Cloudinary

SAP Commerce Cloudinary Extension Integration

Technical Guide

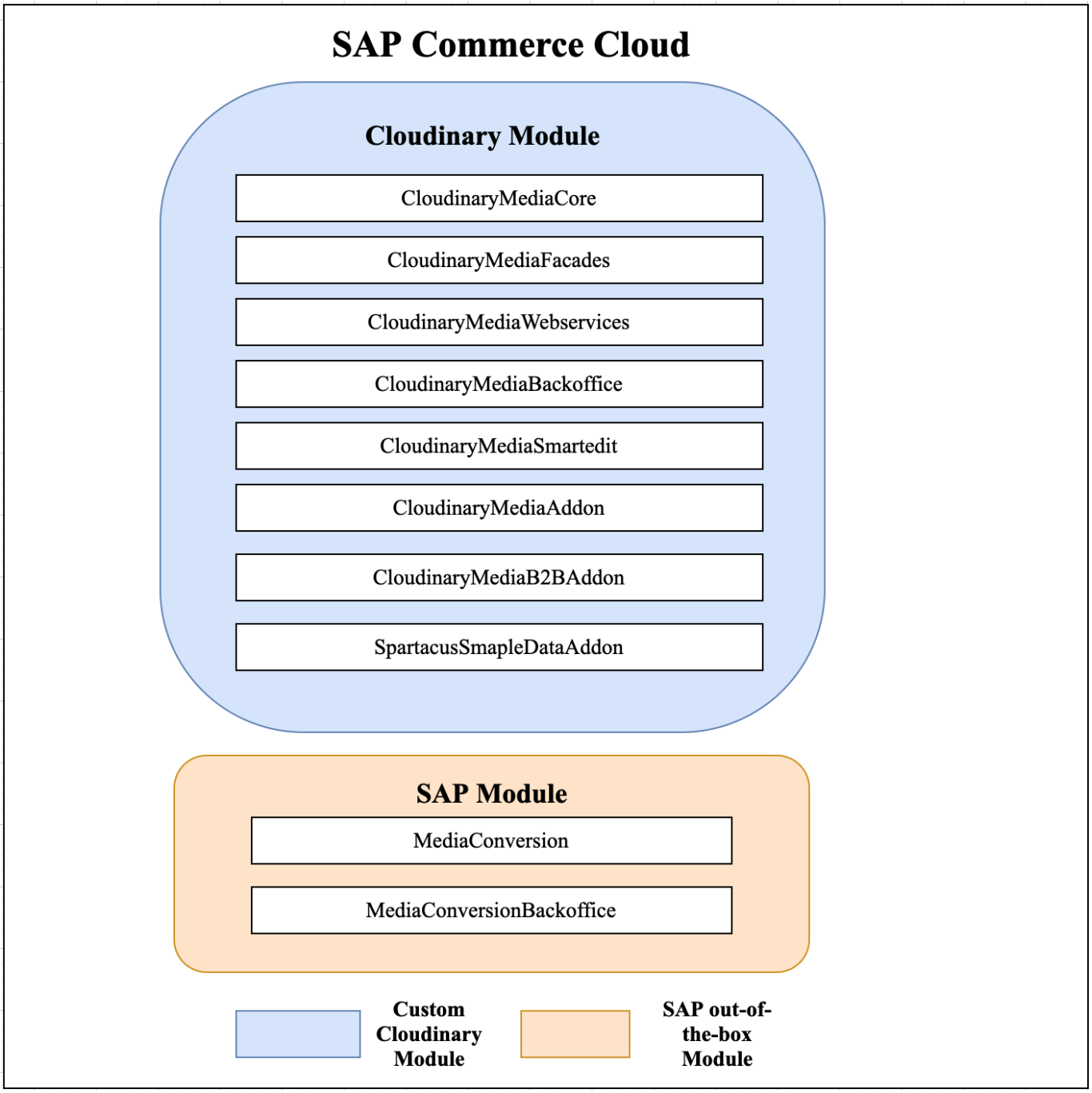
TABLE OF CONTENTS

Introduction

Document Purpose

The SAP Commerce Cloudinary Extension Technical Guide (this document) outlines the technical solution and capabilities that comprise Cloudinary digital asset management, transformations, optimizations and advanced delivery features exposed via the SAP Commerce platform with a plug and play integrated extension.

Extension View



SAP Commerce Cloud is based on a flexible modular concept that allows extending the platform with new functionality into modules. The modules for Cloudinary integration with Cloudinary extension will support a ready to use template for industry specific related functionalities and is based on custom Cloudinary modules that extends a few SAP out-of-the-box modules as detailed below.

The integration is structured into the following modules (pieces of functionality) following the best practices for extending/overriding SAP Commerce default functionality:

- **cloudinarymediacore**

  This is a Cloudinary Core module which extends the SAP mediaconversion data models and module capabilities. It will provide the basic framework and service APIs for integrating core media functionalities related to Cloudinary like the upload, delete and sync functionalities.

- **cloudinarymediafacades**

  This is a Cloudinary facade module which depends on the cloudinarymediacore module. It will provide the populators for integrating core media functionalities related to Cloudinary like the upload, delete and sync functionalities and hides the complexity of the core layer.

- **cloudinarymediabackoffice**

  This is a Cloudinary Backoffice module which extends the SAP MediaConversionBackoffice extension. It will provide the Backoffice widget customizations of media functionalities for backend business users related to media upload and transformations. The admin user will be able to manage the Cloudinary configurations in Backoffice and related customizations will be part of this module. Please refer to the User Management section for the user related access.

- **cloudinarymediawebservices**

  This is a Cloudinary web services module which extends the SAP ycommercewebservices extension and provides the REST APIs for creating media for bulk media upload. The REST API is exposed for product catalog related bulk upload.

- **cloudinarymediaaddon**

  AddOns allows users to extend or override storefront extensions without having to modify the extensions directly. In addition to simply using the preconfigured AddOns provided by SAP, we can create our own custom AddOns. To do so we create a blank AddOn, install it, and then modify its logic. This is a Cloudinary b2c addon module which is the storefront extension and provides the UI customizations for media functionalities on the frontend related to media rendering and responsive breakpoint functionality. The addon will support accelerator-based frontend functionalities.

- **cloudinarymediab2baddon**

  This is a Cloudinary b2b addon module which is the storefront extension and provides the UI customizations for media functionalities on the frontend related to media rendering and responsive breakpoint functionality. The addon will support accelerator-based frontend functionalities.

- **cloudinarymediasmartedit**

This is a Cloudinary Smartedit module which extends the SAP ysmarteditmodule extension. It provides the smartedit customization of media upload functionality for admin users.

