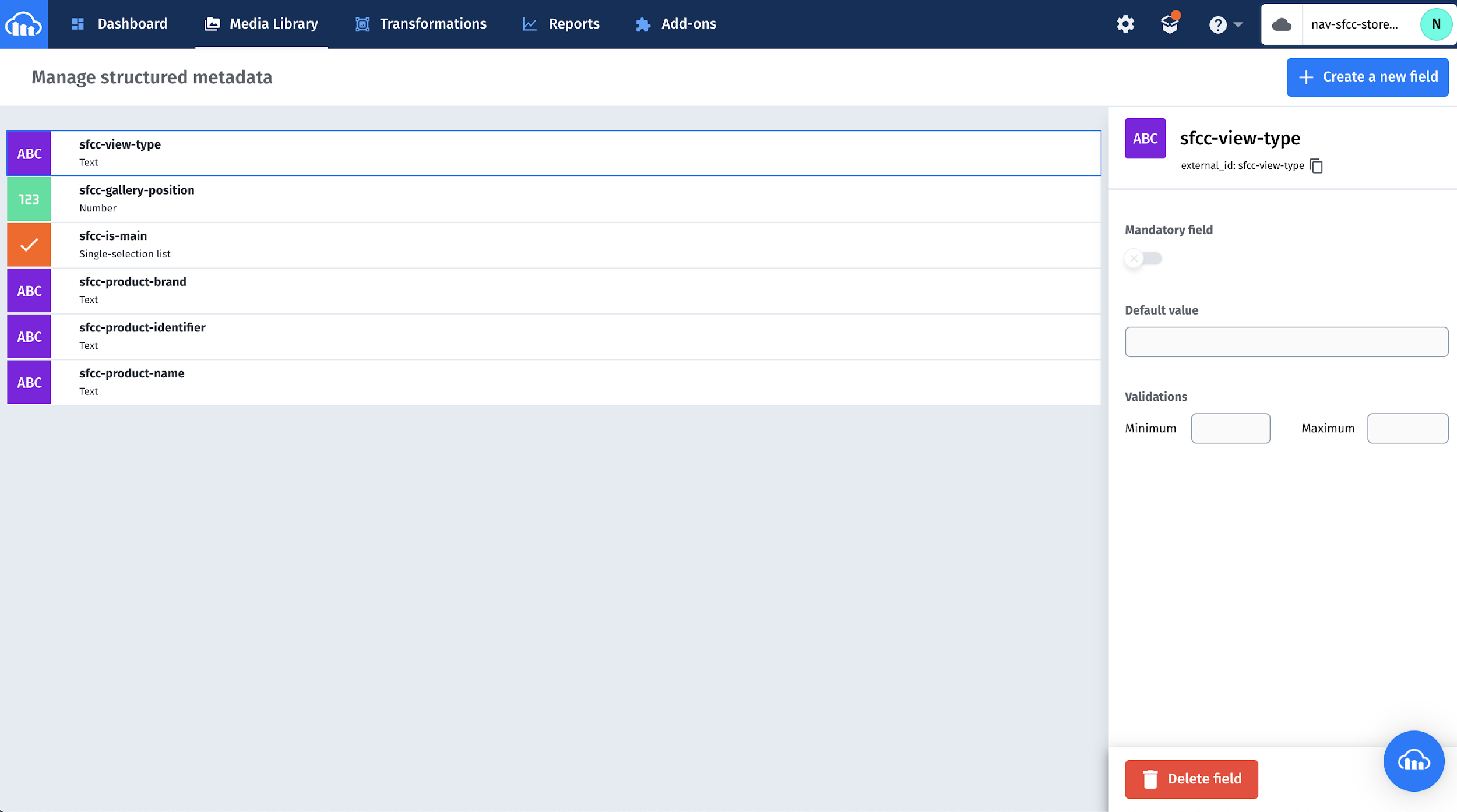
## 3.2 Configuring Cloudinary

### 3.2.1 Adding Structured Metadata in Cloudinary

Create the following structured metadata fields in your Cloudinary account. For instructions on creating structured metadata, see the Cloudinary documentation: <https://cloudinary.com/documentation/dam_media_library_administration#structured_metadata_management>.

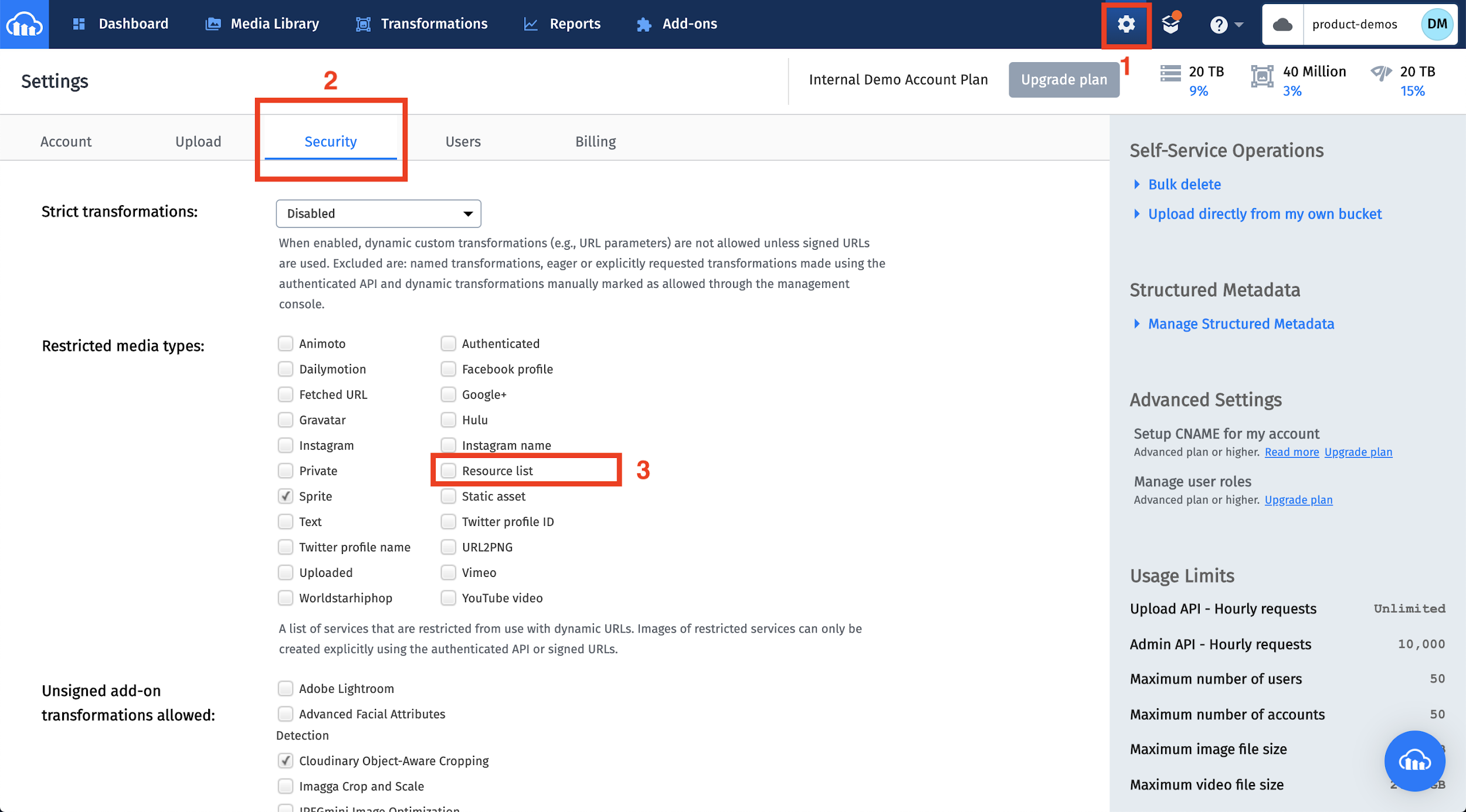
Note: Customize the external ID and the label of each field to match this table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| External ID | Label | Type | Required? | Used for |
| sfcc-product-identifier | sfcc-product-identifier | Text | Yes | Stores the product ID. |
| sfcc-is-main | sfcc-is-main | Single selection list  (Values: True, False) | Yes | Indicates which product image is used in the product listing. |
| sfcc-gallery-position | sfcc-gallery-position | Number | Yes | Indicates the position of the image in the product gallery. |
| sfcc-view-type | sfcc-view-type | Text | Yes | Stores the view type in SFCC. Required for swatches. |
| sfcc-product-name | sfcc-product-name | Text | Optional | Stores the product name, useful for searching in the DAM. |
| sfcc-product-brand | sfcc-product-brand | Text | Optional | Stores the product brand, useful for searching in the DAM. |



### 3.2.2 Enabling Resource Listing

In your account’s Security settings, enable public resource listing, by clearing the Resource list checkbox in Restricted media types.



## 3.3 Configuring the Cartridge Site Preferences:

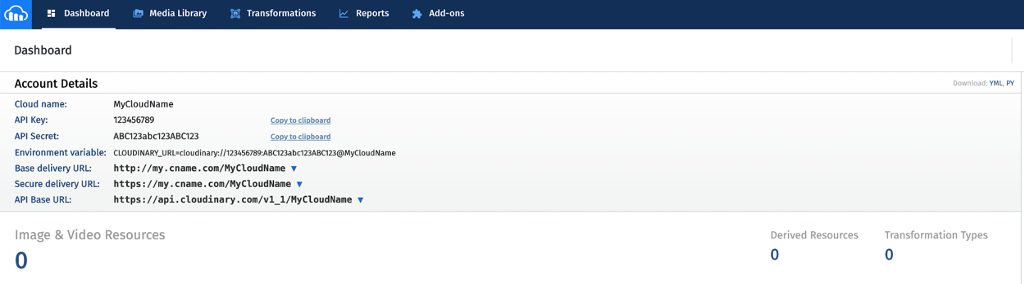
There are four Cloudinary custom preferences groups as follows:

**3.3.1 Cloudinary Core Configurations:**

Go to Business Manager: Sites > Your Site > Site Preferences > Custom Site Preferences > Cloudinary Core Configurations.

To find your Cloudinary account specific information:

1. Log in to the Cloudinary console at <https://cloudinary.com/console>. Use your Cloudinary credentials to log in.
2. Navigate to the Dashboard to see your Cloudinary account details.



|  |  |
| --- | --- |
| **Preference Name** | **Description** |
| Cloudinary Enabled | Enables or disables the Cloudinary cartridge for the site. |
| Cloudinary Base Delivery Path | The base delivery path configured for your Cloudinary account. |
| Cloudinary Cloud Name | The cloud name configured for your Cloudinary account. |
| Cloudinary API Key | The API key configured for your Cloudinary account. |
| Cloudinary API Secret | The API secret configured for your Cloudinary account. |
| Cloudinary Image Page Type Settings | The page types for Cloudinary responsive images on different pages. You can configure any page types of your choice but the ones shown below are mandatory for SiteGenesis pages and these are hardcoded in the cartridge’s constants file.  {  "pdp": {},  "plp": {},  "searchSuggestions": {},  "cldPlpSwatch": {},  "cldPdpSwatch": {},  "cldProductNav": {},  "cart": {},  "quickview": {},  "miniCart": {},  "checkout": {},  "categoryBanner": {},  "recommendationTile": {},  "cldhomePageProductTile": {},  "cldCartProductTile": {}  }  Note: Although these page type keys are mandatory, you may keep them empty if you don’t want responsive images on specific pages. You may also define your own page types for your custom pages.  If you want to make images responsive on a page, for example, “plp”, then its json must have the following structure:  "plp": {  "automateResponsivenessWithJS": true,  "autoResponsiveDimensions": "w\_auto,c\_scale",  "dimensions": "w\_250",  "breakpoints": {  "size200w" : {  "style": "(max-width: 480px) 200px",  "width": "200" ,  "height": "300"  },  "size113w" : {  "style": "(max-width: 768px) 113px",  "width": "200" ,  "height": "150"  }  }  }  By using page types you can achieve responsiveness in two ways,  **(1)** **Generate html srcset and sizes attributes for <img> tag**  To enable this feature, set “automateResponsivenessWithJS” to false. It will then read configurations for all defined breakpoints and generate the required HTML attributes for the <img> tag. For this “dimensions” and ”breakpoints” are mandatory and you may exclude “autoResponsiveDimensions” from the settings.  **(2) Use Cloudinary responsiveness**  If you want to handle image responsiveness using Cloudinary client side libraries then set “automateResponsivenessWithJS” to true. For this “autoResponsiveDimensions” is mandatory and you may exclude “dimensions” and “breakpoints” from the settings.  Use these page types to make images responsive on different pages:   * **pdp**: product details page * **plp**: product tiles on the product listing page * **searchSuggestions**: suggested products under the search bar * **cldPlpSwatch**: swatch images on the product listing page * **cldPdpSwatch**: swatch images on the product details page * **cldProductNav**: product navigation on product details page * **cart**: product cards on the cart page * **quickview**: quick view modal * **miniCart**: mini cart drop down * **checkout**: all parts of the checkout page including shipping, billing and placing an order * **categoryBanner**: category banner image * **recommendationTile**: recommendations tile on product details page. * **cldhomePageProductTile**: product tiles on home page. * **cldCartProductTile**: last visited products tile on cart page. |
| Cloudinary Core Shrinkwrap Static JS URL | The URL for the Cloudinary Shrinkwrap client side JS library. |
| Cloudinary Gallery Static JS URL | The URL for the Cloudinary Product Gallery Widget client side JS library. |
| Cloudinary Video Player Static JS URL | The URL for the Cloudinary Video Player client side JS library. |
| Cloudinary Video Player Static CSS URL | The URL for the Cloudinary Video Player client side CSS library. |
| Base folder for Catalog Images in SFCC | This is the base path for images, configured at the catalog level. Copy this path from your Site Catalog Settings > Image Settings > Base Path. |
| View type for high res images in the catalog | The view type used for the highest resolution of images in the catalog, e.g.: “large”. |
| View type for swatch images in the catalog | The view type used for swatches in the catalog, e.g.: “swatch”. |
| Folder for catalog images in Cloudinary | The folder in the Cloudinary DAM, where catalog images will be stored. |
| Folder For Content Images In Cloudinary | The folder in the Cloudinary DAM, where catalog images will be stored. |
| Folder For Product Images In Cloudinary | The folder in the Cloudinary DAM, where product images will be stored. |
| Folder For Product Videos In Cloudinary | The folder in the Cloudinary DAM, where product videos will be stored. |
| Cloudinary Rest Service Credentials ID | This is the credentials id used by Cloudinary rest service. See the [section 3.6](#_heading=h.2u6wntf) for more details. |

