

Live Story

Version 20.1.0 - Integration Guide



Table of Contents

Summary	2
Component Overview	3
Functional Overview	3
Shoppable content	7
Use Cases	8
Limitations, Constraints	10
Compatibility	11
Privacy, Payment	11
Implementation Guide	12
Setup	12
Configuration	12
Custom Code	14
SiteGenesis Architecture	16
Locale-driven shopping data	17
Upgrade from previous releases	18
Upgrade from 19.1.0	18
External Interfaces	19
Firewall Requirements	19
Testing	19
Operations, Maintenance	19
Data Storage	19
Availability	19
Support	20
User Guide	20
Roles, Responsibilities	20
Business Manager	20
Storefront Functionality	21
Known Issues	22
Release History	23

Summary

Live Story is an editorial commerce platform. It aggregates existing and user-generated content, editorialize it and make that content live, multi-channel, and shoppable.

Live Story's design-forward approach seamlessly integrates into digital platforms: easy to use, easy to customize, easy to engage.

The purpose of this component is the integration between Live Story and Salesforce, in order to allow brand customers to save their visual contents on Commerce Cloud and embed them into their websites.

A Live Story's subscription and then a platform account are compulsory to enable the integration, so additional costs will be charged to the customer by Live Story Inc. at the end of the process. To obtain more information, a contact form to the Sales office can be filled on the company website (<https://www.livestory.nyc/>).

As described in the next sections, the integration requires OCAPI access through the specific Business Manager section and a little addition to StoreFront code.

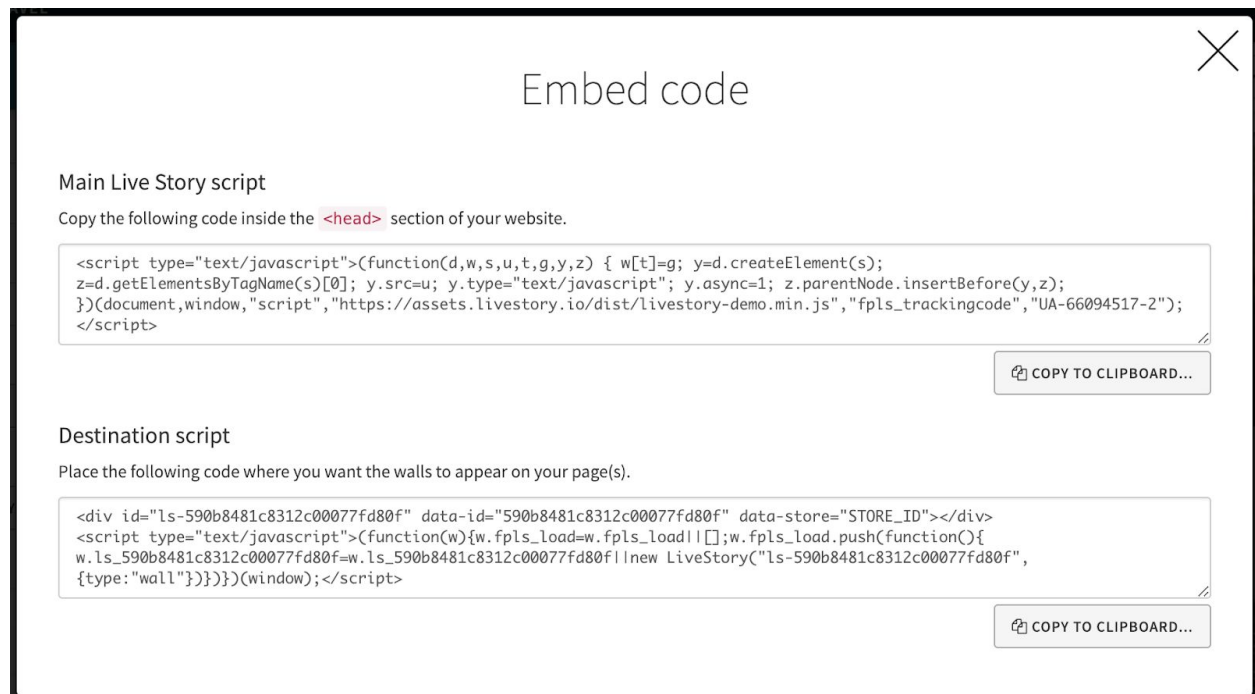
Component Overview

Functional Overview

From a technical side, the purpose of this component is the creation of Commerce Cloud's Content Assets starting from any Live Story's layouts.