- **spartacussampledataaddon**

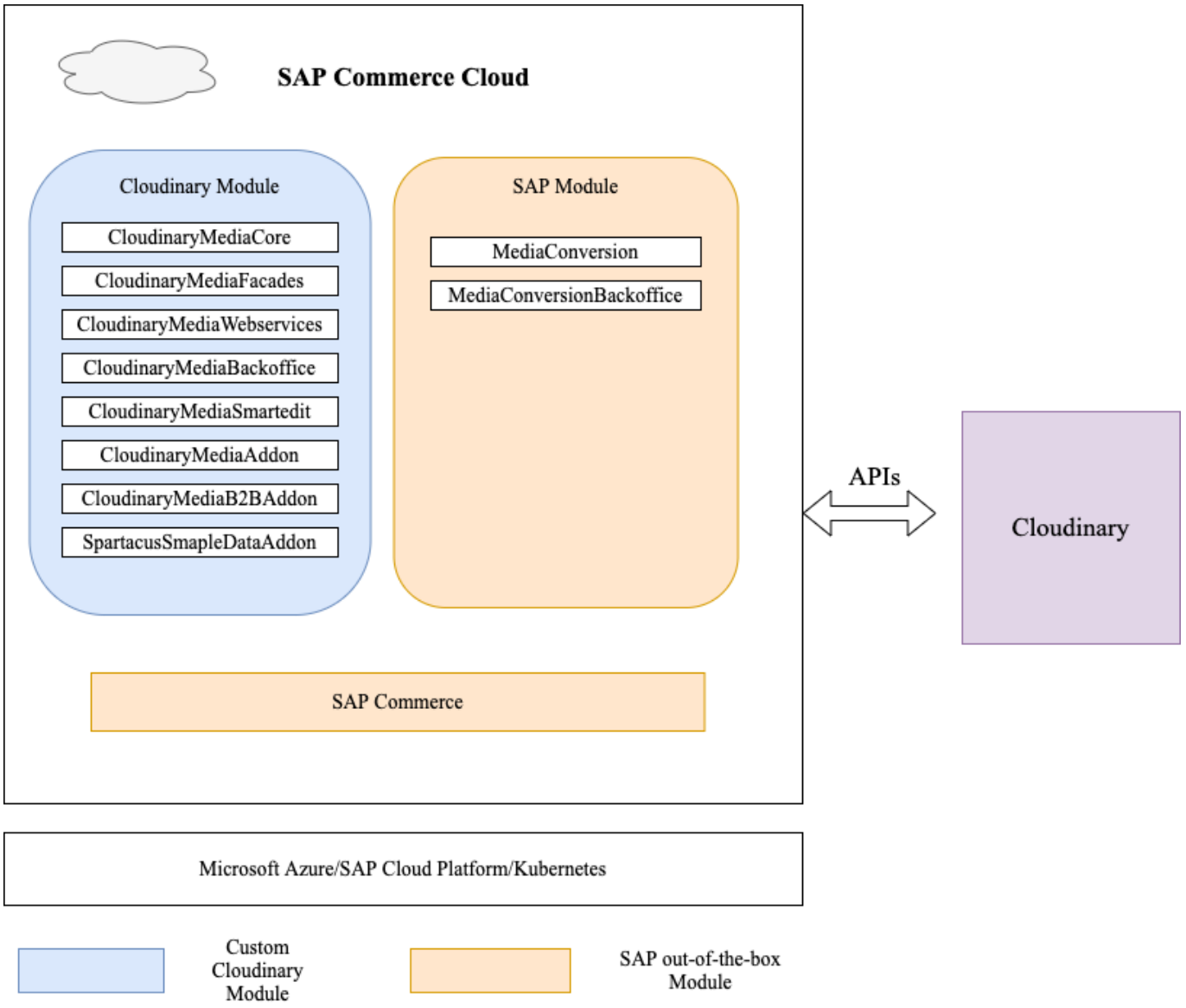
  This extension will create new WCMS sites for Spartacus that share the same product catalog with the default apparel, electronic and power tools sites but with content catalogs that have been modified specifically for Spartacus requirements. The Spartacus Sample Data AddOn is versioned and released with the Spartacus storefront library. This extension was downloaded from the Spartacus Releases page.

All the above custom extensions are bundled as a gradle recipe to easily install the modules.

The release notes will provide the exhaustive list of steps for the installation of the Cloudinary extension under different scenarios i.e. new SAP customers/existing SAP customers. Please note that the **existing customer installation will be based on specific conditions** that have been met during the SAP Commerce original implementation and might require additional steps due to the process followed during the platform extension.

Note - We can only use either cloudinarymediaaddon or cloudinarymediab2baddon while installing based on the storefront we have for the site.

Application View



SAP Commerce Cloud enables management of multiple sales channels on a single platform and delivers a consistent experience across the channels.

It provides a robust **Catalog** framework for the management of products, hierarchies, price and tax data. The Bundling Module enables defining bundles of products offered together as a package and allows different pricing for a product depending on whether it is sold individually or in a bundle. This Catalog framework is relevant for the Media managed against the Product Catalog as well as the Content managed in the Content Catalog.

The **Backoffice** Framework is designed to allow powerful business tools and user interfaces - fully leveraging the commerce platform functionality and allows the development of new features in an easy and consistent way. The Backoffice is extended to support the new widgets to support the management of the media via the Backoffice and allow for a seamless integration with Cloudinary.

The **WCMS** module will provide content management services and is customized to create custom components for rendering.

The **CronJob** functionality is used for executing tasks, called cron jobs, regularly at a certain point of time. Typically, cron jobs can be used for creating data for backups, updating catalog contents, synchronization and executing custom backend tasks. We set up jobs to allow for the data load as part of the migration to SAP Commerce.

SAP Commerce supports **Media** of all sorts such as a Flash animation file, a JPEG image, an MPEG video file, a CSV file, a text file, an XML file. The media entity is leveraged to store external asset URLs and other asset related details.

SAP Commerce allows to unify and coordinate every aspect of media management using SAP Commerce Media Conversion or the SAP Commerce Digital Asset Management. The following section is more for making the Cloudinary team aware of the Media Management concepts in SAP Commerce.

An asset means something valuable. In that context, a digital asset is defined as any item of text or media that has been formatted into a binary source. It is only when you have the right to use that digital file. There are three categories of digital assets:

- A digital asset with the textual content
- With the images it is determined as media assets
- With the multimedia it is media asset