**3.3.2 Cloudinary Asset Management Configurations:**

Go to Business Manager: Sites > Your Site > Site Preferences > Custom Site Preferences > Cloudinary Asset Management Configurations.

|  |  |
| --- | --- |
| **Preference Name** | **Description** |
| Cloudinary Cartridge Operation Mode | Controls how assets are mapped to products. |
| Cloudinary Auto Upload Mapping | The folder configured in the Cloudinary DAM for auto upload mapping. For information on auto upload, see <https://cloudinary.com/documentation/fetch_remote_images#auto_upload_remote_resources>. |
| Cloudinary Cartridge Content Operation Mode | Controls how content assets are mapped to Cloudinary images and videos. |
| Use Cloudinary Gallery On PDPs | Enables or disables the Cloudinary Gallery Widget on PDPs. |
| Cloudinary Gallery Look And Feel | Controls the global look and feel for the Cloudinary gallery on PDPs.  Using the gallery studio generates config which looks like below <https://demo.cloudinary.com/product-gallery/?editor>  {  "container": "#cld-gallery",  "cloudName": "demo",  "mediaAssets": [{  "tag": "shoes\_product\_gallery\_demo",  "mediaType": "image"  }, {  "tag": "shoes\_product\_gallery\_demo",  "mediaType": "video"  }, {  "tag": "shoes\_product\_gallery\_demo\_spinset",  "mediaType": "spin"  }],  "displayProps": {  "mode": "expanded",  "columns": 2,  "spacing": 15  },  "aspectRatio": "3:4",  "transformation": {  "crop": "fill"  },  "bgColor": "transparent",  "carouselLocation": "left",  "carouselOffset": 10,  "navigation": "always",  "thumbnailProps": {  "mediaSymbolSize": 42,  "spacing": 20,  "width": 90,  "height": 90,  "navigationFloat": true,  "navigationShape": "square",  "navigationSize": 40,  "navigationColor": "#ffffff",  "selectedStyle": "border",  "selectedBorderPosition": "bottom",  "selectedBorderWidth": 4  },  "navigationButtonProps": {  "shape": "rectangle",  "iconColor": "#ffffff",  "color": "#000",  "size": 52,  "navigationPosition": "offset",  "navigationOffset": 12  },  "themeProps": {  "primary": "#000000",  "active": "#777777"  }  }  This custom preference expects this configuration without the cloudName and mediaAssets properties as shown below.  {  "container": "#cld-gallery",  "displayProps": {  "mode": "expanded",  "columns": 2,  "spacing": 15  },  "aspectRatio": "3:4",  "transformation": {  "crop": "fill"  },  "bgColor": "transparent",  "carouselLocation": "left",  "carouselOffset": 10,  "navigation": "always",  "thumbnailProps": {  "mediaSymbolSize": 42,  "spacing": 20,  "width": 90,  "height": 90,  "navigationFloat": true,  "navigationShape": "square",  "navigationSize": 40,  "navigationColor": "#ffffff",  "selectedStyle": "border",  "selectedBorderPosition": "bottom",  "selectedBorderWidth": 4  },  "navigationButtonProps": {  "shape": "rectangle",  "iconColor": "#ffffff",  "color": "#000",  "size": 52,  "navigationPosition": "offset",  "navigationOffset": 12  },  "themeProps": {  "primary": "#000000",  "active": "#777777"  }  } |
| Cloudinary Videos Enabled | Enables videos from Cloudinary on the site, globally. |
| Use Cloudinary Video Player | Enables the Cloudinary video player on the site. Cloudinary Videos Enabled should be true for this to work. |
| Cloudinary Include Videos By View Type In PGW | Includes videos fetched by view type in the Product Gallery, when Get Assets By View Type is chosen for Cloudinary Cartridge Operation Mode. |
| Cloudinary Video Player Style | Controls the global look and feel for the Cloudinary video player.  You can use the Cloudinary video studio (<https://studio.cloudinary.com/>) to generate the configuration, which includes the player style (highlighted here):  const player = cld.videoPlayer('player', {  "fluid": true,  "controls": true,  "fontFace": "Lobster",  "colors": {  "base": "#de1e1e",  "accent": "#180cff",  "text": "#222121"  }  });  player.source("elephants", {});  This custom preference expects the configuration for the player’s style only. Therefore, in this case, enter:    {  "fluid": true,  "controls": true,  "fontFace": "Lobster",  "colors": {  "base": "#de1e1e",  "accent": "#180cff",  "text": "#222121"  }  } |
| Cloudinary Use Video Custom Mapping | Enables or disables custom mapping for product videos in Cloudinary. If enabled, this overrides cartridge operation mode settings for videos. |
| Cloudinary Video Custom Map Path | The custom mapping path scheme for Cloudinary videos only, based on video naming conventions in Cloudinary. It is required if Cloudinary Use Video Custom Mapping is enabled. |
| Cloudinary Custom Mapping Video Format | Appends video format while building URLs by using custom mapping. |
| Product identifier used for tags in Cloudinary | The product attribute identifier which should be used to create tags in Cloudinary, e.g. standard attribute like ID or a custom attribute in the catalog. |
| Enable Cloudinary Swatches On PLP | Shows or hides swatch images from Cloudinary on PLP. |
| Cloudinary Metadata Filtering Endpoint | The Cloudinary dynamic endpoint to fetch resources based on metadata and tag name. |
| Cloudinary Structured Metadata Key | The Cloudinary structured metadata key representing a primary image. |
| Cloudinary Structured Metadata Value | The Cloudinary structured metadata value representing a primary image. |

**3.3.3 Cloudinary Transformation Configurations:**

Go to Business Manager: Sites > Your Site > Site Preferences > Custom Site Preferences > Cloudinary Transformation Configurations

|  |  |
| --- | --- |
| **Preference Name** | **Description** |
| Cloudinary Global Image Transformation | Transformations that are applied to all images delivered by Cloudinary. |
| Global Image Format | The default format for all images delivered by Cloudinary. |
| Global Image DPR | The default Device Pixel Ratio (DPR) setting for all images delivered by Cloudinary. |
| Global Image Quality | The default quality setting for all images delivered by Cloudinary. |
| Cloudinary Video Transformation | Transformations that are applied to all videos delivered by Cloudinary. |
| Global Video Format | The default format for all videos delivered by Cloudinary. |
| Global Video Quality | The default quality setting for all videos delivered by Cloudinary. |

**3.3.4 Cloudinary Jobs Configurations:**

Go to Business Manager: Sites > Your Site > Site Preferences > Custom Site Preferences > Cloudinary Jobs Configurations

|  |  |
| --- | --- |
| **Preference Name** | **Description** |
| Content Library Job Last Execution Date | Stores the last date the ‘Cloudinary - Upload content library assets’ job ran. |
| Catalog Content Job Last Execution Date | Stores the last date the ‘Cloudinary - Upload catalog assets’ job ran. |
| Maximum upload limit for assets in MBs | Limits the size of assets uploaded to Cloudinary. Any assets larger than this limit are skipped. |
| Upload Preset | Sets the upload preset while uploading assets on Cloudinary DAM. This preset should exist in the Cloudinary DAM. Leave this empty if an upload preset is not required. |

## 3.4 System Object Types:

Cloudinary metadata contains custom attributes for System Object Types; these are added after successful import of metadata.

**3.4.1 Catalog Custom Attributes Details:**

Navigate to Administration > System Object Types > Catalog. After successful metadata import these attributes are added under *Cloudinary Site Cartridge*.

|  |  |
| --- | --- |
| **Attribute Name** | **Description** |
| Cloudinary Image Transformation | Transformations used to transform Cloudinary images. This attribute is used for two types of image assets.   1. **Catalog images:** When cartridge tries to fetch the catalog images for category banners then this attribute is used to add image transformations specified on catalog level in the URL. The transformations format for image transformations will be as **global\_image\_tranformations/catalog\_image\_tranformations/category\_image\_tranfromations**. If this attribute is empty then it skips adding catalog level image transformations in the URL.   **Note:** It doesn’t override global image transformations specified at level.   1. **Product Images:** When cartridge tries to fetch product images then value in this attribute for transformations will be overridden by transformations specified at product level. |
| Cloudinary Video Transformation | Transformations used to transform Cloudinary videos. This attribute is used for two types of video assets.   1. **Catalog videos:** While fetching catalog videos, these transformations are used to add in the URL. The format for transformation will be as **global\_video\_transformations/cataog\_video\_transformations/category\_video\_transformation.** If this attribute is empty then it skips adding catalog level video transformations in the URL.   **Note:** It doesn’t override global video transformations specified at site level.   1. **Product videos:** When cartridge tries to fetch product videos then value in this attribute for transformations will be overridden by transformations specified at product level. |

**3.4.2 Category Custom Attributes Details:**

Navigate to Administration > System Object Types > Category. After successful metadata import these attributes are added under *Cloudinary*.

|  |  |
| --- | --- |
| **Attribute Name** | **Description** |
| Cloudinary Image Transformation | Transformations used to transform Cloudinary images. When cartridge tries to fetch the catalog images for category banners then this attribute is used to add image transformations specified on category level in the URL. The transformations format for image transformations will be as **global\_image\_tranformations/catalog\_image\_tranformations/category\_image\_tranfromations**.  If this attribute is empty then it skips adding category level image transformations in the URL.  **Note:** It doesn’t override global image transformations specified at level. |
| Cloudinary Video Transformation | Transformations used to transform Cloudinary videos. While fetching catalog videos, these transformations are used to add in the URL. The format for transformation will be as **global\_video\_transformations/cataog\_video\_transformations/category\_video\_transformation.**  If this attribute is empty then it skips adding category level video transformations in the URL.  **Note:** It doesn’t override global video transformations specified at level. |

**3.4.3 Product Custom Attributes Details:**

Navigate to Administration > System Object Types > Product. After successful metadata import these attributes are added under *Cloudinary Site Cartridge*.

|  |  |
| --- | --- |
| **Attribute Name** | **Description** |
| Cloudinary Tag Name | A tag configured in the Cloudinary DAM. It overrides the global setting for tag name specified on the site level. |
| Cloudinary Use Video | Enables or disables use of Cloudinary video. If we want to turn off the Cloudinary video for specific products then we can set it to No and the video will be disabled. It overrides the global setting specified on the site level. |
| Cloudinary Use Video Player | Enables or disables use of the Cloudinary video player. If we want to turn off the Cloudinary video player for specific products then we can set it to No and the video player will be disabled. It overrides the global setting specified on the site level. |
| Cloudinary video player options | Enables or disables Cloudinary video player options. It overrides the global setting specified on the site level. |
| Cloudinary Gallery Styles | Applies or removes Cloudinary gallery styles. It overrides the global setting specified on the site level. |
| Cloudinary Image Transformation | Transformations used to transform Cloudinary images. It overrides the image transformations specified on catalog level. |
| Cloudinary Video Transformation | Transformations used to transform Cloudinary videos. It overrides the video transformations specified on catalog level. |

**3.4.4 Library Custom Attributes Details:**

Navigate to Administration > System Object Types > Library. After successful metadata import these attributes are added under *Cloudinary Site Cartridge*.

|  |  |
| --- | --- |
| **Attribute Name** | **Description** |
| Cloudinary Image Transformation | Transformations used to transform Cloudinary images.While fetching library images, these transformations are used to add in the URL. The format for transformation will be as **global\_image\_transformations/library\_image\_transformations.**  **Note:** It doesn’t override global transformations specified on site level. |
| Cloudinary Video Transformation | Transformations used to transform Cloudinary videos. While fetching library videos, these transformations are used to add in the URL. The format for transformation will be as **global\_video\_transformations/library\_video\_transformations.**  **Note:** It doesn’t override global transformations specified on site level. |
| Cloudinary Use Video | Enables or disables Cloudinary video. If we want to turn off the Cloudinary video for a specific library then we can set it to No and the video will be disabled. It overrides the global setting specified on the site level. |
| Use the Cloudinary video player | Enables or disables use of the Cloudinary video player. If we want to turn off the Cloudinary video player for a specific library then we can set it to No and the video player will be disabled. It overrides the global setting specified on the site level. |
| Cloudinary default video player options | Options used by the Cloudinary video player. It overrides the global setting specified on the site level. |

## 3.5 Jobs Details:

Navigate to Administration > Jobs. After successful import of metadata, these jobs are added.