A layout is a grid of image and text blocks, published to a web page by embedding two HTML and Javascript code snippets:

- **Main script:** the core of Live Story browser component, to be included in every page in the head or footer section
- **Embed code:** the initialization of a specific layout in a specific section of the web page

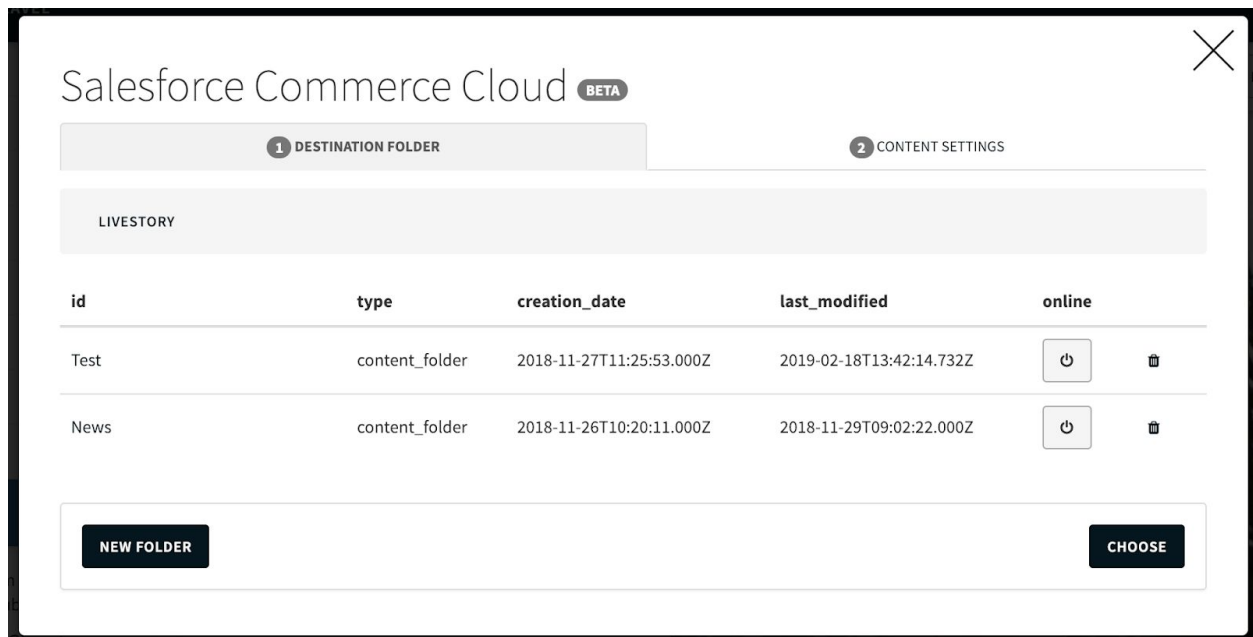


The image above shows a portion of the Live Story Console, where a user can see and copy the necessary codes to paste on the web page to embed a layout.

On a standard website, without using the integration, the workflow happens on *client-side*: the browser loads Live Story main script and then initialize the layout within the provided embed code in the included HTML container. Data comes as payload from an API request to the backend service.

Except for the main script, prior to this integration, a Business Manager user had to manually copy and paste the embed code in a new content asset in order to publish a layout.

This could lead to errors or redundant work: in a multi-language layout, for example, the embed code can contain an attribute called `data-lang` to load a specific content language. So the user is supposed to copy and manually edit the embed code for every language version it needed.



The integration simplifies this process: the user creates the layout, open the Salesforce Commerce Cloud popup, specify the content folder and attributes, click on *Create* and the asset is immediately ready to be included in a site slot.