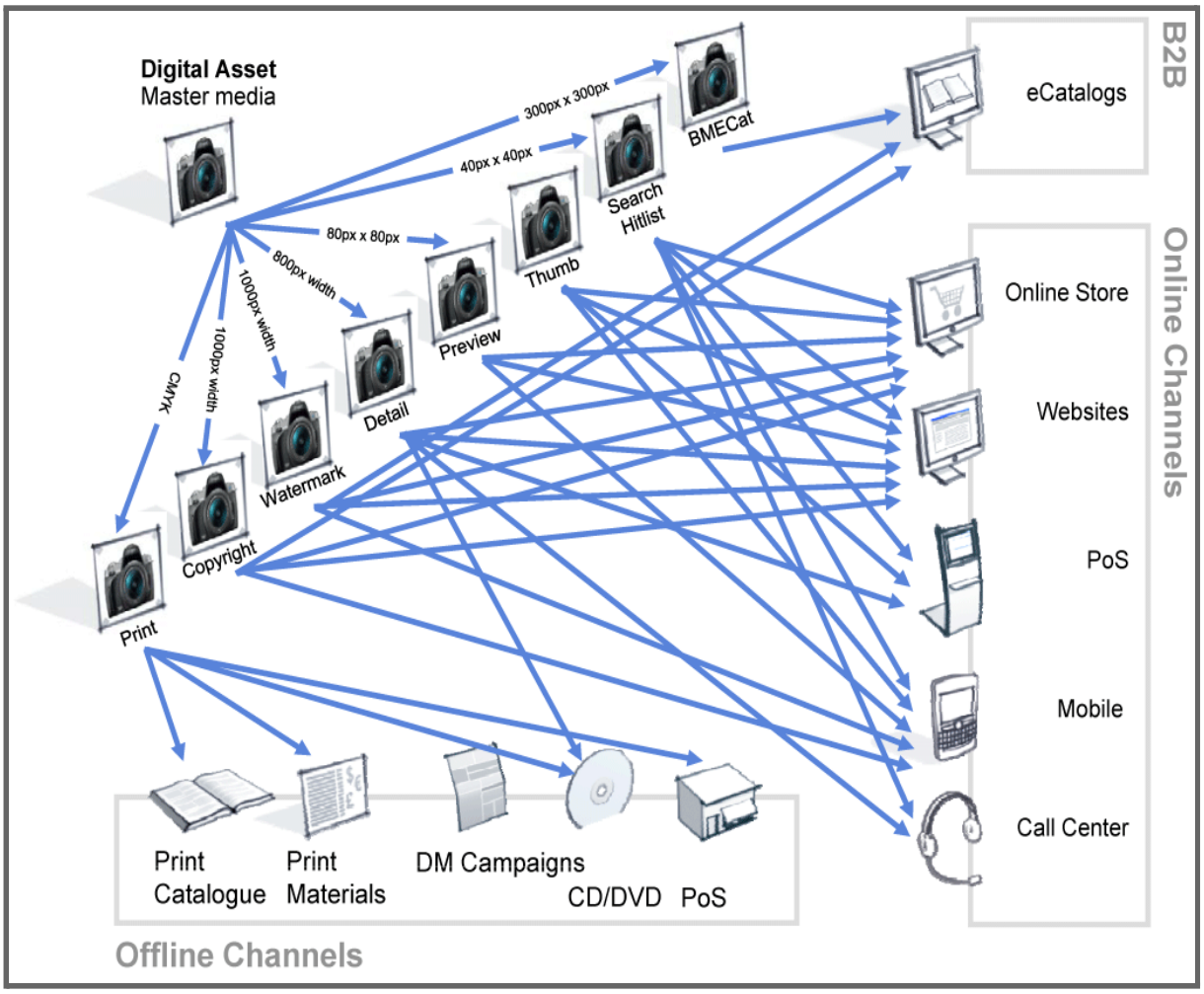
An asset is anything that adds value to an item managed in SAP Commerce. For example, if an item is a product, the asset can be product attributes such as description, name, code, approval, or references to other business items like customer reviews, up-selling products, and categories. In case an asset is a binary file, like an image or a movie, the asset is a digital asset. Typical digital assets for a product are images, logos, illustrations, audio and video files, presentations, layout page files, office documents, and spreadsheets, CAD files, and other digital files in different formats.

Digital Assets in the Multichannel Environment

Making digital assets more easily and consistently available and manageable to all distribution channels is the key for success in today's business. Growing customer requirements driven by print catalogs, websites and a convergent brand management are rapidly increasing the volume of digital media assets and the management overhead associated with. The major challenge is to manage digital assets as well as product data in a centralized and consistent manner. This typically fails, because images, graphics, and videos need to be held in many different formats, sizes and versions resulting in an abundance of formats and repositories. Multichannel publishing increases the requirements on digital assets.

A digital asset needs to be delivered:

- With different strategies: on-demand or scheduled,
- In different formats depending on distribution channel: like B2B or B2C,
- And purpose, such as on-line or off-line.



SAP Commerce is integrated to use Cloudinary, a digital asset management software suite, to manage digital assets for projects. It is possible to upload, edit, transform almost all available image/video formats by having a single source of truth or master media. The various features of Cloudinary like on the fly transformations, product gallery widget which could be used for a product definition page with custom settings, video widget, transformations (for aesthetics and optimizations) and responsive breakpoints can be leveraged with this integration.

Data View

Logical Data Model Extensions

This section details the data model changes as part of the Cloudinary integration. SAP Commerce allows for an easy extension to the existing data types that are available as part of the Platform.

CloudinaryConfig

The CloudinaryConfig is a new custom data model for configuring global Cloudinary settings. SAP Commerce Backoffice is leveraged to present global configurations which may be modified easily by the business users. The configuration values provided here are stored and used for access to Cloudinary APIs and global settings.

Data Glossary

Entities (Item Type)	SAP Commerce (Existing Type)	Description
CloudinaryConfig		A global configuration object of the Cloudinary settings for enabling/disabling a variety of settings

Attributes	Type	SAP Commerce (Existing Type)	Description
cloudinaryURL	string		The Cloudinary URL consisting of api key, api secret and cloud name parameter which is used for all Cloudinary APIs
cloudinaryFolderPath	string		The folder path in Cloudinary to which all media assets are saved/accessed
enableCloudinary	boolean		The Boolean flag to enable/disable Cloudinary
mediaUploadPreset	Preset		Populate all upload preset defined for Cloudinary Account.

cloudinaryImageFormat	string		The auto image format and encoding flag to be used in all image URLs
cloudinaryQuality	string		The auto image quality and encoding flag to be used in all image URLs
cloudinaryGlobalImageTransformation	string		The global image transformation string
cloudinaryGlobalVideoTransformation	string		The global video transformation string
enableOptimizeImage	boolean		Flag to optimize images across site
enableOptimizeVideo	boolean		Flag to optimize video across site
cloudinaryVideoFormat	enumeration		The auto video format and encoding flag to be used in all video URLs
cloudinaryVideoQuality	enumeration		The auto video quality and encoding flag to be used in all video URLs
cloudinaryContentImageFormat	string		The auto image format and encoding flag to be used in all image URLs
cloudinaryContentImageQuality	string		The auto image quality and encoding flag to be used in all image URLs
cloudinaryContentGlobalImageTransformation	string		The global image transformation string
cloudinaryContentGlobalVideoTransformation	string		The global video transformation string
enableOptimizeContentImage	boolean		Flag to optimize images across site
enableOptimizeContentVideo	boolean		Flag to optimize video across site
cloudinaryContentVideoFormat	enumeration		The auto video format and encoding flag to be used in all video URLs

cloudinaryContentVideoQuality	enumeration		The auto video quality and encoding flag to be used in all video URLs
cloudinaryResponsive	boolean		The Boolean flag to enable/disable responsive breakpoints
cloudinaryByteStep	integer		The byte steps used for responsive breakpoints
cloudinaryImageWidthLimitMax	integer		The maximum image width
cloudinaryImageWidthLimitMin	integer		The minimum image width
enableCloudinaryGalleryWidget	boolean		The Boolean flag to enable/disable gallery widget
cloudinaryGalleryConfigJsonString	string		The json string containing the cloudinary gallery widget custom settings and transformations
enableCloudinaryVideoPlaye	boolean		The Boolean flag to enable/disable the video player.
videoPlayerTransformation	string		The json string containing the cloudinary video player custom settings and transformations

Preset

The Preset is a new custom data model for storing “Upload Preset” set in Cloudinary Account. SAP Commerce Backoffice will show all the “Upload Preset” values and allow business users to choose one “Upload Preset” and set the value at CloudinaryConfiguration.