**Job:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Cloudinary - Delta Upload catalog assets | Uploads recently changed catalog assets from SFCC to Cloudinary. |

**Steps:**

|  |  |  |
| --- | --- | --- |
| **Job Step Name** | **Parameter Name** | **Description** |
| UploadCatalogAssets | CLDAssetRenameReportEmail | The email address used to receive the assets rename notification. |
|  | CLDNumberOfAssets | Specifies how many assets are uploaded to Cloudinary in Debug mode. |
|  | CLDJobExecutionMode | Configures the job execution mode. In Prod mode, all assets are uploaded. In Debug mode, the number of assets configured in CLDNumberOfAssets are uploaded. |
|  | CLDSyncMode | Specifies whether to run the job in Full or Delta mode. |

**Job:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Cloudinary - Delta upload library assets | Uploads recently changed content library asset files from SFCC to Cloudinary. |

**Steps:**

|  |  |  |
| --- | --- | --- |
| **Job Step Name** | **Parameter Name** | **Description** |
| UploadContentLibraryAssets | CLDAssetRenameReportEmail | The email address used to receive the assets rename notification. |
|  | CLDNumberOfAssets | Specifies how many assets are uploaded to Cloudinary in Debug mode. |
|  | CLDJobExecutionMode | Configures the job execution mode. In Prod mode, all assets are uploaded. In Debug mode, the number of assets configured in CLDNumberOfAssets are uploaded. |
|  | CLDSyncMode | Specifies whether to run the job in Full or Delta mode. |

**Job:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Cloudinary - Upload catalog assets | Uploads catalog assets from SFCC to Cloudinary. |

**Steps:**

|  |  |  |
| --- | --- | --- |
| **Job Step Name** | **Parameter Name** | **Description** |
| UploadCatalogAssets | CLDAssetRenameReportEmail | The email address used to receive the assets rename notification. |
|  | CLDNumberOfAssets | Specifies how many assets are uploaded to Cloudinary in Debug mode. |
|  | CLDJobExecutionMode | Configures the job execution mode. In Prod mode, all assets are uploaded. In Debug mode, the number of assets configured in CLDNumberOfAssets are uploaded. |
|  | CLDSyncMode | Specifies whether to run the job in Full or Delta mode. |

**Job:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Cloudinary - Upload content library assets | Uploads content library asset files from SFCC to Cloudinary. |

**Steps:**

|  |  |  |
| --- | --- | --- |
| **Job Step Name** | **Parameter Name** | **Description** |
| UploadContentLibraryAssets | CLDAssetRenameReportEmail | The email address used to receive the assets rename notification. |
|  | CLDNumberOfAssets | Specifies how many assets are uploaded to Cloudinary in Debug mode. |
|  | CLDJobExecutionMode | Configures the job execution mode. In Prod mode, all assets are uploaded. In Debug mode, the number of assets configured in CLDNumberOfAssets are uploaded. |
|  | CLDSyncMode | Specifies whether to run the job in Full or Delta mode. |

**Job:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| Cloudinary - Upload product assets | Uploads product asset files from SFCC to Cloudinary. |

**Steps:**

|  |  |  |
| --- | --- | --- |
| **Job Step Name** | **Parameter Name** | **Description** |
| UplaodProductLargeImages | CLDAssetRenameReportEmail | The email address used to receive the assets rename notification. |
|  | CLDNumberOfAssets | Specifies how many assets are uploaded to Cloudinary in Debug mode. |
|  | CLDJobExecutionMode | Configures the job execution mode. In Prod mode, all assets are uploaded. In Debug mode, the number of assets configured in CLDNumberOfAssets are uploaded. |
|  | CLDSyncMode | Specifies whether to run the job in Full or Delta mode. |
|  | CLDCatalogIds | Specifies from which catalogs the assets are uploaded. If empty, then the catalog assigned to the site is used for uploading. |
|  | CLDViewType | Specifies which view type is uploaded to Cloudinary. |
| UploadProductSwatchImages | CLDAssetRenameReportEmail | The email address used to receive the assets rename notification. |
|  | CLDNumberOfAssets | Specifies how many assets are uploaded to Cloudinary in Debug mode. |
|  | CLDJobExecutionMode | Configures the job execution mode. In Prod mode, all assets are uploaded. In Debug mode, the number of assets configured in CLDNumberOfAssets are uploaded. |
|  | CLDSyncMode | Specifies whether to run the job in Full or Delta mode. |
|  | CLDCatalogIds | Specifies from which catalogs the assets are uploaded. If empty, then the catalog assigned tothe site is used for uploading. |
|  | CLDViewType | Specifies which view type is uploaded to Cloudinary. |

Note: You can configure multiple job steps to upload assets from different view types; we have already configured two steps used to upload “large” and “swatch” view type images. For example, follow these steps to add a new job step to upload videos from the view type for videos, if it exists in the catalog.

1- Go to Administration -> Operations -> Jobs.

2- Select the job you want to change.

3- Select the Job Steps tab.

4- Click the + button to add a step.

5- Select ‘Custom.Cloudinary.UploadProductAssets’ and configure it.

6- Enter an ID for this step e.g ‘UploadProductVideos’.

7- Enter your email into the ‘CLDAssetRenameReportEmail’ field.

8- Select ‘CLDJobExecutionMode’ as required.

9- Select ‘CLDSyncMode’ as required.

10- Enter ‘video’ in ‘CLDViewType’ to associate this step with the video view type.

## 3.6 Services Details:

Navigate to Administration > Services. After successful import of metadata the following is added.

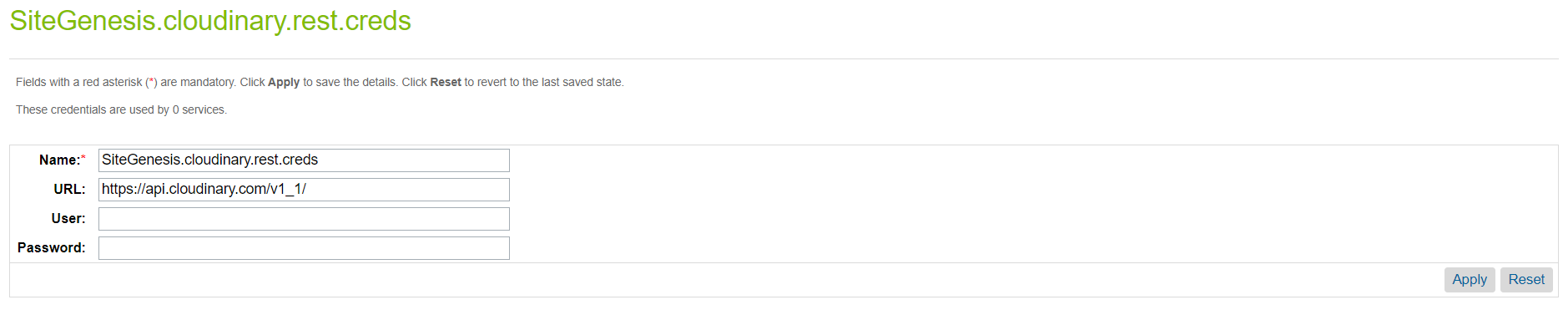
|  |  |  |
| --- | --- | --- |
| **Name** | **Profile** | **Credentials** |
| cloudinary.rest.upload | cloudinary.rest.upload.profile | Need to create site specific credentials. e.g. SiteGenesis.cloudinary.rest.creds |
| cloudinary.rest.list | cloudinary.rest.list.profile | cloudinary.rest.list.creds |

Configure the URL, User and Password in SiteGenesis.cloudinary.rest.creds

and cloudinary.rest.list.creds. You can find all your account details in your Cloudinary console Dashboard, at <https://cloudinary.com/console>.

For SiteGenesis.cloudinary.rest.creds:

1. Enter your base upload API URL in the URL field.
2. Enter your API Key in the User field.
3. Enter your API Secret in the Password field.



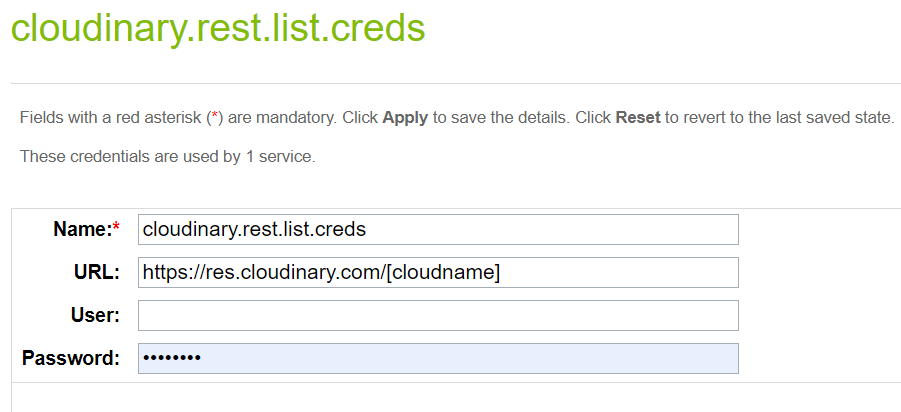
For cloudinary.rest.list.creds:

1. Enter your base delivery URL in the URL field.
   * If you have a private CDN, this will be of the form: https://<cloud\_name>-res.cloudinary.com
   * If you have a Custom Domain Name (CNAME), this will be of the form:

https://<custom domain name>

* + If it’s simple Cloudinary list API URL then we also need to add a placeholder for cloudname i.e. [cloudname]: https://res.cloudinary.com/[cloudname]

1. The User and Password fields do not need to be set.



## 3.7 Cartridge Operation Modes:

The cartridge can operate in three different modes:

* + 1. **Map Assets using tags and metadata in Cloudinary:**

Note: if you intend to use this option, contact Cloudinary support (<https://support.cloudinary.com/hc/en-us>) to set up a special endpoint that will enable media to be delivered based on metadata and tags.

* Cloudinary is the single source of all media for all catalogs.
* For the product catalog, assets in Cloudinary are mapped to products using tags and structured metadata.

***Note****: For product variations, only images are supported, not videos. Both are supported for the master product.*

* For the content and site catalogs, specify URLs to Cloudinary assets in content slots.
* If migrating from an existing site, you can run jobs to bulk upload your media to Cloudinary. Tags and metadata are automatically added to the assets in Cloudinary for the product catalog.
  + 1. **Map assets using view types in the catalog:**
* SFCC is the source of mapping for all media for all catalogs.
* For the product catalog, assets are expected to reside in Cloudinary using the same paths as in SFCC, so that assets are ultimately delivered from Cloudinary. You can run jobs to upload and synchronize your media residing in SFCC to Cloudinary.
* For the content and site catalogs, specify URLs to Cloudinary assets in content slots.
  + 1. **Use existing website URLs to Auto upload into Cloudinary and deliver:**
* Media is automatically uploaded from its existing location to Cloudinary, when requested by the end user, and subsequently delivered from Cloudinary. The auto-upload functionality is described here: <https://cloudinary.com/documentation/fetch_remote_images#auto_upload_remote_resources>
* This option is not recommended, but exists to provide a quick and simple way to deliver your media from Cloudinary.

## 3.8 Code Changes

The following section highlights the required updates to be made to your storefront application code. Code and code placement is based on the SFCC reference application – Site Genesis. Highly customized sites may need to alter/move this code to suit their specific needs.

Cloudinary\_sg\_changes cartridge has overridden templates, controllers, scripts and client side js files along with comments to identify custom code changes for Cloudinary.

**3.8.1 Make changes in client side JS files**

Open cloudinary\_sg\_changes cartridge and go to the “js” folder, it has all overridden client side js files to integrate Cloudinary.

1. Open file js/pages/product/addToCart.js and copy code mentioned below and place in your core cartridge addToCart.js file on the same path.

From line no. 53 to 57

1. // Custom start: Update cloudinary responsive image //
2. **if** (**typeof** window.makeCloudinaryImagesResponsive !== 'undefined') {
3. window.makeCloudinaryImagesResponsive();
4. }
5. // Custom end: Update cloudinary responsive image //
6. Open file js/pages/product/image.js and copy code mentioned below and place in your core cartridge image.js file on the same path.