Salesforce Commerce Cloud BETA

1 DESTINATION FOLDER

2 CONTENT SETTINGS

SELECTED FOLDER: LIVESTORY

ONLINE: ON

SYSTEM ATTRIBUTES

ID

ls-#trailescape

ADD

ADD ATTRIBUTE GROUP

LIVESTORY

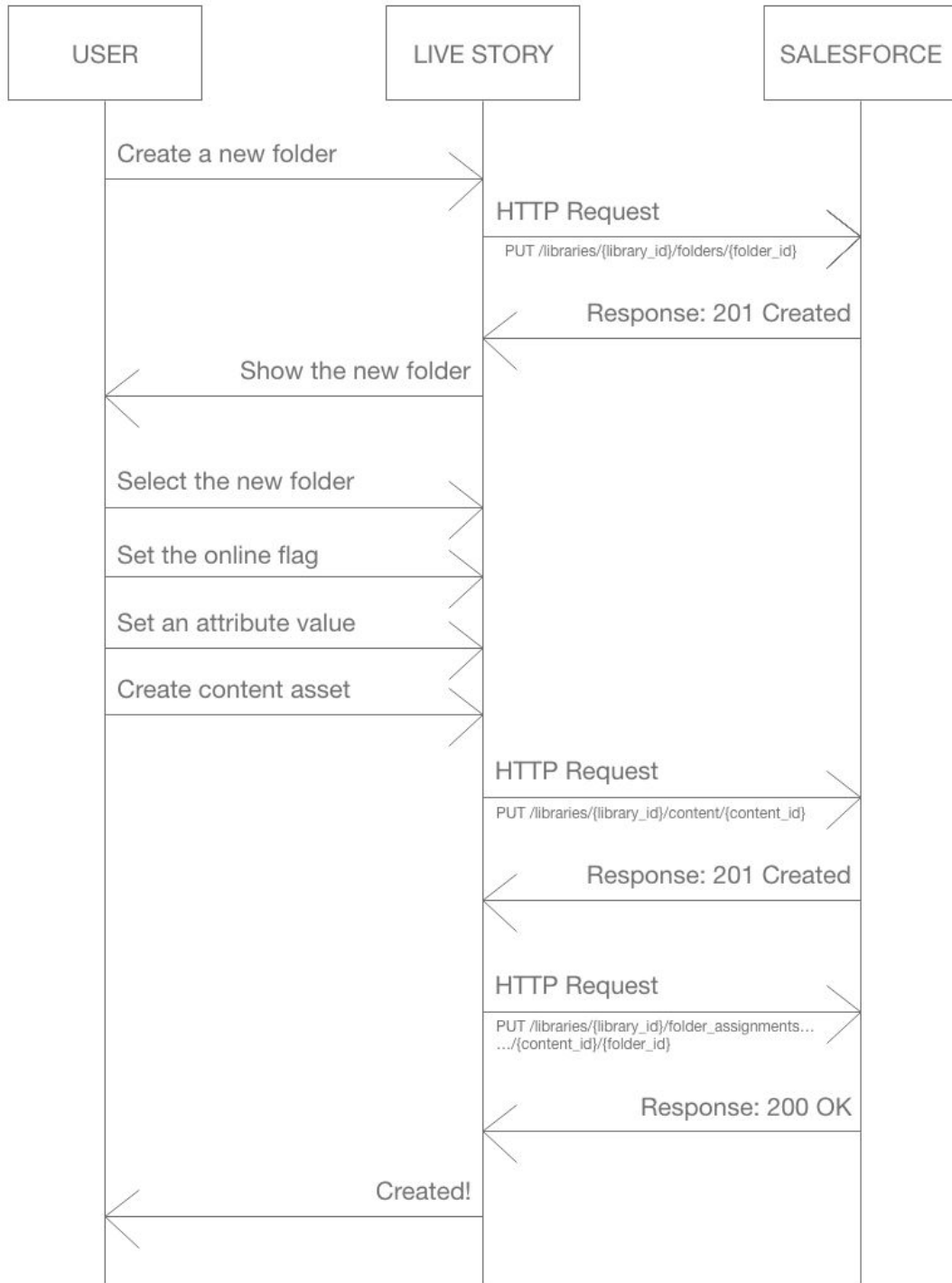
LIVE STORY - TITLE 🌐

LIVE STORY - ABSTRACT 🌐

CREATE

Last saved: Dec 3rd 2018, 9:49:38 am

Under the hood, a Live Story backend service interact with Salesforce Commerce Cloud Data API (OCAPI) to apply user commands and create the asset. The sequence chart below shows an example flow between user, Live Story and Salesforce.



With this flow, a Live Story layout assumes the capabilities to provide an *SEO-friendly* content, since the HTML code is provided from Commerce Cloud server and it becomes readable from any search engine crawler service.

Client-side, Live Story core module replaces the HTML content with a fresh version loaded from the REST API payload, following the standard workflow. With this architecture, the updates of a Live Story content becomes immediately available on StoreFront and there is no need to wait for the *staging-to-production* replication time of Commerce Cloud infrastructure.


The supported requests set is available as *Postman Collection* (<https://www.getpostman.com/>), equipped with payload samples. For more information, read the *Testing* section of this documentation.






Shoppable content