Data Glossary

Entities (Item Type)	SAP Commerce (Existing Type)	Description

Preset		A custom object to set “upload preset” at Cloudinary Configuration.

Attributes	**Type**	**SAP Commerce (Existing Type)**	**Description**
name	string		This attribute will store the name of the upload preset returned from the Admin API and populate in SAP Backoffice.

Media

The media, mediacontainer, mediaformat, mediaformat and conversiongroup objects are part of the existing data model and are extended to have Cloudinary specific attributes.
To aggregate media of the same kind but in different formats, the *MediaContainerModel* is leveraged. With this, a MediaContainerModel can be seen as a logical media having several physical representations, for example, media. Within a MediaContainerModel, there can be one representation for each MediaFormat in the system retaining the original image from which other MediaModel is derived or converted from.

This original media, also referred to as master, has a special role. It is identified by the media within the MediaContainerFormat that has neither the original nor the originalDataPK attribute set. These two attributes are manageable information stored by the mediaconversion extension. It is used to track the conversion paths and to detect changes and required reconversion.

As illustrated, upon conversion the newly created or updated MediaModel gets a reference to its original media set, original attribute. The originalDataPK is set representing the exact version of the conversion input. The conversion is also responsible for setting the derived mediaContainer, catalogVersion, and mediaFormat accordingly.

The *ConversionMediaFormat* is introduced as a new media format which extends the MediaFormat type and denotes a format that will be used for conversion by the media conversion extension.

The *ConversionMediaFormat* or one of its subtypes contains all the information required to convert a MediaModel into a new format. The main attributes to do so are:

- **conversionStrategy**: An attribute that contains the Spring bean ID of the Cloudinary MediaConversionStrategy to use. This bean must be available in the Spring context.

- **transformation**: An attribute that contains the conversion instructions or URL parameters. This attribute is dependent on the MediaConversionStrategy used.

- **Mime Type of the output**: Optional. It can be set depending on the conversionStrategy used. If no mime type is provided, the mime type of the media input is used.

ConversionGroup - Not all input media are supported or appropriate for every converted media format. For example, the banner or promotion images need to be converted to completely different formats than the product detail pictures. To reflect this, the media conversion extension introduces the concept of a ConversionGroup, which is a simple aggregation of target conversion formats attached to the MediaContainer. If this optional attribute is set on the MediaContainer, all conversion mechanisms are converted to the referenced ConversionMediaFormat, including parent formats in the conversion chain.
Specifying a ConversionGroup is optional. If none is set, all ConversionMediaFormats in the system are considered as target formats. The product pictures in a MediaContainer

leverage ConversionGroups for transformation within SAP Commerce for any media asset.

Data Glossary

Entities	SAP Commerce (Existing Type)	Description
Media	Yes	A media can be anything that can be saved on a file system, such as a Flash animation file, a JPEG image, an MPEG video file, a CSV file, a text file, an XML file, and other.
MediaContainer	Yes	MediaContainer can be seen as a logical media having several physical representations, for example, medias
ConversionGroup	Yes	A ConversionGroup is a simple aggregation of target conversion formats attached to the MediaContainer
MediaFormat	Yes	This is an identifier for a media dimension
ConversionMediaFormat	Yes	The *ConversionMediaFormat* or one of its subtypes contains all the information required to convert a Media into a new format

Entity	Attribute	Type	SAP Commerce (Existing Type)	Description
Media	Code	string	Yes	The unique code of media
	CatalogVersion	enumeration	Yes	The catalog version of the media (staged/online)
	URL	string	Yes	The local URL of the media asset stored
	cloudinaryURL	string		The full cloudinary URL of the media asset stored
	cloudinaryResourceType	string		The resource type for the media asset
	cloudinaryType	string		The type for the media asset

	cloudinaryTransformation	string		The transformation for the media asset
	cloudinaryPublicId	string		The public id of the media asset in cloudinary
	cloudinaryMediaFormat	string		The format of the media returned by cloudinary
	isCloudinaryOverride	boolean		Flag to override media level transformation

Entity	**Attribute**	**Type**	**SAP Commerce (Existing Type)**	**Description**
MediaContainer	qualifier	string	Yes	The unique qualifier of mediacontainer
	CatalogVersion	enumeration	Yes	The catalog version of the mediacontainer (staged/online)
	conversionGroup	ConversionGroup	Yes	The conversion group attached to the media container

Entity	**Attribute**	**Type**	**SAP Commerce (Existing Type)**	**Description**
ConversionGroup	Code	string	Yes	The unique code of group
	name	string	Yes	The name of the group
	supportedMediaFormats	MediaFormat		The media formats with the transformations

Entity	**Attribute**	**Type**	**SAP Commerce (Existing**	**Description**

			Type)	
MediaFormat	qualifier	string	Yes	The unique qualifier of mediaformat object
	transformation	string		The comma separated transformation values
	conversionStrategy	string	Yes	The Cloudinary strategy for transformation
	mimeType	string	Yes	The mime type of the media asset stored

Product

The product entity is an SAP type which represents the SAP Commerce product. The base product type is extended to represent product level transformations applied at the product level.

Data Glossary

Entities	SAP Commerce (Existing Type)	Description
Product	Yes	This is the SAP commerce product type which will have separate video and image transformations applicable to the specific product

Entity	Attribute	Type	SAP Commerce (Existing Type)	Description

Product	code	string	Yes	The unique code of product
	CatalogVersion	enumeration	Yes	The catalog version of the product (staged/online)
	cloudinaryImageTransformation	string		The image transformation string to be applied to the product
	cloudinaryVideoTransformation	string		The video transformation string to be applied to the product
	isCloudinaryOverride	boolean		Flag to override product level transformation
	cloudinaryImageSpinTag	string		The spinset tag from the cloudinary to show the spinset image for product.

Category

The category entity is an SAP type which represents the SAP commerce category. The base category type is extended to represent category level transformations applied at the category level

Data Glossary

Entities	SAP Commerce (Existing Type)	Description
Category	Yes	This is the SAP commerce category type which will have separate video and image transformations applicable to the specific category and subcategories

Entity	Attribute	Type	SAP Commerce (Existing	Description

			Type)	
Category	code	string	Yes	The unique code of category
	CatalogVersion	enumeration	Yes	The catalog version of the category (staged/online)
	cloudinaryImageTransformation	string		The image transformation string to be applied to the category
	cloudinaryVideoTransformation	string		The video transformation string to be applied to the category
	isCloudinaryOverride	boolean		Flag to override category level transformation

Cloudinary Gallery Component

The Cloudinary gallery component data type is a custom component type which extends the SimpleCMSComponent type. The Cloudinary gallery component represents the custom settings for the gallery widget and is used for rendering the Cloudinary gallery widget.