From line no. 60 to 67

1. // Custom Start: Replace pinterest button url //
2. **var** primaryImg = document.querySelector('#pdpMain .primary-image');
3. **if** (primaryImg && primaryImg.length > 0) {
4. newUrl = primaryImg.getAttribute('src');
5. } **else** {
6. newUrl = document.querySelector('.share-icon[data-share=pinterest]').getAttribute('href');
7. }
8. // Custom End: Replace pinterest button url //

From line no. 88 to 101

1. // Custom start: Update cloudinary PGW //
2. **var** cloudinary = $newImages.data('cld');
3. **if** (cloudinary && cloudinary.isEnabled && cloudinary.isGalleryEnabled) {
4. **if** (typeof cldGallery !== 'undefined' && cldGallery && cloudinary.galleryWidgetOptions && cloudinary.galleryWidgetOptions.mediaAssets) {
5. **var** galleryOptions = cloudinary.galleryWidgetOptions;
6. **if** (cloudinary.container) {
7. galleryOptions.container = cloudinary.container;
8. }
9. cldGallery.update(galleryOptions);
10. }
11. } **else** {
12. $imageContainer.html($newImages.html());
13. }
14. // Custom end: Update cloudinary PGW //
15. Open file js/pages/product/productSet.js and copy code mentioned below and place in your core cartridge productSet.js file on the same path.

Line no. 6

1. image = require('./image'); // Custom : Require image module to render cloudinary PGW //

From line no. 40 to 42

1. // Custom Start : Render cloudinary PGW //
2. image.replaceImages();
3. // Custom End : Render cloudinary PGW //
4. Open file js/pages/search.js and copy code mentioned below and place in your core cartridge search.js file on the same path.

From line no. 66 to 70

1. // Custom start: Update cloudinary responsive image //
2. **if** (**typeof** window.makeCloudinaryImagesResponsive !== 'undefined') {
3. window.makeCloudinaryImagesResponsive();
4. }
5. // Custom end: Update cloudinary responsive image //
6. Open file js/product-tile.js and copy code mentioned below and place in your core cartridge product-tile.js file on the same path.

From line no. 32 to 36

1. // Custom start: Update cloudinary responsive image //
2. **if** (**typeof** window.makeCloudinaryImagesResponsive !== 'undefined') {
3. window.makeCloudinaryImagesResponsive();
4. }
5. // Custom end: Update cloudinary responsive image //

From line no. 62 to 67

1. // Custom start: Update cloudinary responsive image //
2. **if** (**typeof** window.makeCloudinaryImagesResponsive !== 'undefined') {
3. $thumb.attr('data-src', data.src);
4. window.makeCloudinaryImagesResponsive();
5. }
6. // Custom end: Update cloudinary responsive image //

From line no. 88 to 93

1. // Custom start: Update cloudinary responsive image //
2. **if** ($thumb.hasClass('cld-responsive') && **typeof** window.makeCloudinaryImagesResponsive !== 'undefined') {
3. $thumb.attr('data-src', data.src);
4. window.makeCloudinaryImagesResponsive();
5. }
6. // Custom end: Update cloudinary responsive image //

From line no. 119 to 124

1. // Custom start: Update cloudinary responsive image //
2. **if** ($thumb.hasClass('cld-responsive') && **typeof** window.makeCloudinaryImagesResponsive !== 'undefined') {
3. $thumb.attr('data-src', data.src);
4. window.makeCloudinaryImagesResponsive();
5. }
6. // Custom end: Update cloudinary responsive image //
7. Open file js/quickview.js and copy code mentioned below and place in your core cartridge quickview.js file on the same path.

From line no. 69 to 76

1. // Custom Start: Render cloudinary PGW on quickview //
2. **if** (**typeof** window.renderCloudinaryGalleryWidget !== 'undefined') {
3. window.renderCloudinaryGalleryWidget();
4. }
5. **if** (**typeof** window.renderCloudinarySetGalleryWidgets !== 'undefined') {
6. window.renderCloudinarySetGalleryWidgets();
7. }
8. // Custom End: Render cloudinary PGW on quickview //

From line no. 115 to 122

1. // Custom Start: Render cloudinary PGW on quickview //
2. **if** (**typeof** window.renderCloudinaryGalleryWidget !== 'undefined') {
3. window.renderCloudinaryGalleryWidget();
4. }
5. **if** (**typeof** window.renderCloudinarySetGalleryWidgets !== 'undefined') {
6. window.renderCloudinarySetGalleryWidgets();
7. }
8. // Custom End: Render cloudinary PGW on quickview //
9. Open file js/searchsuggest.js and copy code mentioned below and place in your core cartridge searchsuggest.js file on the same path.

From line no. 139 to 143

1. // Custom Start: Make suggested products images responsive //
2. **if** (**typeof** window.makeCloudinaryImagesResponsive !== 'undefined') {
3. window.makeCloudinaryImagesResponsive();
4. }
5. // Custom End: Make suggested products images responsive //
6. To compile js run the command ‘npm run js’. If npm is not installed then run ‘npm install’ first and then run ‘npm run js’.

**3.8.2 Make changes in Controllers**

Open cloudinary\_sg\_changes cartridge and go to the “controllers” folder, it has all overridden controllers to integrate Cloudinary on different pages.

* 1. Open file controllers/Cart.js and copy code mentioned below and place in your core cartridge Cart.js file on the same path.

From line no. 20 to 22

1. // Custom Start: Cloudinary script module //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. // Custom End: Cloudinary script module //

From line no. 279 to 287

1. // Custom Start: Add cloudinary object //
2. **var** cloudinary = {};
3. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
5. **if** (cloudinaryConstants) {
6. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
7. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.MINI\_CART;
8. }
9. // Custom End: Add cloudinary object //

From line no. 314 to 321

1. // Custom Start: Add cloudinary object //
2. **var** cloudinary = {};
3. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
4. **if** (cloudinaryConstants.CLD\_ENABLED) {
5. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
6. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.MINI\_CART;
7. }
8. // Custom End: Add cloudinary object //
   1. Open file controllers/COBilling.js and copy code mentioned below and place in your core cartridge COBilling.js file on the same path.

From line no. 82 to 91

1. // Custom Start: Add cloudinary object //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinary = {};
5. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
6. **if** (cloudinaryConstants) {
7. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
8. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.CHECKOUT;
9. }
10. // Custom End: Add cloudinary object //

From line no. 628 to 637

1. // Custom Start: Add cloudinary object //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinary = {};
5. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
6. **if** (cloudinaryConstants) {
7. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
8. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.CHECKOUT;
9. }
10. // Custom End: Add cloudinary object //
    1. Open file controllers/COShipping.js and copy code mentioned below and place in your core cartridge COShipping.js file on the same path.

From line no. 110 to 119

1. // Custom Start: Add cloudinary object //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinary = {};
5. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
6. **if** (cloudinaryConstants) {
7. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
8. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.CHECKOUT;
9. }
10. // Custom End: Add cloudinary object //
    1. Open file controllers/COShippingMultiple.js and copy code mentioned below and place in your core cartridge COShippingMultiple.js file on the same path.

From line no. 52 to 61

1. // Custom Start: Add cloudinary object //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinary = {};
5. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
6. **if** (cloudinaryConstants) {
7. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
8. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.CHECKOUT;
9. }
10. // Custom End: Add cloudinary object //

From line no. 145 to 154

1. // Custom Start: Add cloudinary object //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinary = {};
5. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
6. **if** (cloudinaryConstants) {
7. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
8. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.CHECKOUT;
9. }
10. // Custom End: Add cloudinary object //
    1. Open file controllers/COSummary.js and copy code mentioned below and place in your core cartridge COSummary.js file on the same path.

From line no. 48 to 57

1. // Custom Start: Add cloudinary object //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinary = {};
5. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
6. **if** (cloudinaryConstants) {
7. cloudinary.highResImgViewType = cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE;
8. cloudinary.pageType = cloudinaryConstants.PAGE\_TYPES.CHECKOUT;
9. }
10. // Custom End: Add cloudinary object //
    1. Open file controllers/Product.js and copy code mentioned below and place in your core cartridge Product.js file on the same path.

From line no. 16 to 20

1. // Custom Start: cloudinary script modules //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
4. **var** cloudinaryHelper = require('\*/cartridge/scripts/helpers/cloudinaryHelpers');
5. // Custom End: cloudinary script modules //

From line no. 37 to 56

1. **var** cloudinary = {};
2. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
3. **if** (cloudinaryConstants.CLD\_ENABLED) {
4. **var** colorAttrValueID = params.get('dwvar\_' + params.pid.stringValue + '\_color').stringValue;
5. cloudinary.galleryEnabled = cloudinaryConstants.CLD\_GALLERY\_ENABLED;
6. cloudinary.images = cloudinaryModel.getCloudinaryImages(product.object.ID, {
7. pageType: cloudinaryConstants.PAGE\_TYPES.PDP,
8. variationAttrValueID: colorAttrValueID
9. });
11. **var** isVideoEnabled = cloudinaryHelper.isVideoEnabled(product.object);
12. **var** isVideoPlayerEnabled = cloudinaryHelper.isVideoPlayerEnabled(product.object)
14. cloudinary.videoEnabled = isVideoEnabled;
15. cloudinary.videoPlayerEnabled = isVideoPlayerEnabled;
16. cloudinary.cloudName = cloudinaryConstants.CLD\_CLOUDNAME;
17. **if** (isVideoEnabled) {
18. cloudinary.video = cloudinaryModel.getCloudinaryVideo(product.object.ID, request.locale);
19. }
20. }

From line no. 90 to 113

1. // Custom Start: Load cloudinary assets //
2. **var** cloudinary = {};
3. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
5. **if** (cloudinaryConstants.CLD\_ENABLED) {
6. **var** colorAttrValueID = params.get('dwvar\_' + params.pid.stringValue + '\_color').stringValue;
7. cloudinary.galleryEnabled = cloudinaryConstants.CLD\_GALLERY\_ENABLED;
8. cloudinary.images = cloudinaryModel.getCloudinaryImages(product.object.ID, {
9. pageType: cloudinaryConstants.PAGE\_TYPES.PDP,
10. variationAttrValueID: colorAttrValueID
11. });
13. **var** isVideoEnabled = cloudinaryHelper.isVideoEnabled(product.object);
14. **var** isVideoPlayerEnabled = cloudinaryHelper.isVideoPlayerEnabled(product.object)
16. cloudinary.videoEnabled = isVideoEnabled;
17. cloudinary.videoPlayerEnabled = isVideoPlayerEnabled;
18. cloudinary.cloudName = cloudinaryConstants.CLD\_CLOUDNAME;
19. cloudinary.isQuickView = params.source && params.source.stringValue === 'quickview' ? **true** : **false**;
20. **if** (isVideoEnabled) {
21. cloudinary.video = cloudinaryModel.getCloudinaryVideo(product.object.ID, request.locale);
22. }
23. }
24. // Custom End: Load cloudinary assets //

From line no. 168 to 177

1. // Custom Start: Add cloudinary object //
2. **var** cloudinary = {};
3. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
5. **if** (cloudinaryConstants.CLD\_ENABLED) {
6. cloudinary.isSwachesEnabled = cloudinaryConstants.CLD\_ENABLE\_SWATCH\_ON\_PLP;
7. cloudinary.pageType = params.pageType ? params.pageType.stringValue : **null**;
8. cloudinary.swatchesPageType = params.swatchesPageType ? params.swatchesPageType.string Value : **null**;
9. }
10. // Custom End: Add cloudinary object //

From line no. 296 to 314

1. // Custom Start: Load cloudinary assets //
2. **var** cloudinary = {};
3. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
5. **if** (cloudinaryConstants.CLD\_ENABLED) {
6. **var** colorAttrValueID = params.get('dwvar\_' + params.pid.stringValue + '\_color').stringValue;
7. **var** cldAssets = cloudinaryModel.getCloudinaryImages(product.object.ID, {
8. pageType: cloudinaryConstants.PAGE\_TYPES.PDP,
9. variationAttrValueID: colorAttrValueID
10. });
12. **if** (cldAssets && cldAssets.galleryWidget && cldAssets.galleryWidget.options &&
13. cldAssets.galleryWidget.options.mediaAssets) {
14. cloudinary.galleryWidgetOptions = cldAssets.galleryWidget.options;
15. }
17. cloudinary.isGalleryEnabled = cloudinaryConstants.CLD\_GALLERY\_ENABLED;
18. }
19. // Custom End: Load cloudinary assets //

From line no. 431 to 450

1. // Custom Start: Load cloudinary assets //
2. **var** cloudinary = {};
3. cloudinary.isEnabled = cloudinaryConstants.CLD\_ENABLED;
5. **if** (cloudinaryConstants.CLD\_ENABLED) {
6. **var** colorAttrValueID = params.get('dwvar\_' + params.pid.stringValue + '\_color').stringValue;
7. **var** cldAssets = cloudinaryModel.getCloudinaryImages(product.object.ID, {
8. pageType: cloudinaryConstants.PAGE\_TYPES.PDP,
9. variationAttrValueID: colorAttrValueID
10. });
12. **if** (cldAssets && cldAssets.galleryWidget && cldAssets.galleryWidget.options &&
13. cldAssets.galleryWidget.options.mediaAssets) {
14. cloudinary.mediaAssets =  cldAssets.galleryWidget.options.mediaAssets;
15. cloudinary.container = cldAssets.galleryWidget.options.container + '-' + (product.object.variant ? product.object.masterProduct.ID : product.object.ID);
16. }
18. cloudinary.isGalleryEnabled = cloudinaryConstants.CLD\_GALLERY\_ENABLED;
19. }
20. // Custom End: Load cloudinary assets //

**Note:** In controller files if you find lines of code which has comments like below then please ignore them as you don’t need to change ~ to \* in your custom/core cartridges.