If customer's environment exports its product catalog in the form of a Google Merchant¹ feeds set, the Live Story account can be configured to automatically sync those feeds with an internal brand's product database.

In this case, the customer can make its content *shoppable*, by associating pictures or social posts with the previously imported products so that, in the final layouts, they will contain a *call-to-action* link leading the visitors to the specific product pages.

Subscriptions

 Feeds are processed automatically once a day

ACTIVE SUBSCRIPTIONS			
	Url	Type	Store code
	https://example.com/feeds/it.xml	xml	it
	https://example.com/feeds/uk.xml	xml	uk
	https://example.com/feeds/de.xml	xml	de
	https://example.com/feeds/at.xml	xml	at
	https://example.com/feeds/fr.xml	xml	fr

¹ https://support.google.com/merchants/answer/188493?ref_topic=3163841

The screenshot shows a detail of the Live Story Console, the section where the user can manage its feed subscriptions. The most important value to notice is presented in the **Store code** column, on the right.

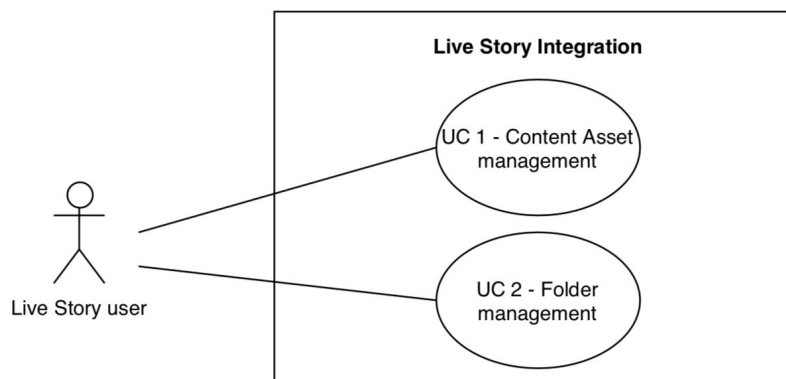
The store code values registered in this Console section can be used in the *frontend embed code* in an attribute called `data-store` that specifies to the Live Story engine what feed to use for fetching the data to be used as product information.