Data Glossary

Entities	SAP Commerce (Existing Type)	Description
CloudinaryGalleryComponent		This component contains the JavaScript and the div tags which are reusable for the product definition pages. This also contains user defined settings applicable for the component.

Entity	Attribute	Type	SAP Commerce (Existing Type)	Description
CloudinaryGalleryComponent	code	string	Yes	The unique code of CloudinaryGalleryComponent
	CatalogVersion	enumeration	Yes	The catalog version of the CloudinaryGalleryComponent (staged/online)

Cloudinary Video Component

The Cloudinary video component data type is a custom component type which extends the SimpleCMSComponent type. The Cloudinary video component represents the custom settings for the video widget and is used for rendering the Cloudinary video widget.

Data Glossary

Entities	SAP Commerce (Existing Type)	Description
CloudinaryVideoComponent		This component contains the JavaScript and the div tags which are reusable for video pages. This also contains user defined settings applicable for the component.

Entity	Attribute	Type	SAP Commerce (Existing Type)	Description
CloudinaryVideoComponent	code	string	Yes	The unique code of CloudinaryGalleryComponent
	CatalogVersion	enumeration	Yes	The catalog version of the

				CloudinaryGalleryComponent (staged/online)
	cloudinaryVideo	media		The media asset

Initial/Bootstrap Data Setup

As part of the Cloudinary Extension Implementation, a decision will be made on the environment set up based on the demo environments that will be made available by the Cloudinary Team. Any other Cloud environments will be provisioned by Cloudinary team as part of the Implementation agreement.

Cloudinary integration with SAP commerce will provide 2 ways of data setup. Both these processes may require a 1-step Data Migration that is detailed as part of the Cron Jobs section:

- Initialization - Setting up of initial/bootstrap data for new SAP Commerce customers
- Update - Setting up of initial/bootstrap data for existing SAP Commerce customers

SAP Commerce Initialization Process

SAP Commerce has its own method for initialization where it sets the system up. Once a new system has been set up, or an existing system needs to be 'reset', an administrator can choose to initialize via the SAP Commerce Administration Console. The initialization process:

- Wipes the schema that this SAP Commerce instance has been configured to use.
- Creates all tables on the schema that are needed based on the type system configuration within SAP Commerce.
- Creates necessary data for SAP Commerce types.
- Allows an extension to run a custom piece of code to trigger importing or retrieving initial data to be loaded into the database.

The initialization process is designed to give a fully working system once it is completed. It should not be necessary to import or manipulate any additional data manually after a full initialization.

Ways of importing data during an initialization

There are a number of methods which are utilized for initialization and might require a project specific update when the extension is leveraged on a Live Project:

- Impex
    - Import/Export Script. Essentially a semicolon-separated file with a header to indicate the data type and columns.
    - This is the default method for importing data into the SAP Commerce solution during an initialization.

    - Most useful for data that does not need to change. For example, a list of Titles (Mr./Mrs./etc.)
    - It allows for a very quick import of data.
    - This is the preferred way of importing data for the Cloudinary SAP Commerce integration wherein headers for the import are provided for the entities to be imported.

- DB Dump/Cleanse/Restore + Media folder copy
    - Creating a DB dump from the production environment and running a cleansing script on this output to remove any sensitive data.
    - Additionally, since SAP Commerce uses PK references in its Media folder, this will need to be copied from production too.
    - Doing this will create a production clone environment.

SAP Commerce Update Process

The update mechanism makes sure that all data that existed in the system before the update is still accessible after the update. The update process:

- Does not wipe out data or drop any tables and columns and retains the existing database
- Preserves the table and column name to which it was previously mapped
- Drops and recreates indices, if they are added or changed in items.xml
- It is typically used when new modules or functionalities are added on to an existing SAP Commerce Platform

Ways of importing data during update

There are a number of methods which are utilized for data update:

- Impex
    - Import/Export Script. Essentially a semicolon-separated file with a header to indicate the data type and columns.
    - This is the default method for importing data into the SAP Commerce solution during an update.
    - Most useful for data that does not need to change. For example, a list of Titles (Mr./Mrs./etc.)
    - It allows for a very quick import of data.
    - This is the preferred way of importing data for the Cloudinary SAP Commerce integration wherein headers for the import are provided for the entities to be imported.

Extension Functional Capabilities

The following sections detail all the functional capabilities that the extension will meet from the SAP Commerce capabilities perspective:

Cloudinary Config Widget

Overview

The Cloudinary config widget consists of the global configuration settings for SAP commerce and is used for the following -

- Enable or disable the Cloudinary module
- Global image/video transformations which are applied to all content image/video
- Global image/video transformations which are applied to all content image/video
- Enable/disable responsive image settings using responsive breakpoints
- Configure global delivery settings to control how all the images/videos look when delivered
- Configure the folder to which all the media is uploaded in Cloudinary
- Enable/disable Cloudinary product gallery widget
- Enable/disable Cloudinary video player
- View the usage statistics for the Cloudinary module

The Cloudinary config widget leverages and extends SAP commerce Backoffice for the Cloudinary config widget.

Media Upload

Overview

The upload API provides service APIs that will call the Cloudinary API for uploading media assets like image and video onto Cloudinary DAM. The asset upload via Backoffice and Smartedit for business users is done through the cloudinary media library widget integrated into the backoffice and Smartedit.

Once the asset is uploaded to Cloudinary successfully, the URL of the asset returned by Cloudinary will be split into 5 parts and stored in a media. The first part will be the config and will be replaced by a global CNAME if present and the second part will be the resource type (image/video/raw), the third part will be the type (upload/fetch), the fourth part will be the transformation string and the fifth part will be the public id of the asset. The asset will be rendered using the combination of all the parts into a URL in the media on the storefront with Cloudinary as the single source of truth. This way whenever the CNAME is changed on Cloudinary the global parameter in SAP commerce could be updated manually to ensure the URLs are unaffected and continue rendering.

Currently, the recommended approaches to upload assets to Cloudinary in SAP Commerce cloud are -

- Backoffice
- Smartedit
- Impex Framework

- REST API

Backoffice Administration Cockpit is a user-centric backend interface that enables the management of any kind of data within the SAP Commerce system. It provides specific widgets needed to create administrative and management tools, such as search, list views, or navigation tree. The backend user can log in to a Backoffice and create a media object and upload the asset to Cloudinary if enabled, else the asset is saved to the local storage.