1. // Custom: Changed ~ to \*

**3.8.3 Make changes in Templates**

Open cloudinary\_sg\_changes cartridge and go to the “templates” folder, it has all overridden templates to integrate Cloudinary on different pages.

* 1. Open file templates/default/checkout/cart/cart.isml and copy code mentioned below and place in your core cartridge cart.isml file on the same path.

From line no. 6 to 12

1. **<iscomment>**Custom Start: Check if cloudinary is enabled and require scripts**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isset** name="cloudinaryModel" value="${require('\*/cartridge/scripts/model/cloudinaryModel')}" scope="page" **/>**
4. **<isset** name="cloudinaryConstants" value="${require('\*/cartridge/scripts/util/cloudinaryConstants')}" scope="page" **/>**
5. **<isset** name="cldProductImage" value="${}" scope="page" **/>**
6. **</isif>**
7. **<iscomment>**Custom End: Check if cloudinary is enabled and require scripts**</iscomment>**

From line no. 175 to 207

1. **<iscomment>**Custom start: Fetch cloudinary image **</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isscript>**
4. cldProductImage = cloudinaryModel.getProductPrimaryImage(lineItem.product.ID, pdict.cloudinary.highResImgViewType, {
5. pageType: pdict.cloudinary.pageType
6. });
7. **</isscript>**
8. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
9. **<img** src="${cldProductImage.url}"
10. style="max-width:100%"
11. alt="${lineItem.product.getImage('small',0).alt}"
12. title="${lineItem.product.getImage('small',0).title}"
13. class="cld-responsive"
14. data-src="${cldProductImage.url}"
15. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
17. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
18. srcset="${cldProductImage.srcset}"
19. **</isif>**
20. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
21. sizes="${cldProductImage.sizes}"
22. **</isif>**
23. **</isif>**
24. **/>**
25. **</isif>**
26. **<iselse/>**
27. **<isif** condition="${lineItem.product != null && lineItem.product.getImage('small',0) != null}"**>**
28. **<img** src="${lineItem.product.getImage('small',0).getURL()}"  alt="${lineItem.product.getImage('small',0).alt}" title="${lineItem.product.getImage('small',0).title}" **/>**
29. **<iselse/>**
30. **<img** src="${URLUtils.staticURL('/images/noimagesmall.png')}" alt="${lineItem.productName}" title="${lineItem.productName}" **/>**
31. **</isif>**
32. **</isif>**
33. **<iscomment>**Custom end: Fetch cloudinary image **</iscomment>**

From line no. 385 to 412

1. **<iscomment>**Custom Start: Add cloudinary image**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isscript>**
4. cldProductImage = cloudinaryModel.getProductPrimaryImage(bli.productID, pdict.cloudinary.highResImgViewType, {
5. pageType: pdict.cloudinary.pageType
6. });
7. **</isscript>**
8. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
9. **<img** src="${cldProductImage.url}"
10. alt="${bli.product.getImage('small',0).alt}"
11. title="${bli.product.getImage('small',0).title}"
12. class="cld-responsive"
13. data-src="${cldProductImage.url}"
14. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
15. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
16. srcset="${cldProductImage.srcset}"
17. **</isif>**
18. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
19. sizes="${cldProductImage.sizes}"
20. **</isif>**
21. **</isif>**
22. **/>**
23. **</isif>**
24. **<iselse/>**
25. **<img** src="${URLUtils.staticURL('/images/noimagesmall.png')}" alt="${bli.productName}" title="${bli.productName}" **/>**
26. **</isif>**
27. **<iscomment>**Custom End: Add cloudinary image**</iscomment>**
28. Open file templates/default/checkout/components/minilineitems.isml and copy code mentioned below and place in your core cartridge minilineitems.isml file on the same path.

From line no. 45 to 50

1. // Custom Start: Check if cloudinary is enabled and require scripts //
2. **if** (pdict.cloudinary && pdict.cloudinary.isEnabled) {
3. **var** cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
4. **var** cldProductImage;
5. }
6. // Custom End: Check if cloudinary is enabled and require scripts //

From line no 85 to 140

1. **<iscomment>**Custom Start: Fetch cloudinary image **</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isscript>**
4. cldProductImage = cloudinaryModel.getProductPrimaryImage(productLineItem.product.ID, pdict.cloudinary.highResImgViewType, {
5. pageType: pdict.cloudinary.pageType
6. });
7. **</isscript>**
8. **<isif** condition="${'isResponsive' in cldProductImage && cldProductImage.isResponsive}"**>**
9. **<style>**
10. .checkout-mini-cart .mini-cart-image {
11. width: 93px;
12. max-width: 93px
13. }
14. .mini-cart-image {
15. width: 130px;
16. max-width: 130px
17. }
18. @media screen and (max-width: 480px) {
19. .mini-cart-image {
20. width: 97px;
21. max-width: 97px
22. }
23. }
24. @media screen and (max-width: 480px) {
25. .checkout-mini-cart .mini-cart-image {
26. width: 115PX;
27. max-width: 115PX
28. }
29. }
30. **</style>**
31. **</isif>**
32. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
33. **<img** src="${cldProductImage.url}"
34. alt="${productLineItem.product.getImage('small',0).alt}"
35. title="${productLineItem.product.getImage('small',0).title}"
36. class="cld-responsive"
37. data-src="${cldProductImage.url}"
38. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
39. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
40. srcset="${cldProductImage.srcset}"
41. **</isif>**
42. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
43. sizes="${cldProductImage.sizes}"
44. **</isif>**
45. **</isif>**
46. **/>**
47. **</isif>**
48. **<iselse/>**
49. **<isif** condition="${productLineItem.product != null && productLineItem.product.getImage('small',0) != null}"**>**
50. **<img** src="${productLineItem.product.getImage('small',0).getURL()}" alt="${productLineItem.product.getImage('small',0).alt}" title="${productLineItem.product.getImage('small',0).title}"**/>**
51. **<iselse/>**
52. **<img** src="${URLUtils.staticURL('/images/noimagesmall.png')}" alt="${productLineItem.productName}"  title="${productLineItem.productName}"**/>**
53. **</isif>**
54. **</isif>**
55. **<iscomment>**Custom End: Fetch cloudinary image **</iscomment>**
56. Open file templates/default/checkout/shipping/multishipping/multishippingaddresses.isml and copy code mentioned below and place in your core cartridge multishippingaddresses.isml file on the same path.

From line no. 36 to 65

1. **<iscomment>**Custom start: Render cloudinary image**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isscript>**
4. var cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
5. var cldProductImage = cloudinaryModel.getProductPrimaryImage(quantityLIproduct.ID, pdict.cloudinary.highResImgViewType, {
6. pageType: pdict.cloudinary.pageType
7. });
8. **</isscript>**
9. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
10. **<img** src="${cldProductImage.url}"
11. alt="${quantityLIproduct.getImage('small',0).alt}"
12. title="${quantityLIproduct.getImage('small',0).title}"
13. style="max-width:50%"
14. class="cld-responsive"
15. data-src="${cldProductImage.url}"
16. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
17. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
18. srcset="${cldProductImage.srcset}"
19. **</isif>**
20. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
21. sizes="${cldProductImage.sizes}"
22. **</isif>**
23. **</isif>**
24. **/>**
25. **</isif>**
26. **<iselse/>**
27. **<img** src="${quantityLIproduct.getImage('small',0).getURL()}" alt="${quantityLIproduct.getImage('small',0).alt}" title="${quantityLIproduct.getImage('small',0).title}"**/>**
28. **</isif>**
29. **<iscomment>**Custom end: Render cloudinary image**</iscomment>**
30. Open file templates/default/checkout/shipping/multishipping/multishippingshipments.isml and copy code mentioned below and place in your core cartridge multishippingshipments.isml file on the same path.

From line no. 116 to 145

1. **<iscomment>**Custom Start: Load cloudinary image**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isscript>**
4. var cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
5. var cldProductImage = cloudinaryModel.getProductPrimaryImage(productLineItem.product.ID, pdict.cloudinary.highResImgViewType, {
6. pageType: pdict.cloudinary.pageType
7. });
8. **</isscript>**
9. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
10. **<img** src="${cldProductImage.url}"
11. alt="${productLineItem.product.getImage('small',0).alt}"
12. title="${productLineItem.product.getImage('small',0).title}"
13. style="max-width:50%"
14. class="cld-responsive"
15. data-src="${cldProductImage.url}"
16. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
17. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
18. srcset="${cldProductImage.srcset}"
19. **</isif>**
20. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
21. sizes="${cldProductImage.sizes}"
22. **</isif>**
23. **</isif>**
24. **/>**
25. **</isif>**
26. **<iselse/>**
27. **<img** src="${productLineItem.product.getImage('small',0).getURL()}" alt="${productLineItem.product.getImage('small',0).alt}" title="${productLineItem.product.getImage('small',0).title}"**/>**
28. **</isif>**
29. **<iscomment>**Custom End: Load cloudinary image**</iscomment>**
30. Open file templates/default/checkout/summary/summary.isml and copy code mentioned below and place in your core cartridge summary.isml file on the same path.

From line no. 55 5o 84

1. **<iscomment>**Custom Start: Render cloudinary image**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isscript>**
4. var cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
5. var cldProductImage = cloudinaryModel.getProductPrimaryImage(productLineItem.product.ID, pdict.cloudinary.highResImgViewType, {
6. pageType: pdict.cloudinary.pageType
7. });
8. **</isscript>**
9. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
10. **<img** src="${cldProductImage.url}"
11. alt="${productLineItem.product.getImage('small',0).alt}"
12. title="${productLineItem.product.getImage('small',0).title}"
13. style="max-width:100%"
14. class="cld-responsive"
15. data-src="${cldProductImage.url}"
16. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
17. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
18. srcset="${cldProductImage.srcset}"
19. **</isif>**
20. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
21. sizes="${cldProductImage.sizes}"
22. **</isif>**
23. **</isif>**
24. **/>**
25. **</isif>**
26. **<iselse/>**
27. **<img** src="${productLineItem.product.getImage('small',0).getURL()}" alt="${productLineItem.product.getImage('small',0).alt}" title="${productLineItem.product.getImage('small',0).title}"**/>**
28. **</isif>**
29. **<iscomment>**Custom End: Render cloudinary image**</iscomment>**
30. Open file templates/default/components/header/htmlhead.isml and copy code mentioned below and place in your core cartridge htmlhead.isml file on the same path.

From line no. 95 to 97

1. **<iscomment>**Custom Start: Cloudinary static libs**</iscomment>**
2. **<isinclude** template="include/cloudinaryStaticLibs"**/>**
3. **<iscomment>**Custom End: Cloudinary static libs**</iscomment>**
4. Open file templates/default/product/components/productsetproduct.isml and copy code mentioned below and place in your core cartridge productsetproduct.isml file on the same path.

From line no. 8 to 19

1. **<iscomment>**Custom Start: Add cloudinary PGW and skip thumbnail**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<div** class="cloudinary-gallery-container"**>**
4. **<isinclude** template="include/cloudinaryProductSetGallery"**/>**
5. **</div>**
6. **<iselse>**
7. **<div** class="product-set-image"**>**
8. **<isset** name="smImg" value="${pdict.Product.getImages('small')[0]}" scope="page" **/>**
9. **<img** class="product-thumbnail" src="${smImg.getURL()}" alt="${smImg.alt}"**/>**
10. **</div>**
11. **</isif>**
12. **<iscomment>**Custom End: Add cloudinary PGW and skip thumbnail**</iscomment>**

From line no. 99 to 105

1. **<iscomment>**Custom Start: Add cloudinary data attribute to be accessed from js**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled && pdict.cloudinary.mediaAssets}"**>**
3. **<div** id="update-images" style="display:none"
4. data-cld="${JSON.stringify(pdict.cloudinary)}"**>**
5. **</div>**
6. **</isif>**
7. **<iscomment>**Custom End: Add cloudinary data attribute to be accessed from js**</iscomment>**
8. Open file templates/default/product/components/recommendations.isml and copy code mentioned below and place in your core cartridge recommendations.isml file on the same path.

From line no. 39 to 68

1. **<iscomment>**Custom Start: Render cloudinary image**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isscript>**
4. var cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
5. var cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants')
6. var cldProductImage = cloudinaryModel.getProductPrimaryImage(pdict.Product.ID, cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE, {
7. pageType: cloudinaryConstants.PAGE\_TYPES.RECOMMENDATION\_TILE
8. });
9. **</isscript>**
10. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
11. **<img** src="${cldProductImage.url}"
12. style="max-width:50%"
13. alt="${pdict.Product.getImage('small',0).alt}"
14. class="cld-responsive"
15. data-src="${cldProductImage.url}"
16. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
17. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
18. srcset="${cldProductImage.srcset}"
19. **</isif>**
20. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
21. sizes="${cldProductImage.sizes}"
22. **</isif>**
23. **</isif>**
24. **/>**
25. **</isif>**
26. **<iselse/>**
27. **<img** src="${pdict.Product.getImage('small',0).getURL()}" alt="${pdict.Product.getImage('small',0).alt}"**/>**
28. **</isif>**
29. **<iscomment>**Custom End: Render cloudinary image**</iscomment>**
30. Open file templates/default/product/components/variations.isml and copy code mentioned below and place in your core cartridge variations.isml file on the same path.