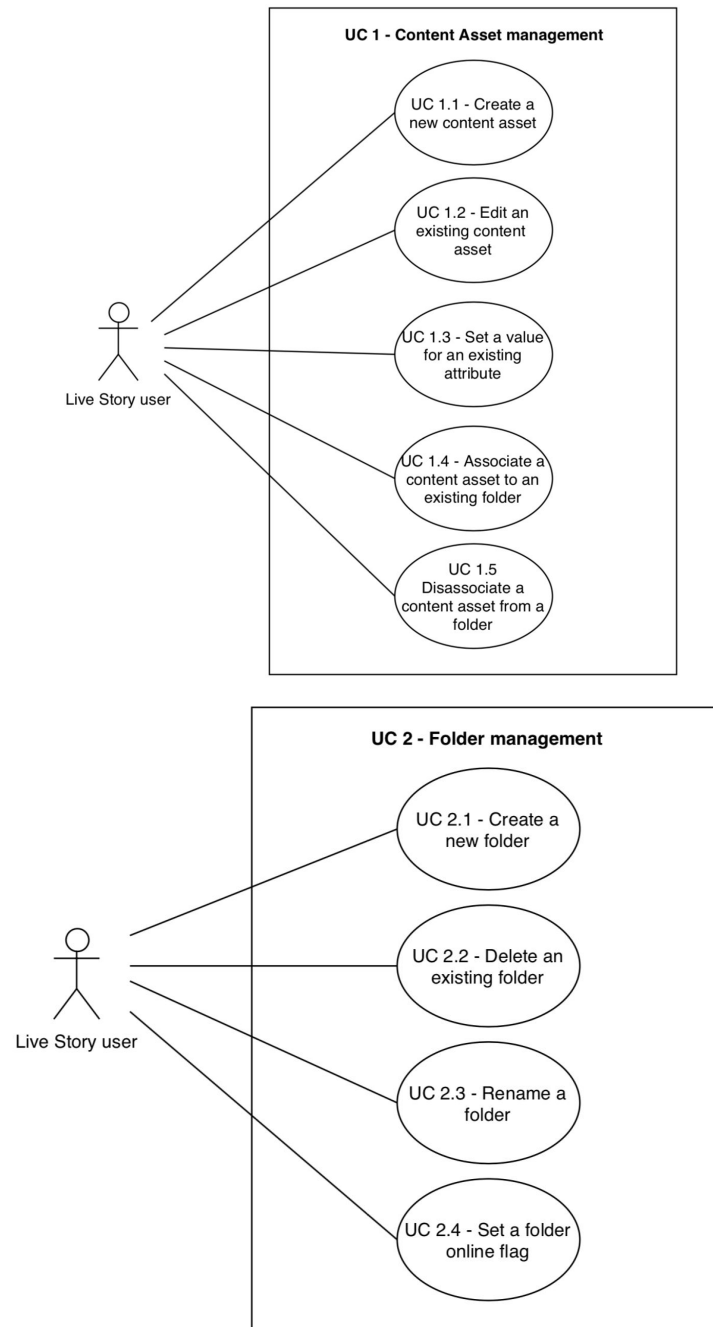
This is used to display the correct data according to site language, store, currency and other typical **locale-related e-commerce information**. In the *Implementation Guide* section will be explained an example of a possible Commerce Cloud scenario to drive Live Story merchant data based on current locale.

Use Cases

The main requirement, as briefly described in the previous overview, is to logically associate a Live Story layout to a Salesforce content asset. This concept extends the possible use cases to other secondary needs, like manage folders, set content attributes, manage folder associations, and so on.

The following UML diagrams graphically list the considered use cases for this integration.





The derived requirements from the considered use cases regard the management of the linked content assets and basic folder management, limited to create and update operations in addition to the possibility to change content to folder associations.

Limitations, Constraints

In order to associate a Live Story layout to a Salesforce content asset, the layout's HTML code must be copied into a content asset's field. During the design and development, a few limitations about this detail have been encountered:

- The default value type of Content object's `body` field is HTML, but through OCAPI this type is available only with read permissions². Live Story integration cannot use it to save its layout code.
- While looking for an alternative field to save the HTML code, the type `String` is strongly discouraged due to its length limitation³.

The resulting solution involves the creation of a custom Content attribute of type `Text` and the configuration in Live Story integration to use that field as a layout code destination. See *Configuration* section in *Implementation* chapter for more details.

Another encountered limitation regards the `PUT` method in HTTP requests to OCAPI, that is available only for `dev` and `sandbox` instances⁴. To simplify the integration development, the `PUT` method is never used but always replaced with a `POST` request enriched with the documented `x-dw-http-method-override` set to `PUT` value.