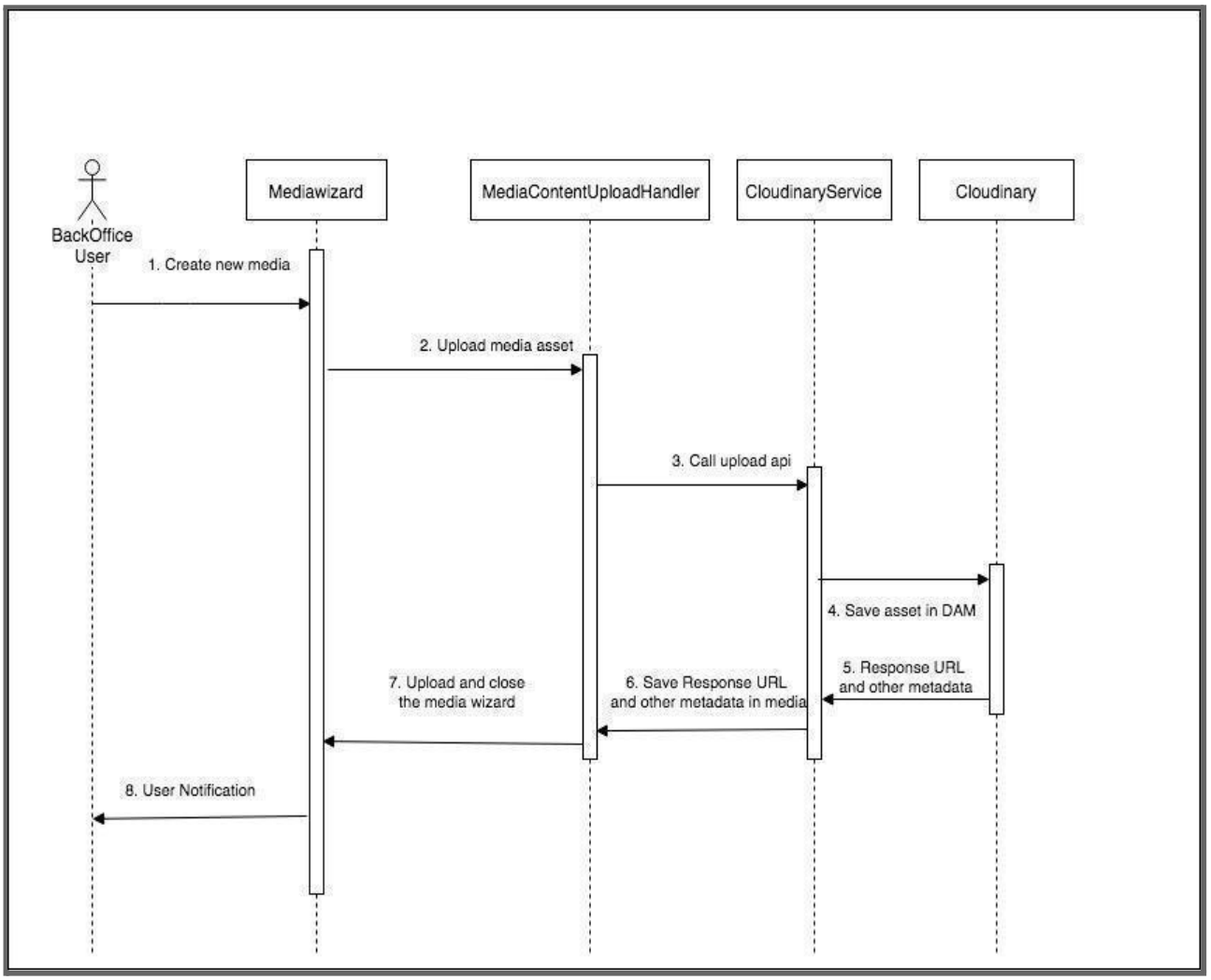
Smartedit SmartEdit is the SAP content management system editor. With each release, SmartEdit is closer to feature parity with the WCMS Cockpit and Live Edit. The business user can login to a Smartedit and create/update a media object and upload the asset to Cloudinary.

Impex Framework SAP commerce has the ability to import data in bulk via the hot folder import process. With hot folder data importing, CSV files are imported automatically when they are moved to a folder that is scanned periodically by the system. The hot folder import process will have impex headers defined in spring xml. The CSV files will map each row value to impex headers with respective columns ensuring that the correct column value for each row is saved. The system processes any files found in the folder and on successful processing will be archived, else moved to the error folder. A SAP media object with the cloudinary url will be created for each individual asset in SAP commerce.

REST API SAP commerce has the ability to import data in bulk and is based on RESTful web services. It is a next-generation commerce API that offers a broad set of commerce and data services which enables users to use and leverage the complete SAP Commerce functionality anywhere in the existing application landscape. A product catalog API will be exposed to programmatically link the Cloudinary media to products by providing the URL and product SKU. For example, if there are existing media assets with Cloudinary, then can use the product catalog API to add media assets to the product listing pages in bulk. The OAuth 2.0 authorization framework is the default authorization framework for the commerce driven Web Services under the ycommercewebservices extension. The key benefit of using OAuth 2.0 (compared to basic authentication, even over HTTPS) is that the API client does not have to save or, in some cases, even obtain the user's credentials. Instead, access tokens are returned to the client that can use refresh tokens to obtain new access tokens once they have expired.

The spring beans that would be added/overridden as part of the upload are detailed in the release notes.

The diagram below shows a Backoffice user interaction for media upload functionality that is implemented as part of the extension-



Transformation

Overview

Transformations on the media assets can be applied to the assets at various levels. The levels can be broken down into the following:

Global level content catalog transformations - A global level content catalog transformation can be configured at the “Content Catalog” tab in Cloudinary configuration. This will be applied to all assets of the content catalog being delivered by Cloudinary.

Global level product catalog transformations - A global level transformation can be configured at the “Product Catalog” tab in Cloudinary configuration. This will be applied to all assets of the product catalog being delivered by Cloudinary. This will be overwritten if the lower level transformation configurations are set up at lower levels with the isOverride flag.

Category level transformations - This is defined at the category level and will have the transformation applied at the category and subcategories media items.

Product level transformations - This is defined at the product level and will have the transformation applied at the product media items.

Asset level transformations - Asset level transformations are defined at the media and media container. Every media container will have a conversiongroup which helps in grouping various media formats at which the transformations are defined.
All the defined transformations will be applied automatically to the asset.

Specific predefined and on-the-fly transformations could be done both in storefront and Backoffice. As for pre-defined transformations, the concept of mediacontainer and conversion groups are leveraged. For every media container item there will be a conversiongroup associated which is a group of conversion media formats and will have a Cloudinary specific conversion strategy. This strategy will have the logic to pick the master media from the container and apply the conversion command or transformation on the master image URL and create a media URL with the transformation. This transformed URL will be stored in a media with the respective media format in the same container. The same media could be used to render on the user interface.

The cloudinary transformation cronjob will take care of generating the related transformations attached to the conversion groups as detailed above. This job will be triggered at regular intervals to generate the required media with transformations.