From line no. 48 to 67

1. **<iscomment>**Custom start: Add cloudinary swatch images**</iscomment>**
2. **<isif** condition="${'cldSwatchImg' in attrValue && !empty(attrValue.cldSwatchImg)}"**>**
3. **<img** src="${attrValue.cldSwatchImg.url}"
4. alt="${attrValue.displayValue}"
5. class="cld-responsive"
6. data-src="${attrValue.cldSwatchImg.url}"
7. **<isif** condition="${!('isResponsive' in attrValue.cldSwatchImg && attrValue.cldSwatchImg.isResponsive)}"**>**
8. **<isif** condition="${'srcset' in attrValue.cldSwatchImg && !empty(attrValue.cldSwatchImg.srcset)}"**>**
9. srcset="${attrValue.cldSwatchImg.srcset}"
10. **</isif>**
11. **<isif** condition="${'sizes' in attrValue.cldSwatchImg && !empty(attrValue.cldSwatchImg.sizes)}"**>**
12. sizes="${attrValue.cldSwatchImg.sizes}"
13. **</isif>**
14. **</isif>**
15. **>**
16. **<iselse/>**
17. **<img** src="${attrValue.swatchImageUrl}" alt="${attrValue.displayValue}"**>**
18. **</isif>**
19. **<iscomment>**Custom end: Add cloudinary swatch images**</iscomment>**
20. Open file templates/default/product/lastvisited.isml and copy code mentioned below and place in your core cartridge lastvisited.isml file on the same path.

From line no. 11 to 13

1. **<iscomment>**Custom Start: Add cloudinary page type as query param**</iscomment>**
2. **<isinclude** url="${URLUtils.url('Product-HitTile','pid',product.ID,'showswatches','false','showpricing','false','showpromotion','true','showrating','false', 'pageType', 'cldCartProductTile')}"**/>**
3. **<iscomment>**Custom End: Add cloudinary page type as query param**</iscomment>**
4. Open file templates/default/product/productcontent.isml and copy code mentioned below and place in your core cartridge productcontent.isml file on the same path.

From line no. 278 to 285

1. **<iscomment>**Custom Start: Add cloudinary data attribute to be accessed from js**</iscomment>**
2. **<div id="update-images" style="display:none" data-cld="${JSON.stringify(pdict.cloudinary)}">**
3. **<isinclude** template="product/components/productimages"**/>**
4. **</div>**
5. **<iscomment>**Custom End: Add cloudinary data attribute to be accessed from js**</iscomment>**
6. Open file templates/default/product/producttile.isml and copy code mentioned below and place in your core cartridge producttile.isml file on the same path.

From line no. 29 to 33

1. // Custom Start: Load cloudinary tile wrapper //
2. **if** (pdict.cloudinary && pdict.cloudinary.isEnabled) {
3. **var** cldProductTileWrapper = require('\*/cartridge/scripts/wrappers/cldProductTileWrapper');
4. }
5. // Custom End: Load cloudinary tile wrapper //

From line no. 53 to 61

1. // Custom start: Cloudinary product Tile Image
2. **if** (pdict.cloudinary && pdict.cloudinary.isEnabled && pdict.cloudinary.isSwachesEnabled) {
3. selectableColors = cldProductTileWrapper.getCloudinaryColorVariationAttrValues(Product.ID, {
4. pageType: pdict.cloudinary.swatchesPageType
5. });
6. } **else** {
7. selectableColors = PVM.getFilteredValues(colorVarAttr);
8. }
9. // Custom end: Cloudinary product Tile Image

From line no. 101 to 109

1. // Custom start: Cloudinary product Tile Image
2. **if** (pdict.cloudinary && pdict.cloudinary.isEnabled) {
3. image = cldProductTileWrapper.getCloudinaryProductTileImage(imageSource.ID, {
4. pageType: pdict.cloudinary.pageType
5. });
6. } **else** {
7. image = imageSource.getImage(imageSize, 0);
8. }
9. // Custom end: Cloudinary product Tile Image

From line no. 146 to159

1. **<iscomment>**Custom start: Cloudinary product Tile Image attributes**</iscomment>**
2. **<img** src="${thumbnailUrl}" alt="${thumbnailAlt}" class="cld-responsive"
3. data-src="${image.getURL()}"
4. **<isif** condition="${!('isResponsive' in image && image.isResponsive)}"**>**
5. **<isif** condition="${'srcset' in image && !empty(image.srcset)}"**>**
6. srcset="${image.srcset}"
7. **</isif>**
8. **<isif** condition="${'sizes' in image && !empty(image.sizes)}"**>**
9. sizes="${image.sizes}"
10. **</isif>**
11. **</isif>**
12. **/>**
13. **<iscomment>**Custom end: Cloudinary product Tile Image attributes**</iscomment>**

From line no. 329 to 342

1. **<iscomment>**Custom Start: Add Cloudinary product swatch/thumbmail image attributes**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled && 'cldSwatchImage' in colorValue &&
3. !empty(colorValue.cldSwatchImage)}"**>**
4. **<isset** name="colorSwatch" value="${colorValue.cldSwatchImage}" scope="page"**/>**
5. **<iselse/>**
6. **<isset** name="colorSwatch" value="${colorValue.getImage('swatch')}" scope="page"**/>**
7. **</isif>**
8. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled && 'cldThumbnailImage' in colorValue &&
9. !empty(colorValue.cldThumbnailImage)}"**>**
10. **<isset** name="colorThumbnail" value="${colorValue.cldThumbnailImage}" scope="page"**/>**
11. **<iselse/>**
12. **<isset** name="colorThumbnail" value="${colorValue.getImage('medium')}" scope="page"**/>**
13. **</isif>**
14. **<iscomment>**Custom End: Add Cloudinary product swatch/thumbmail image**</iscomment**

From line no. 378 to 393

1. **<iscomment>**Custom Start: Add cloudinary swatch image and attributes**</iscomment>**
2. **<img** class="swatch-image cld-responsive"
3. src="${swatchUrl}" alt="${swatchAlt}"
4. data-thumb='{"src":"${thumbnailUrl}","alt":"${thumbnailAlt}","title":"${thumbnailTitle}"}'
5. data-src="${swatchUrl}"
6. **<isif** condition="${!('isResponsive' in colorSwatch && colorSwatch.isResponsive)}"**>**
7. **<isif** condition="${'srcset' in colorSwatch && !empty(colorSwatch.srcset)}"**>**
8. srcset="${colorSwatch.srcset}"
9. **</isif>**
10. **<isif** condition="${'sizes' in colorSwatch && !empty(colorSwatch.sizes)}"**>**
11. sizes="${colorSwatch.sizes}"
12. **</isif>**
13. **</isif>**
14. **/>**
15. **<iscomment>**Custom Start: Add cloudinary swatch image and attributes**</iscomment>**
16. Open file templates/default/product/producttopcontent.isml and copy code mentioned below and place in your core cartridge producttopcontent.isml file on the same path.

From line no. 14 to 24

1. **<iscomment>**Custom Start: Add cloudinary PGW**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isQuickView}"**>**
4. **<div** class="cloudinary-data-container visually-hidden" data-cloudinary="<isprint value="${JSON.stringify(pdict.cloudinary)}" encoding="htmlunquote" **/>**"**></div>**
5. **</isif>**
6. **<isinclude** template="product/components/cloudinaryGallery" **/>**
7. **<isset** name="isBundleOrSet" value="${false}" scope="page" **/>**
8. **<iselse>**
9. **<isinclude** template="product/components/productimages"**/>**
10. **</isif>**
11. **<iscomment>**Custom End: Add cloudinary PGW**</iscomment>**

From line no. 25 to 32

1. **<iscomment>**Custom Start: Cloudinary video player**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled && pdict.cloudinary.videoEnabled}"**>**
3. **<isset** name="cldProductID" value="${pdict.Product.ID}" scope="page" **/>**
4. **<div** style="margin-top:35px"**>**
5. **<isinclude** template="experience/pages/cloudinaryVideo" **/>**
6. **</div>**
7. **</isif>**
8. **<iscomment>**Custom End: Cloudinary video player**</iscomment>**
9. Open file templates/default/product/producttopcontentPS.isml and copy code mentioned below and place in your core cartridge producttopcontentPS.isml file on the same path.

From line no. 11 to 15

1. **<iscomment>**Custom Start: Add cloudinary product set/bundle styles**</iscomment>**
2. **<isif** condition="${pdict.cloudinary && pdict.cloudinary.isEnabled}"**>**
3. **<isinclude** template="include/cloudinaryProductSetBundleStyles"**/>**
4. **</isif>**
5. **<iscomment>**Custom End: Add cloudinary product set/bundle styles**</iscomment>**

From line no. 54 to 114

1. **<iscomment>**Custom Start: Skip SG product Set/bundle primary images carousel**</iscomment>**
2. **<isif** condition="${!pdict.cloudinary || !pdict.cloudinary.isEnabled}"**>**
3. **<isset** name="FirstProductInSet" value="${pdict.ProductSetList[0]}" scope="pdict" **/>**
4. **<isset** name="imageProduct" value="${pdict.FirstProductInSet}" scope="page"**/>**
6. **<isif** condition="${!empty(imageProduct.getImage('large', 0))}"**>**
7. **<isset** name="imageUrl" value="${imageProduct.getImage('large', 0).getURL()}" scope="page"**/>**
8. **<isset** name="imageAlt" value="${imageProduct.getImage('large', 0).alt}" scope="page"**/>**
9. **<isset** name="imageTitle" value="${imageProduct.getImage('large', 0).title}" scope="page"**/>**
10. **<iselse/>**
11. **<isset** name="imageUrl" value="${URLUtils.staticURL('/images/noimagelarge.png')}" scope="page"**/>**
12. **<isset** name="imageAlt" value="${pdict.Product.name}" scope="page"**/>**
13. **<isset** name="imageTitle" value="${pdict.Product.name}" scope="page"**/>**
14. **</isif>**
16. **<isif** condition="${!empty(imageProduct.getImage('hi-res', 0))}"**>**
17. **<isset** name="imageUrlHiRes" value="${imageProduct.getImage('hi-res', 0).getURL()}" scope="page"**/>**
18. **<iselse/>**
19. **<isset** name="imageUrlHiRes" value="${URLUtils.staticURL('/images/noimagelarge.png')}" scope="page"**/>**
20. **</isif>**
22. **<div** class="product-primary-image"**>**
23. **<h2** class="visually-hidden"**>**Images**</h2>**
25. **<isif** condition="${!empty(imageProduct.getImages('hi-res')) && imageProduct.getImages('hi-res').length>0}"**>**
26. **<isset** name="hi\_res" scope="page" value="${imageProduct.getImages('hi-res')[0].getURL().toString()}" **/>**
27. **<isset** name="href" scope="page" value="${'href='+hi\_res+' target=\_blank'}" **/>**
28. **<iselse/>**
29. **<isset** name="hi\_res" scope="page" value="${''}" **/>**
30. **<isset** name="href" scope="page" value="${'name=product\_detail\_image'}" **/>**
31. **</isif>**
32. **<a** ${href} class="product-image main-image" title="${pdict.Product.name}${imageTitle}"**>**
33. **<img** itemprop="image" class="primary-image" src="${imageUrl}" alt="${imageAlt}"**/>**
34. **</a>**
35. **</div>**
37. **<div** id="thumbnails" class="product-thumbnails"**>**
38. **<ul>**
39. **<isloop** items="${pdict.ProductSetList}" var="product" status="loopstatus"**>**
40. **<isset** name="lgImage" value="${product.getImages('large')[0]}" scope="page" **/>**
41. **<isif** condition="${!empty(imageProduct.getImage('hi-res', 0))}"**>**
42. **<isset** name="hi\_res" value="${product.getImages('hi-res')[0]}" scope="page" **/>**
43. **</isif>**
44. **<isset** name="href" value="${''}" scope="page" **/>**
45. **<isif** condition="${!empty(hi\_res)}"**>**
46. **<isset** name="href" value="${hi\_res.getURL()}" scope="page" **/>**
47. **</isif>**
48. **<isset** name="smImage" value="${product.getImages('small')[0]}" scope="page" **/>**
49. **<li** class="thumb ${loopstatus.first ? 'selected' : ''}"**>**
50. **<a** href="${href}" target="\_blank" class="thumbnail-link"**>**
51. **<img** class="productthumbnail"
52. src="${smImage.getURL()}"
53. alt="${smImage.alt}"
54. data-lgimg='{"url":"${lgImage.getURL()}", "title":"${lgImage.title}", "alt":"${lgImage.alt}", "hires":"${href}"}'**/>**
55. **</a>**
56. **</li>**
57. **</isloop>**
58. **</ul>**
59. **</div>**
60. **</isif>**
61. **<iscomment>**Custom End: Skip SG product Set/bundle primary images carousel**</iscomment>**

From line no. 199 to 206

1. **1<iscomment>**Custom Start: Cloudinary video player**</iscomment>**
2. **<isif** condition="${!isQuickView && pdict.cloudinary && pdict.cloudinary.isEnabled && pdict.cloudinary.videoEnabled}"**>**
3. **<isset** name="cldProductID" value="${pdict.Product.ID}" scope="page" **/>**
4. **<div** style="margin-top:35px; max-width: 50%"**>**
5. **<isinclude** template="experience/pages/cloudinaryVideo" **/>**
6. **</div>**
7. **</isif>**
8. **<iscomment>**Custom End: Cloudinary video player**</iscomment>**
9. Open file templates/default/search/productgrid.isml and copy code mentioned below and place in your core cartridge productgrid.isml file on the same path.