In addition, this workflow does not provide synchronization for a content asset created by Live Story and updated using Business Manager. In this scenario, in case of an update using Live Story Console, the previous changes could be overwritten.

Encoding must be kept off in Live Story's involved templates, since the platform needs to output HTML code in the content assets created by the integration.

Finally, as described in the *Summary* section, the Live Story platform account is not part of this component, it requires a separate subscription and contract with Live Story Inc. company.

²

<https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/current/usage/CustomProperties.html>

³

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/content/b2c-commerce/topics/site_development/b2c_supported_data_types.html

⁴

<https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/current/usage/HttpMethods.html>

Compatibility

The integration has been tested with the following environments:

- OCAPI Data API versions 17.6, 18.3, 18.7, 18.8, 19.5 and 20.4
- Commerce Cloud version 20.5, from which come the screenshots of this documentation
- Compatibility mode 19.10
- SFRA 5.0.1

Privacy, Payment

Customer profile data is never accessed by this integration, including no credit card is stored or processed nor within Commerce Cloud or Live Story platform.

Implementation Guide

Setup

Once created a Live Story account with a subscription that includes the Salesforce integration, there is no need for a particular setup procedure. The only necessary steps involve the configuration of Live Story API usage and Salesforce OCAPI permission, in addition to the storefront changes to output the layout code. These steps are described in the following *Configuration* and *Custom code* sections.

Configuration

The Live Story Console has a dedicated section for Salesforce integration configuration, accessible by clicking on the brand name on the top left corner, then choosing *Brand configurations* and then *Platform integrations*.

This page contains three areas: *Source*, *Library* and *Content Asset Settings*. The first, *Source*, arranges the settings related to API URL and credentials.

The URL to set is the Data API base URL, specifying the API version to use. The credentials and the instance OCAPI permissions can be tested by clicking the specific buttons. Also, the eventually cached authentication token can be reset with the *Reset Server Authentication* button. For more information, visit the *Testing* section of this documentation.

The Section *Library* contains settings related to Content Asset hierarchy. In order to load the folder structure and then choose a possible content allocation in a specific folder, a Library ID and a Root Folder ID must be configured.

LIBRARY

LIBRARY ID

RefArchSharedLibrary

ROOT FOLDER ID

livestory

The last section, *Content Asset Settings*, allows configuring the required attributes necessary for Live Story integration as well as other default attributes or groups to show during the creation of a new Content Asset.

Content Asset Settings

CONTENT BODY

c_lsBody

CHANGE

CONTENT THUMBNAI

c_lsThumbnail

CHANGE

CONTENT TEMPLATE

content/livestory/livestoryContent.isml

SYSTEM ATTRIBUTES

ADD ATTRIBUTE

ATTRIBUTE GROUPS

ADD GROUP

livestory ✕


Here is the list of configurable attributes:

- **Content Body (required):** as described in chapter *Component Overview*, section *Limitations, Constraints*, this field is required to save a layout code into the asset. It must be of type `Text`.
- **Content Thumbnail (required):** every layout has a thumbnail that should be saved in the Content Asset as a preview.
- **Content Template (required):** the necessity of a custom body field requires a dedicated template to output the layout to the StoreFront. Its scope is only print the specified variable instead of the default body field, an example can be found on the *Custom code* section or in the cartridge packages, at paths `/cartrdiges/int_livestory` and `/cartrdiges/int_livestory_sfra`.
- **System attributes:** additional system attributes can be added to the new content asset wizard and saved to Salesforce by Live Story.
- **Attribute groups:** same for *System attributes* but for Content's attribute groups.



In addition, a set of default custom objects must be imported. It can be found in the cartridge package, at path `/metadata`. After the import process, visit the Business Manager console, select the site and select from the *Merchant Tools* menu the item *Custom Preferences* under section *Site Preferences*. Then select *livestory* preferences group and set the following options:

- **IsActive:** enable the cartridge for the site
- **IsBrandCode:** paste here your subscription brand code.

Merchant Tools / Site Preferences / Custom Site Preference Groups /

Live Story 

Instance Type
Sandbox ▼

Search by IDs...  