To make it easier to leverage on-demand transformation, the mediaconversion extension will be made available with a little tag library (taglib) to include for the task. The taglib will be automated by adding the following snippet to the web extension buildcallbacks.xml file. This is a small snippet that will be extended as part of the release notes for the Cloudinary extension:

```
<macrodef name="CloudinaryDAMAddon_before_compile_web">
    <sequential>
      <mkdir dir="${ext.yourextension.path}/web/webroot/WEB-INF/tld" />
        <copy
        file="${ext.mediaconversion.path}/resources/mediaconversion/taglib/mediaconversion.tld"
        tofile="${ext.yourextension.path}/web/webroot/WEB-INF/tld/mediaconversion.tld"/>
   </sequential>
</macrodef>
```

The following tags are provided as part of the taglib library -

- **URL Tag**

  URL tag is a tag to output or set a variable to the URL of a media in a given format. The URL generated either points directly to the media or, if the specified format has to be generated first, to the mediaconversion extension servlet that makes transformation in a different thread.

The following table describes the attributes:

Name	Description	Required	Runtime Expression	Type
container	The container in which the media in question resides.	true	true	MediaContainerModel
format	The qualifier of the output format (MediaFormat Model). If no format is specified or the specified format is the empty string, the master media of the container is used.	false	true	java.lang.String
scope	The scope of the variable to set. One of page, request, session, or application.	false	false	java.lang.String
var	The name of the variable to set.	false	false	java.lang.String

- **HTML Image Tag**

  HTML Image Tag renders a HTML <img> element that references a (converted) media in a specified format. The generated URL either points directly to the media or, if the specified format has to be generated first, to the mediaconversion convert servlet that makes conversion in a different thread.

  The following table describes the attributes:

Name	Description	Required	Runtime Expression	Type
align	Optional attribute for the generated HTML <img> element. The possible values for this attribute are top, middle, bottom, left, and right.	false	true	java.lang.String

alt	Optional alternative text to override alternative text set on the media.	false	true	java.lang.String
border	Optional attribute for the generated HTML <img> element.	false	true	java.lang.String
container	The container to output.	true	true	MediaContainerModel
cssClass	Optional CSS class to be used.	false	true	java.lang.String
format	The qualifier of the output format (MediaFormatModel). If no format is specified or the specified format is the empty string, the master media of the container is addressed.	false	true	java.lang.String
height	Optional height of the generated <img> element. This height is not obeyed in an image scaling, for example it is only used in the generated HTML output.	false	true	java.lang.String
id	Optional attribute for the generated HTML <img> element.	false	true	java.lang.String
ismap	Optional attribute for the generated HTML <img> element.	false	true	java.lang.Boolean
longdesc	Optional attribute for the generated HTML <img> element.	false	true	java.lang.String
name	Optional attribute for the generated HTML <img> element.	false	true	java.lang.String
style	Optional (CSS) style attribute.	false	true	java.lang.String
title	Optional attribute for the generated HTML <img> element.	false	true	java.lang.String
usemap	Optional attribute for the generated HTML <img> element.	false	true	java.lang.String
vspace	Optional attribute for the generated HTML <img> element.	false	true	java.lang.String

width	Optional width of the generated <img> element. This width is not obeyed in an image scaling, for example it is only used in the generated HTML output.	false	true	java.lang.String

- **Dynamic HTML Image Tag**

  The dynamic HTML Image Tag renders an HTML <img> element that references a (converted) media in a specified format as above. The only difference being that instead of the format qualifiers there is a transformation string that can be specified in the tag. The generated URL either points directly to the media or, if the specified format has to be generated first, to the mediaconversion convert servlet that makes conversion in a different thread.
  When converting media on the fly on a website, it is important to separate the HTML rendering request from the request that makes the conversion.

  As shown in the following diagram, the complete flow is separated into three phases:

  - Upon the page request (1) the controller, servlet, or JSP checks for the availability of the media in the required format (2). If such a media exists, its URL is returned as usual. If not, a substitute URL is embedded in the HTML page and returned (3).
  - The client browser requests the media from the substitute URL (4). A simple servlet makes the transformation (5) and sends back a redirect to the URL of the newly created image (6).
  - The client requests the media (7) and retrieves it (8).

  The spring beans that would be added/overridden as part of the transformation are detailed in the release notes.

Media Management and Sync

Overview

The media sync is an important functionality which is needed to keep both the DAM and SAP CX in sync with the media assets. As Cloudinary is the single source of truth for all media assets when switched on, SAP Commerce will serve as a fallback for all new

media assets when Cloudinary is switched off. Once Cloudinary is switched on again all the existing media assets on SAP commerce not present in cloudinary will be automatically synced to Cloudinary.

Currently, the recommended approaches to sync assets to Cloudinary in SAP Commerce cloud are -

- **On Demand** - When Cloudinary is active and a user requests for an image then the sync will happen to Cloudinary from SAP Commerce
- **Bulk Sync** - This is automated via a cronjob. As part of the automation the cronjob will have a business logic which will pick up all media assets in SAP commerce which are present in the local storage once Cloudinary is switched on. The assets will be synced on to Cloudinary using the upload API. This is uni-directional from SAP Commerce to Cloudinary.

  For bulk sync from Cloudinary to SAP Commerce for customers who already have assets in Cloudinary and are migrating their commerce platform to SAP Commerce, REST API web service for product catalog will be used. Please refer to the upload API section for further details.

The CronJob functionality is used for executing tasks regularly at a certain point in time. Typically, CronJob can be used for creating data for sync, backups, updating data model entities, executing business processes, etc. A brief description of the SAP Commerce Cronjob concept is as follows:

Product Gallery Widget

Overview

The SAP commerce product gallery functionality is used for configuring Cloudinary product galleries with a variety of media assets like image, video and 3d/360 images. This is enabled through a custom CMS (Content Management System) component which could be configured on any of the storefront pages with certain parameters.
The product gallery widget CMS component consists of a controller and view in which the tags and JavaScript provided by Cloudinary will be placed for rendering. The component can be used in the product definition page and reused for all products.

The cloudinary gallery widget will use a json string in the global configurations to add cloudinary specific gallery parameters to the component.

The widget uses tags specific to product skus with format sap_sku_<product code>.
Every product when associated with a media will be tagged with the related media in Cloudinary. For bulk media tagging, a tagging cronjob will take care of tagging the assets in Cloudinary.

The Cloudinary product gallery widget can be edited in backoffice.

Video Player Widget

Overview

The SAP commerce video widget functionality is used for media assets like video using a self-hosted player from Cloudinary. This could be enabled through a custom CMS (Content Management System) component which could be configured on any of the storefront pages with certain parameters. The custom Cloudinary Configuration adds few dynamic custom attributes for Video player like enableCloudinaryVideoPlayer and videoPlayerTransformations and could be extended to add more functionalities. The custom attributes can be edited by the business user as per the requirement.

The CMS component consists of a video tag in which the <video> tags are placed for rendering. The CMS component has a video asset attached in a media which is rendered wherever the component is placed in the frontend. The Cloudinary video widget can be edited in the backoffice.

Responsive Breakpoints

Overview

The responsive breakpoints are configured globally within the Cloudinary configuration widget in Backoffice. When breakpoints are enabled, the configured values are picked up by the JavaScript library provided by Cloudinary and images rendered in a responsive manner.