From line no. 12 to 16

1. **<iscomment>**Custom Start: Add cloudinary query params**</iscomment>**
2. **<isset** name="tileShowURL" value="${ProductGridUtils.buildHitTileURL(LoopProductHit, pdict.CurrentHttpParameterMap.cgid.value)}" scope="page" **/>**
3. **<iscomment>**<!-- Self contained product information. Can be cached separately even if search finds this product in different categories or positions within the grid  -->**</iscomment>**
4. **<isinclude** url="${tileShowURL + '&pageType=plp&swatchesPageType=cldPlpSwatch'}"**/>**
5. **<iscomment>**Custom End: Add cloudinary query params**</iscomment>**
6. Open file templates/default/search/productnav.isml and copy code mentioned below and place in your core cartridge productnav.isml file on the same path.

From line no. 70 to 90

1. // Custom Start: Add Cloudianry primary image //
2. **if** (dw.system.Site.current.preferences.getCustom().CLDEnabled) {
3. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
4. **var** cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
5. **var** cldImg = cloudinaryModel.getProductPrimaryImage(product.ID, cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE, {
6. pageType: cloudinaryConstants.PAGE\_TYPES.CLD\_PRODUCT\_NAV
7. });
9. **if** (cldImg) {
10. **return** {
11. pUrl: productUrl + '#' + hashAppend,
12. imgUrl: (cldImg.url != **null**) ? cldImg.url : URLUtils.staticURL('/images/noimagesmall.png'),
13. srcset: (cldImg.srcset != **null**) ? cldImg.srcset : **null**,
14. sizes: (cldImg.sizes != **null**) ? cldImg.sizes : **null**,
15. isResponsive: (cldImg.isResponsive != **null**) ? cldImg.isResponsive : **null**,
16. imgAlt: (imgFile != **null**) ? imgFile.alt : product.name,
17. imgTitle: (imgFile != **null**) ? imgFile.title : product.name
18. };
19. }
20. }
21. // Custom End: Add Cloudianry primary image //

From line no. 136 to 152

1. **<iscomment>**Custom Start: Add cloudinary image and attributes**</iscomment>**
2. **<img** alt="${prevProdUrls.imgAlt}"
3. title="${prevProdUrls.imgTitle}"
4. src="${prevProdUrls.imgUrl}"
5. class="cld-responsive"
6. data-src="${prevProdUrls.imgUrl}"
7. **<isif** condition="${!('isResponsive' in prevProdUrls && prevProdUrls.isResponsive)}"**>**
8. **<isif** condition="${'srcset' in prevProdUrls && !empty(prevProdUrls.srcset)}"**>**
9. srcset="${prevProdUrls.srcset}"
10. **</isif>**
11. **<isif** condition="${'sizes' in prevProdUrls && !empty(prevProdUrls.sizes)}"**>**
12. sizes="${prevProdUrls.sizes}"
13. **</isif>**
14. **</isif>**
15. **/>**
16. **<iscomment>**Custom End: Add cloudinary image and attributes**</iscomment>**

From line no. 163 to 179

1. **<iscomment>**Custom Start: Add cloudinary image and attributes**</iscomment>**
2. **<img** alt="${nextProdUrls.imgAlt}"
3. title="${nextProdUrls.imgTitle}"
4. src="${nextProdUrls.imgUrl}"
5. class="cld-responsive"
6. data-src="${nextProdUrls.imgUrl}"
7. **<isif** condition="${!('isResponsive' in nextProdUrls && nextProdUrls.isResponsive)}"**>**
8. **<isif** condition="${'srcset' in nextProdUrls && !empty(nextProdUrls.srcset)}"**>**
9. srcset="${nextProdUrls.srcset}"
10. **</isif>**
11. **<isif** condition="${'sizes' in nextProdUrls && !empty(nextProdUrls.sizes)}"**>**
12. sizes="${nextProdUrls.sizes}"
13. **</isif>**
14. **</isif>**
15. **/>**
16. **<iscomment>**Custom End: Add cloudinary image and attributes**</iscomment>**
17. Open file templates/default/search/suggestions.isml and copy code mentioned below and place in your core cartridge suggestions.isml file on the same path.

From line no. 26 to 33

1. // Custom Start: Check if cloudinary is enabled and require scripts //
2. **var** isCloudinaryEnabled = dw.system.Site.current.preferences.getCustom().CLDEnabled;
3. **if** (isCloudinaryEnabled) {
4. **var** cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
5. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
6. **var** cldProductImage;
7. }
8. // Custom End: Check if cloudinary is enabled and require scripts //

From line no. 84 to 92

1. **<iscomment>**Custom Start: Fetch cloudinary image **</iscomment>**
2. **<isif** condition="${isCloudinaryEnabled}"**>**
3. **<isscript>**
4. cldProductImage = cloudinaryModel.getProductPrimaryImage(product.ID, cloudinaryConstants.CLD\_HIGH\_RES\_IMAGES\_VIEW\_TYPE, {
5. pageType: cloudinaryConstants.PAGE\_TYPES.SEARCH\_SUGGESTIONS
6. });
7. **</isscript>**
8. **</isif>**
9. **<iscomment>**Custom End: Fetch cloudinary image **</iscomment>**

From line no. 96 to 116

1. **<iscomment>**Custom Start: Add cloudinary image **</iscomment>**
2. **<isif** condition="${isCloudinaryEnabled && cldProductImage && 'url' in cldProductImage}"**>**
3. **<img** src="${cldProductImage.url}"
4. alt="${productImage.alt}"
5. title="${productImage.title}"
6. class="cld-responsive"
7. data-src="${cldProductImage.url}"
8. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
9. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
10. srcset="${cldProductImage.srcset}"
11. **</isif>**
12. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
13. sizes="${cldProductImage.sizes}"
14. **</isif>**
15. **</isif>**
16. **/>**
17. **<iselse/>**
18. **<img** src="${productImage.getURL()}" alt="${productImage.alt}" title="${productImage.title}" **/>**
19. **</isif>**
20. **<iscomment>**Custom End: Add cloudinary image **</iscomment>**
21. Open file templates/default/slots/category/cat-banner.isml and copy code mentioned below and place in your core cartridge cat-banner.isml file on the same path.

From line no. 10 to 34

1. **<iscomment>**Custom start: Add cloudinary product image**</iscomment>**
2. **<isif** condition="${dw.system.Site.current.preferences.getCustom().CLDEnabled}"**>**
3. **<isset** name="cloudinaryAPI" value="${require('\*/cartridge/scripts/api/cloudinaryApi')}" scope="page" **/>**
4. **<isset** name="cloudinaryConstants" value="${require('\*/cartridge/scripts/util/cloudinaryConstants')}" scope="page" **/>**
5. **<isset** name="cldProductImage" value="${cloudinaryAPI.getCatalogImageAbsURLFromRelURL(category.image.URL.toString(), category.ID, cloudinaryConstants.PAGE\_TYPES.CATEGORY\_BANNER);}" scope="page" **/>**
6. **<isif** condition="${cldProductImage && 'url' in cldProductImage}"**>**
7. **<img** src="${cldProductImage.url}"
8. alt="${category.displayName}"
9. class="cld-responsive"
10. data-src="${cldProductImage.url}"
11. **<isif** condition="${!('isResponsive' in cldProductImage && cldProductImage.isResponsive)}"**>**
12. **<isif** condition="${'srcset' in cldProductImage && cldProductImage.srcset}"**>**
13. srcset="${cldProductImage.srcset}"
14. **</isif>**
15. **<isif** condition="${'sizes' in cldProductImage && cldProductImage.sizes}"**>**
16. sizes="${cldProductImage.sizes}"
17. **</isif>**
18. **</isif>**
19. **/>**
20. **</isif>**
21. **<iselse/>**
22. **<img** src="${category.image.httpURL}" alt="${category.displayName}" **/>**
23. **</isif>**
24. **<iscomment>**Custom end: Add cloudinary product image**</iscomment>**
25. Open file templates/default/slots/product/homerightcarousel.isml and copy code mentioned below and place in your core cartridge homerightcarousel.isml file on the same path.

From line no. 12 to 29

1. **<iscomment>**Custom Start: Build cloudinary object to pass in producttile template**</iscomment>**
2. **<isset** name="isCloudinaryEnabled" value="${dw.system.Site.current.preferences.getCustom().CLDEnabled}" scope="page" **/>**
3. **<isif** condition="${isCloudinaryEnabled}"**>**
4. **<style>**
5. .home-bottom-right .product-tile img {
6. max-width: 75%;
7. }
8. **</style>**
9. **<isscript>**
10. var cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
11. var cloudinary = {
12. isEnabled: isCloudinaryEnabled,
13. isSwachesEnabled: false,
14. pageType: cloudinaryConstants.PAGE\_TYPES.CLD\_HOME\_PAGE\_TILE
15. }
16. **</isscript>**
17. **</isif>**
18. **<iscomment>**Custom End: Build cloudinary object to pass in producttile template**</iscomment>**
19. Open file templates/default/slots/product/product\_listing.isml and copy code mentioned below and place in your core cartridge product\_listing.isml file on the same path.

From line no. 5 to 17

1. **<iscomment>**Custom Start: Build cloudinary object to pass in producttile template**</iscomment>**
2. **<isset** name="isCloudinaryEnabled" value="${dw.system.Site.current.preferences.getCustom().CLDEnabled}" scope="page" **/>**
3. **<isif** condition="${isCloudinaryEnabled}"**>**
4. **<isscript>**
5. var cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
6. var cloudinary = {
7. isEnabled: isCloudinaryEnabled,
8. isSwachesEnabled: false,
9. pageType: cloudinaryConstants.PAGE\_TYPES.CLD\_CART\_PRODUCT\_TILE
10. }
11. **</isscript>**
12. **</isif>**
13. **<iscomment>**Custom End: Build cloudinary object to pass in producttile template**</iscomment>**

From line no. 24 to 31

1. **<iscomment>**Custom Start: Pass cloudinary object to product tile**</iscomment>**
2. **<isproducttile** product="${product}"
3. showswatches="${true}"
4. showpricing="${true}"
5. showpromotion="${true}"
6. showrating="${true}"
7. cloudinary="${isCloudinaryEnabled ? cloudinary : null}"**/>**
8. **<iscomment>**Custom End: Pass cloudinary object to product tile**</iscomment>**

**3.8.4 Make changes in server side script files**

There are only two script files in cloudinary\_sg\_changes cartridge which are required to change as described below but you don’t need to make changes in other overridden script files as they were only moved to refer “.ds” files in overridden templates.

* + - 1. Open file scripts/product/VariationUtils.js and copy code mentioned below and place in your core cartridge VariationUtils.js file on the same path.

From line no. 7 to 10

1. // Custom Start: Cloudinary script modules //
2. **var** cloudinaryConstants = require('\*/cartridge/scripts/util/cloudinaryConstants');
3. **var** cloudinaryModel = require('\*/cartridge/scripts/model/cloudinaryModel');
4. // Custom End: Cloudinary script modules //

From line no. 268 to 281

1. // Custom Start: Add cloudinary swatch image //
2. **var** linkUrl;
3. **if** (cloudinaryConstants.CLD\_ENABLED) {
4. linkUrl = variationMaster.urlSelectVariationValue('Product-Variation', attr, attrValue);
5. processedValue.cldSwatchImg = cloudinaryModel.getCloudinaryProductSwatchImage(variationMaster.master.ID, {
6. pageType: cloudinaryConstants.PAGE\_TYPES.CLD\_PDP\_SWATCH,
7. variationAttrValueID: attrValue.ID
8. });
9. } **else** {
10. linkUrl = isSelected
11. ? variationMaster.urlUnselectVariationValue('Product-Variation', attr)
12. : variationMaster.urlSelectVariationValue('Product-Variation', attr, attrValue);
13. }
14. // Custom End: Add cloudinary swatch image //
    * + 1. Open file scripts/search/ProductGridUtils.ds and copy code mentioned below and place in your core cartridge ProductGridUtils.ds file on the same path.

From line no. 66 to 68

1. // Custom Start: Export module //
2. module.exports = ProductGridUtils;
3. // Custom End: Export module //

**Note:** In script files if you find lines of code which has comments like below then please ignore them as you don’t need to change ~ to \* in your custom/core cartridges.

1. // Custom: Changed ~ to \*

**3.8.5 Load Library Content Image from Cloudinary on Content Pages**

Open any content asset where you want to load images from Cloudinary, for example open the “about-us” content asset and place the following code to load a top hero banner background image from Cloudinary.

$Include('Cloudinary-GetContentImage', 'url', 'images/AboutUsBanner/aboutus.jpg', 'pageType', 'aboutUs', 'isUrl', true)$

The query params are as follows:

**url**: The content image relative URL

**pageType**: Used to load responsive images include srcset and sizes attributes

**isURL**: Indicates either load only transformed URL or image tag having all srcset and sizes attributes.

**3.8.6 Include Cloudinary Video Player on Content Pages**

Open any content asset where you want to load images from Cloudinary, for example open the “about-us” content asset and place the following code to load the Cloudinary Video Player.