Name	Value
Live Story - Status (IsActive)	Yes
Live Story - Brand Code (IsBrandCode) (String)	demo

In addition, OCAPI permissions must be configured in Commerce Cloud Business Manager, at the section *Administration, Site Development, Open Commerce API Settings*. Select *Data* in the type menu and paste the permission scheme to the textarea. Please remind that it is not possible to select a *context* different from `Global` due to Data API limitations⁵.

An example of OCAPI configuration code, ready to be imported, can be found in the cartridge package. It's included in the *Site Import* package at path `/metadata`.

Custom Code

Locale-driven shopping data

In order to enable Live Story's shoppable features, as described in section *Functional Overview, Shoppable Content*, a list of feed subscriptions has to be provided in the Console. The system integrator must design it in order to cover the customer needs in terms of languages, currencies, catalogs and product scope in Live Story (master product or color-variant product).

⁵

<https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/current/usage/OCAPISettings.html>

Live Story, as a *best practice*, suggests a design that usually works well with a common Commerce Cloud environment: **a subscription for every e-commerce configured country** as a compromise between versatility and implementation effort.

With this scenario, the `data-store` attribute used by Live Story's embed codes assumes as values the configured country codes. However, those depend on the site locale used by the current visitor, it cannot be automatically configured by this cartridge.

For this purpose, the content assets created by Live Story contain a placeholder to help change the attribute value in the template code: `data-store="STORE_ID"`, where `STORE_ID` is the actual placeholder to be replaced.

A simple implementation, considering the Live Story's content assets template code, can be done by changing the line `<isprint value="{contentAsset.livestory.body}" encoding="off" />` with the following `isscript` block.

```
<isscript>
var Locale = require('dw/util/Locale');
var localeId = request.locale;
var currentLocale = Locale.getLocale(localeId);
var country = currentLocale.country.toLowerCase();
out.print(
    pdict.content.livestory.body.replace('STORE_ID', country)
);
</isscript>
```

In the case of custom environments, Live Story will provide advice regarding how to efficiently implement the shoppable features. Please, remember that this aspect isn't compulsory for the cartridge to be fully operative, but however, can be useful to take advantage of all the features available in Live Story.

Upgrade from previous releases

Upgrade from 19.1.0

1. Copy in your previous installation the following directories:
 - `controllers/`
 - `scripts/`
2. Define the JavaScript variable `LS_CONTROLLER_URL` inside the `<head />` section of your page templates. For a reference, consider the following example files in the 20.1.0 package:

- **SiteGenesis:**
templates/default/components/header/livestoryHead.isml
 - **SFRA:** templates/default/common/livestoryHead.isml
3. Apply changes to the Content Asset's rendering files, comparing the actual code with this reference:
- **SiteGenesis:**
templates/default/content/content/htmlcontentasset.isml
 - **SFRA:**
 - i. models/content.js
 - ii. templates/default/content/contentAsset.ism
4. Import the `metadata` package and set the Live Story options in the *Site Preferences* section, as described in *Implementation Guide, Configuration* section.

External Interfaces

The Live Story integration does not use nor expose an external interface such as web services or HTTP clients.

Firewall Requirements

This component does not require any modification to Commerce Cloud firewall.

Testing

For testing the cartridge code please follow the dedicated test guide available in this cartridge package.

Operations, Maintenance

Data Storage

The only data stored within Commerce Cloud is the created content asset themselves. With Live Story is not possible to create other custom objects nor the integration makes use of any jobs.

In Live Story backend a copy of every created content asset is saved within the layout data and it persists as long as the association between the layout itself and the content asset has been deleted.

Availability

Live Story's infrastructure is based on *Amazon Web Services* platform, that guarantees one of the best availability rate in the cloud market. Services, instances and tasks in Live Story are designed to be *fault tolerant*, redundant and scalable, to maintain and high standard in terms of performance and reliability.

The error rate in API responses is under 0.01%, considering an average number of daily requests of 7 million, around five thousand per minute.