Cloudinary provides an automatic JavaScript solution for handling dynamic generation of images that match the required dimensions according to the exact image width available and the DPR (Device Pixel Ratio) of every device. Cloudinary's JavaScript library includes a method that dynamically builds the dynamic image URLs and works as follows:

- A Cloudinary dynamic manipulation URL is automatically built on-the-fly to deliver an image that is scaled to the exact available width and resolution.
- If the browser window is consequently enlarged then new higher resolution images are automatically delivered, while using breakpoint steps (every 100px by default) to prevent loading too many images.
- If the browser window is scaled down, browser-side scaling is used instead of delivering a new image.

This feature allows users to upload one high resolution image to Cloudinary, and have it automatically adapted to the resolution and size appropriate to each user's device or browser on-the-fly.

Cloudinary responsive breakpoints integration with Accelerator storefront

The steps to include the JavaScript are detailed below, this will also be detailed as part of the Release notes -

- Include the Cloudinary JavaScript library in the custom addon extension

- Set the image tag parameters data-src, width and crop parameters and cld-responsive class style automatically with a custom JavaScript
- Add a call to the Cloudinary responsive JavaScript method at the end of the HTML page.

Cloudinary responsive breakpoints integration with Spartacus storefront

If you are using the Accelerator storefront, no further changes are needed to achieve responsive images with Cloudinary. If you are using the Spartacus storefront, there is some development work required to integrate Cloudinary responsive functionality with the Media Components in Spartacus.

To learn more about Media Components, see the Spartacus documentation:

https://sap.github.io/spartacus-docs/media-component/

Refer to the following media configuration and service files from Spartacus:

https://github.com/SAP/spartacus/blob/develop/projects/storefrontlib/recipes/config/default-media.config.ts

https://github.com/SAP/spartacus/blob/develop/projects/storefrontlib/shared/components/media/media.service.ts

Media components use srcset in <img> elements to show images responsively, and srcset values are populated based on mediaFormats in mediaConfig. However, Cloudinary image breakpoints are calculated dynamically based on the configuration in Backoffice.

Customize Media Components in Spartacus

To use Cloudinary's responsive breakpoints solution in Spartacus, follow these steps:

1. In **app.component.ts** (created when Spartacus is installed), include "https://unpkg.com/cloudinary-core@latest/cloudinary-core-shrinkwrap.js"

   For production use, we recommend that you use a specific version, and not the latest, to protect yourself from any breaking changes.

2. Add code to make the following changes to any **<img>** elements within the app, after the view is initialized:
   a. Remove the **srcset** attribute
   b. Add **class="cld-responsive"**

   For example, you could add custom logic into **ngAfterViewInit()** of **app.component.ts** to achieve this.

3. In **app.component.ts**, call the **responsive()** method:

```
<script type="text/javascript">
    var cl = cloudinary.Cloudinary.new({cloud_name: "demo"});
    // replace 'demo' with your cloud name in the line above
    cl.responsive();
</script>
```

Note: the **Image Breakpoints** tab in the Cloudinary Configuration section of Backoffice is used to configure the breakpoints for both the Accelerator and Spartacus storefronts.

Integration View

Overview

SAP Commerce will use the following APIs to integrate with Cloudinary -

- Upload API
- Admin API
- Get Upload Presets API

Upload API

The Upload API is used for uploading resources, creating new resources such as images, videos, ZIP files and sprites, and manipulating existing resources. The Cloudinary server-side Java SDK is used for the uploads. Uploading is performed synchronously, and once finished, the uploaded asset is immediately available for transformation and delivery and stored in SAP media object.

An upload API call returns a response that includes the HTTP and HTTPS URLs for accessing the uploaded file, as well as additional information regarding the uploaded asset. Among these are the assigned Public ID and current version of the asset (used in the Media Library, Admin API, and for building transformation and delivery URLs), the asset's dimensions, the file format and a signature for verifying the response. The response URL and other metadata are stored in the SAP media object.

Admin API

The Admin API is used for getting the usage statistics for the user's Cloudinary account. The Cloudinary server-side Java SDK is used for all the admin related API calls.

For each Admin API call, standard HTTP headers are returned with details on the current usage statistics, including the per-hour limit, remaining number of actions and the time the hourly count will be reset.

SAP Commerce will expose consumer REST APIs for consuming bulk media URLs for product catalog -

REST API

SAP commerce has the ability to import data in bulk and is based on RESTful web services. It is a next-generation commerce API that offers a broad set of commerce and data services which enables users to use and leverage the complete SAP Commerce functionality anywhere in the existing application landscape. A product catalog API is exposed to programmatically link the Cloudinary media to products by providing the URL and product SKU. For example, if there are existing media assets with Cloudinary, then can use the product catalog API to add media assets to the product listing pages in bulk. The OAuth 2.0 authorization framework is the default authorization framework for the commerce driven Web Services under the ycommercewebservices extension. The key benefit of using OAuth 2.0 (compared to basic authentication, even over HTTPS) is that the API client does not have to save or, in some cases, even obtain the user's credentials. Instead, access tokens are returned to the client that can use refresh tokens to obtain new access tokens once they have expired.

GET UPLOAD PRESET API

This API fetches all the upload presets defined for your Account. The Cloudinary server-side Java SDK is used for these API calls.

Get upload preset API call returns a response that includes all upload presets of the Account with attributes like name, unsigned and settings. SAP commerce creates an instance for each preset in the custom Preset object.

Upgrade Path View

Overview

Prior to the release of SAP Commerce 1808--which uses the format YYMM– SAP Commerce used a versioning system to explain whether a particular release is a major, a feature, a minor, or a patch release.

The extension is designed/implemented to be compatible with Version <<1811>> and onwards.

Any upgrades of the SAP Commerce Platform beyond the current version 2005 on which the extension is compatible, will have to be reviewed as part of the Support agreement of the extension with the Implementation team.

Understanding the process by reviewing Supported Releases can help to determine how

behind the version is from the latest release and the potential effort required to upgrade. For example, a patch release will not typically require any code change outside of switching out the platform binaries. This should also mean a quicker regression test.

In comparison, a major upgrade might require refactoring of the customizations based on the changes introduced by the SAP Product team, and an extensive set of regression to make sure nothing is negatively impacted in the extension.

Monitoring and Analytics

User agent and other plugin related information like the SAP commerce version and Cloudinary version and SDK version will be sent in the below scenarios -

- When the user opens the media library widget
- When the user enables/disables cloudinary in backoffice

The User Agent will be in the following format: [CloudinarySAP/[plugin version] ([SAP env info incl. version]) [SDK you are using/version]]

E.g. - CloudinarySAPCC/1.1.0 (SAP Commerce Cloud 2005) CloudinaryJava/1.26.0

Cloudinary exposes the below endpoints which are called every time the user enables/disables Cloudinary in backoffice:

- https://analytics-api.cloudinary.com/sap_plugin_activated
- https://analytics-api.cloudinary.com/sap_plugin_deactivated