$Include('Cloudinary-GetContentVideo', 'url', 'videos/cat.mp4')$

Here, url is the video relative URL present in your static content library.

**3.8.7 Cloudinary Script APIs**

The cartridge includes low level script APIs that are used by the cartridge itself to fetch assets from Cloudinary and apply transformations on the URLs. Integrators may use them to customize the cartridge according to their needs, or make use of them in their own custom cartridges. Each of the methods has documentation in the code that explains the method’s usage.

Here are the details about the script APIs:

1. **cloudinaryApi:** Used to communicate directly with Cloudinary APIs to fetch different types of assets and apply transformations on the URLs.
2. **cloudinaryTranformationApi:** Used to provide transformations specified on several levels for different types of assets.

**3.8.8 Site version for Cloudinary Test Suite**

There is a cartridge int\_cloudinary\_testsuite which is used to test script APIs that directly communicate with Cloudinary web services.

To make it work with any Commerce Cloud version either it’s SiteGenesis or SFRA, you need to update a constant present inside cloudinaryConstants.js file accordingly. The constant name and possible values are as follows.

|  |  |  |
| --- | --- | --- |
| **Commerce Cloud Site Version** | **Constant Name** | **Value** |
| SiteGenesis | SITE\_VERSION\_FOR\_TEST\_SUITE | sitegenesis |
| SFRA | SITE\_VERSION\_FOR\_TEST\_SUITE | sfra |

**Note:** By default, it’s set to “sfra”.

## 3.9 Managing Videos

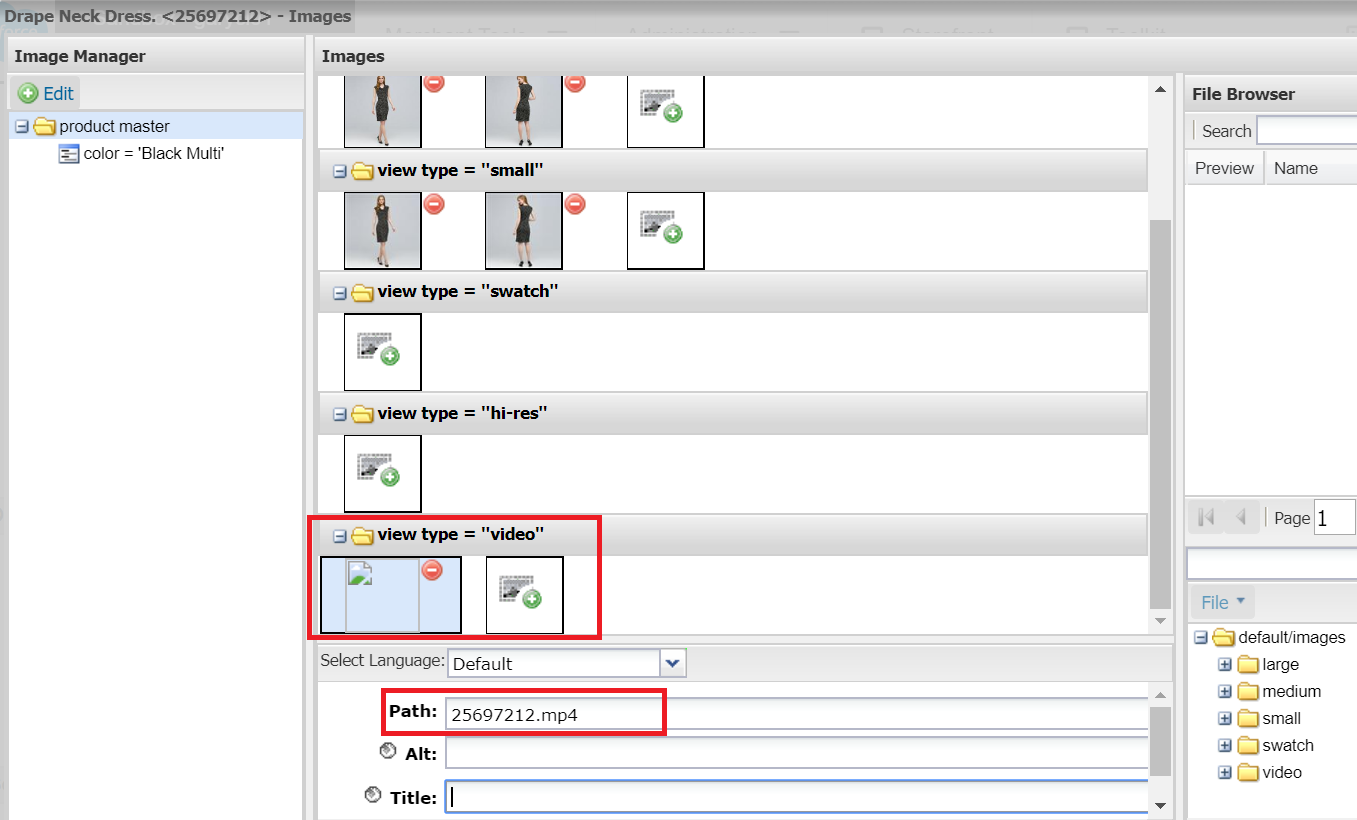
There are two ways to make videos available for products,

**3.9.1 By Using SFCC view type:**

First, create a view type at the catalog level by navigating through the following path **Merchant Tools > Products and Catalogs > Catalogs > (Your Catalog) > Image Settings.** Against **View Types** you can see there are already standard view types provided by SFCC. Click on the **Add View Type** link to add a new view type with name, “video”.



After adding the view type, go to any product and click on the edit button to edit product images. Notice the new view type in the pop-up window with the name, “video”. Click on the icon to assign video to this view type for this product and in the field “path”, add the public ID of a video that is already present in your Cloudinary DAM.



**Note**: At present, only one video can be assigned to the view type, so if you upload multiple videos then it would pick the first one.

* + 1. **By using custom mapping**

Custom mapping assumes the video is already present in your Cloudinary DAM. Define the custom mapping scheme using the site level preference, “Cloudinary Video Custom Map Path”.

Note: Make sure you have enabled custom mapping for videos by using site preference, “Cloudinary Use Video Custom Mapping”.

# 4. Operations, Maintenance

## 4.1 Data Storage

*Intentionally left blank.*

## 4.2 Availability

Cloudinary services are based on a highly available infrastructure designed for 100% uptime. The status of Cloudinary’s services can be found on our service status page -<https://status.cloudinary.com/>.  From here, you can subscribe to service status updates.  In the unlikely event that Cloudinary services go down, storefronts will continue to display any assets previously requested from Cloudinary since they are cached on the edge using CDNs.  However, assets that are being requested by end users for the very first time will not be displayed by storefronts.  Please contact the Cloudinary support team to address such incidents.

## 4.3 Support

Contact our support team if you have any issues with the cartridge.

<https://support.cloudinary.com/hc/en-us>

## 4.4 Cloudinary Workflow

Workflow Steps:

1. Set up images on your staging environment through the Sync Jobs. Make sure job executions are complete before doing any production replication.
2. Test the image functionality on the staging storefront. There may be a time gap where missing images appear on your staging environment while the job is executing.
3. If images appear correctly on staging, replicate to production.

# 5. User Guide

## 5.1 Roles, Responsibilities

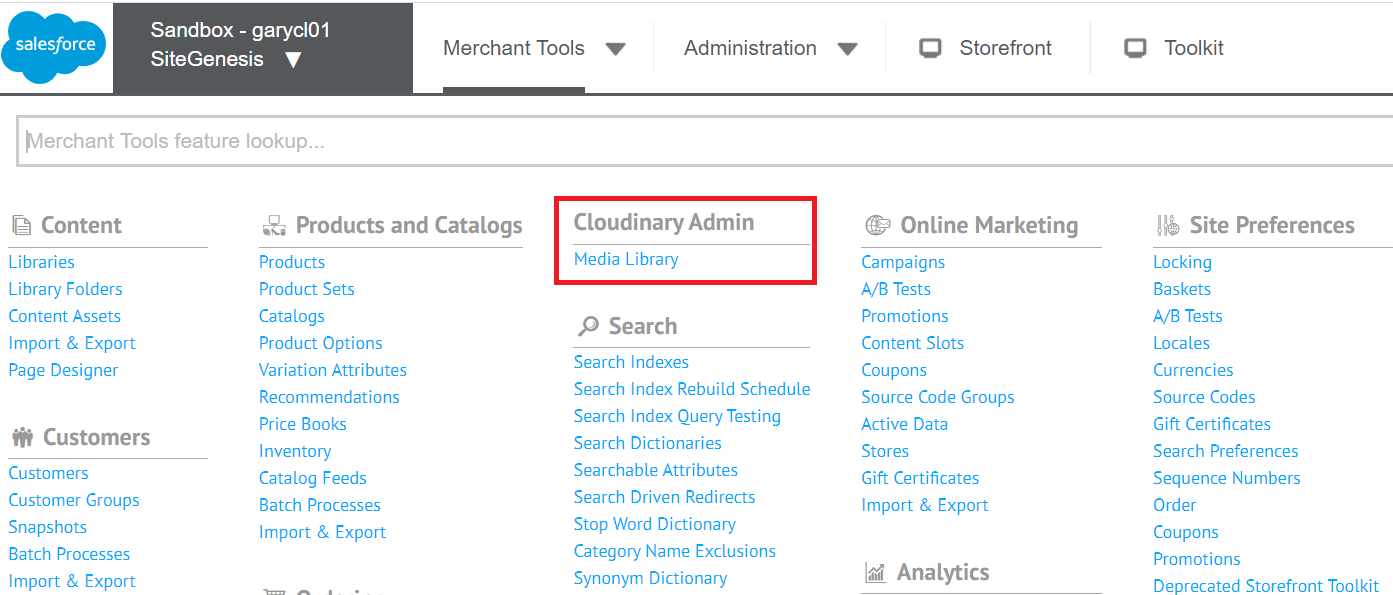
*Intentionally left blank.*

## 5.2 Business Manager

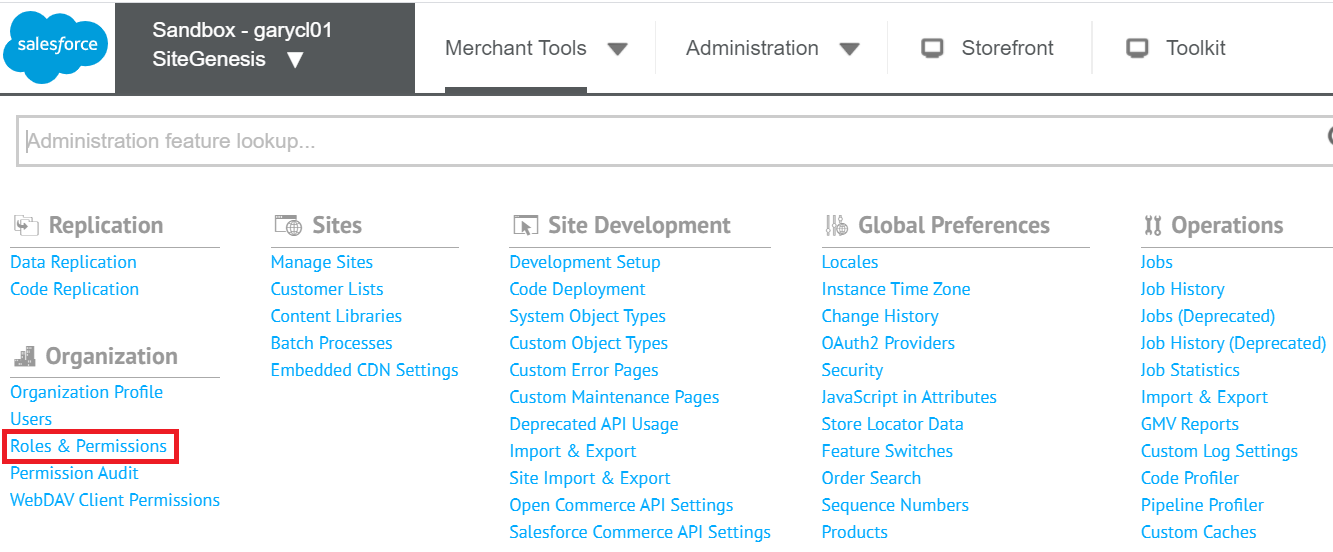
**5.2.1 Access Cloudinary Media Library through Business Manager**

To open the Cloudinary Media Library within your Business Manager:

* Open Business Manager and select your site.
* Click Merchant Tools -> Cloudinary Admin -> Media Library. The Media Library opens.



* If you don’t find this module then try granting BM module permission to the logged in user.
* To assign BM module permissions click Administration > Organization > Roles & Permissions.



* Click “Administration” and go into the “Business Manager Modules” tab.
* Select your context and select both checkboxes for the “Cloudinary Admin” and “Media Library” modules, then save.



* If still you aren't able to see the Media Library module then try logging in again.

Note: For the Media Library, the “bm\_cloudinary” cartridge must be in your Business Manager site effective path.

## 5.3 Storefront Functionality

*Intentionally left blank.*

# 6 Known Issues

*Intentionally left blank.*

# 7 Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0.0 | 03/31/2020 | Initial release |
|  |  |  |