In case of failure of the backend service that serves content to the web, the layout on the page just collapses, avoiding to leave blank spaces or empty frames. Also, no Commerce Cloud feature would be blocked in case of Live Story failure, both in StoreFront and Business Manager.

There is a Live Story's API endpoint that can be used as a ping health check and it is available at the address <https://api.livestory.io/front/health>. It replies with a 200 HTTP response code in case of normal activity and it fails to answer in case of failure.

If a customer encounters an issue with Live Story operativity, it can get in touch with the Support Team, as described in the *Support* section of this documentation.

Support

In case of detected anomalies or suggested improvements, a user can provide its feedback to Live Story's Support Team with an e-mail message to address support@livestory.nyc.

User Guide

Roles, Responsibilities

There are not any recurring tasks that need to be fulfilled by a customer. There is just an on-demand workflow when a Live Story layout needs to be saved on Commerce Cloud. A Live Story Console account of role *administrator* is needed, as well as a Business Manager account capable of managing contents. It is an easy workflow that can be handled by a single person.

Business Manager

The integration does not add any new business manager modules nor dedicated UI on Commerce Cloud. Following, a screenshot of the editing page of a Content Asset that shows the fields populated by Live Story. It has been created a custom attribute group thinking about a news section created with Live Story. In addition to the body and the thumbnail fields, there is a text block for a title and another for an abstract.

Live Story	
Live Story - Thumbnail:	https://mediacdnlivestory.io/v1/demo/wall-covers/orig/58b7fe858c94ec00070b8412_1550591835972.jpg
Live Story - Title:	test title
Live Story - Abstract:	test abstract
Live Story - Body:	<pre><div id="ls-58b7fe858c94ec00070b8412" data-id="58b7fe858c94ec00070b8412" data-store="STORE_ID"><div class="fpls-wall-grid-default fpls-lang-default fpls-58b7fe858c94ec00070b8412"><div class="fpls-wall-content"> <div class="fpls-feed-container" style="max-width: 100%; margin-left: auto; margin-right: auto; position: relative; height: 1066.625px;"><script type="text/javascript" id="fpls_cache_58b7fe858c94ec00070b8412">window.fpls_cache_58b7fe858c94ec00070b8412 = "835abd0a76c4b0170183b206935ab40039ec4a330025c5fa8a5aa1121dd65f3c12c742df9a5a9d6eba8f285000257abc785a9cf2705ee75c00112e8f8dc5a9ceb225ee75c00112e83065a9c55645e e75c00112d6f1c5a93346174040726b30542005a8897054d1af000435654a95a8895f14d1af000435652665a7de5b64999420011905a025a58daa49a56c70e2bedc0f859f71cf93bc727001160416 859bc3e6ab7279600256fc82259b81c30d0448a00071e409458ebf99d4f659b00c5c3b68d58dbf06a8f73d2000729b29058d54fa9ec7de900070d854058d3a05eec7de9000709964f58d15ea23e0 12c0007ec55b0577d1bf4c87cbbec96a0f720577d1bf4c87cbbec96a0f722577d1bf4c87cbbec96a0f726577d1bf4c87cbbec96a0f728"</script><div class="fpls-masonry-size" style="width: 33.333333333333336%;"></div><div class="fpls-box fpls-box-photo fpls-square fpls-portrait" data-timestamp="2018-03-29T15:44:23.000Z" style="width: 66.66666666666667%; position:</pre>

Storefront Functionality

What the end user sees on the page is a layout from Live Story, perfectly integrated into sizes and styles. An overview of examples and covered used cases on layouts can be found on the company website, visiting the address <https://www.livestory.nyc/trusted-by/>.

Known Issues

The following list represents the known issues not attributable to Live Story integration. The Live Story Tech Team is always in touch with Salesforce Commerce Cloud support to find possible solutions.

- The integration cannot read or write content attributes owned by a folder's attribute group, tickets W-5394463 and W-5394463.

Release History

<i>Version</i>	<i>Date</i>	<i>Changes</i>
19.1.0	Mon, 11th March 2019	Initial release
20.1.0	Tue, 1st December 2020	Added link parser and shopping-data documentation