Cumulus DAM connector

<table>
<thead>
<tr>
<th>Download</th>
<th>canto-cumulus-integration.jar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edition</td>
<td>EE Pro, Cloud</td>
</tr>
<tr>
<td>License</td>
<td>MLA</td>
</tr>
<tr>
<td>Issues</td>
<td>MGNLCLS</td>
</tr>
<tr>
<td>Maven site</td>
<td>Canto Cumulus</td>
</tr>
<tr>
<td>Latest version</td>
<td>1.3</td>
</tr>
</tbody>
</table>

Related:
- Magnolia Cumulus Conn. examples (Git repo)
- Digital Asset Management

Magnolia’s Cumulus DAM Connector allows users to connect a Canto Cumulus DAM to Magnolia. Cumulus is a popular, enterprise-grade digital asset management platform that offers digital rights management and many back-end connections, including Adobe Photoshop and InDesign and more. Visit Canto’s Cumulus to learn more.

You can use Cumulus assets in a Magnolia website like you would use native Magnolia content. The Cumulus DAM Connector installs the Cumulus Asset app that allows you to browse assets in the remote (Cumulus) DAM.

The Magnolia Cumulus DAM connector module comes with a working configuration pointing to a sandbox account provided by Canto cumulus to try out different assets with your Magnolia installation.

- Installing
- Uninstalling
- Configuring connections
- Caching and performance
- How it works
  - Displaying Cumulus assets with a Magnolia content app
  - Connecting to Cumulus DAM using REST API
  - Based on Magnolia dam-external-app
- Cumulus Assets app
  - Publishing and unpublishing actions
- Using Cumulus assets in templates
  - Example 1 - simple component with an image
  - Example 2 - extending the component textImage from MTK
  - Get the examples from git
- Resources

Installing

The Cumulus DAM module is not bundled with Magnolia. You have to add the modules to your webapps yourself with Maven or with the jar files.

Maven is the easiest way to install the module. Add the following dependencies to your bundle:

```xml
<dependency>
  <groupId>info.magnolia.cumulus</groupId>
  <artifactId>canto-cumulus-integration</artifactId>
  <version>1.3</version>
</dependency>
```

Pre-built jars are also available for download. See Installing a module for help.

- canto-cumulus-integration.jar
- magnolia-dam-external-app.jar
- magnolia-module-observation.jar
Uninstalling

Before uninstalling the module, remove the `/server/filters/cumulus` filter node.

Configuring connections

The module `canto-cumulus-integration` comes with a working configuration pointing to a sandbox account provided by Canto cumulus. To connect with your own account, change the configuration of the module in the Configuration app.

Add the location of your server in both locations:

- `/modules/canto-cumulus-integration/rest-client/cumulus/baseUrl`
- `/modules/canto-cumulus-integration/config/`

Caching and performance

Magnolia automatically caches all REST calls made to Cumulus if the response calls contain valid cache expire headers with an appropriate time interval describing the response as cacheable. All responses without cache headers are assumed to be non-cacheable.

You can configure a custom cache policy in addition to the code provided by the module. Include an extra cache filter in the REST client configuration. One way to improve performance is to implement a custom cache configuration that stores the details of calls in distributed memory shared by all public nodes. This can help in environments where Magnolia public instances perform a large number of calls.

Rendered images and their corresponding variations are served directly from the Cumulus cloud to the users browsing the website. Direct serving ensures that Cumulus can substitute images in case of DRM restrictions or in case the copyright for a rendered asset or its variation has expired.

How it works

As the owner of a Canto Cumulus DAM system, you administer assets on Cumulus with tools they provide. The Magnolia Cumulus DAM connector enables you to seamlessly integrate Cumulus assets into web pages served by Magnolia.

Displaying Cumulus assets with a Magnolia content app

The connector module provides a content app named `cumulus-app` that enables you to browse Cumulus assets from the Magnolia interface. For example, you could use the content app in a Choose dialog to select an asset in a component template. You can see this put into practice below in Example 1 - simple component with an image.

Connecting to Cumulus DAM using REST API

All communication between Magnolia and Canto Cumulus passes through the REST API provided by Cumulus (using the Magnolia REST client module). All calls dealing with Cumulus are made over an authenticated connection using the technical account configured for access between Magnolia and Cumulus. Magnolia does not access any non-public or deprecated API calls of Cumulus.

Based on Magnolia dam-external-app

The Magnolia Cumulus DAM connector is based on the External DAM app module, a Magnolia DAM submodule.

Cumulus Assets app

The Cumulus DAM Connector installs the Cumulus Assets app that allows you to browse all assets in the remote DAM. Such assets can be images, videos or documents stored by other users.

The asset information is read-only because the product update API is not exposed by Cumulus DAM Connector. Assets cannot be updated in Magnolia. The only exception is that you can set asset flags to public or not public by publishing or unpublishing assets.

Open the app using the Find Bar or the app launcher to the right of the Find Bar.
Publishing and unpublishing actions

You can publish or unpublish assets as you do with other content. Only published assets are rendered on public instances.

The publish and unpublish actions carried out in Magnolia set a flag in the Cumulus Canto DAM through a REST call.

Using Cumulus assets in templates

Using assets from Cumulus works in the same way as using assets from the Magnolia DAM. You can use `damfn` templating functions to render the asset. If you are new to templating and assets, read How to work with images using damfn.

Note that Magnolia cannot create renditions for assets on the cumulus DAM. If you need images with different sizes, you have to organize them accordingly in the Cumulus DAM.

When you configure a dialog definition to select Cumulus assets, for the link field you must:

- Set the `identifierToPathConverter` property to the `AssetCompositeIdKeyTranslator` class.
- Set the `appName` to `cumulus-app`.

Example 1 - simple component with an image

In this example you create a component template with a single field for an asset. The code is very similar to what is explained on the page How to work with images using damfn; the only difference when compared is in the dialog where you define the asset link field.

Dialog
/cumulus-templates-examples/dialogs/components/cumulusSimpleImage.yaml

```yaml
form:
  tabs:
    - name: tabImage
      fields:
        - name: image
          class: info.magnolia.dam.external.app.field.definition.AssetLinkFieldDefinition
          appName: cumulus-app
          identifierToPathConverter:
            class: info.magnolia.dam.external.app.field.AssetCompositeIdKeyTranslator
          contentPreviewDefinition:
            contentPreviewClass: info.magnolia.dam.app.ui.field.DamFilePreviewComponent
actions:
  commit:
    class: info.magnolia.ui.admincentral.dialog.action.SaveDialogActionDefinition
  cancel:
    class: info.magnolia.ui.admincentral.dialog.action.CancelDialogActionDefinition
```

- Line 7: `appName` to choose the asset must be `cumulus-app`
- Line 9: `class` property for the `identifierToPathConverter` must be `info.magnolia.dam.external.app.field.AssetCompositeIdKeyTranslator`
- There is no need to specify the property `targetWorkspace`

**Template definition**

/cumulus-templates-examples/templates/components/cumulusSimpleImage.yaml

```yaml
title: cumulusSimpleImage
renderType: freemarker
templateScript: /cumulus-templates-examples/templates/components/cumulusSimpleImage.ftl
dialog: cumulus-templates-examples:components/cumulusSimpleImage
```

**Template script**

/cumulus-templates-examples/templates/components/cumulusSimpleImage.ftl

```html
<div class="cumulusSimpleImage">
    [%assign imgItemKey = content.image!%]
    [%if imgItemKey??%]
        [%assign imgRef = damfn.getAssetLink(imgItemKey)!%]
        [%if imgRef??%]
            <img src="${imgRef}"
               [/%if]
        [/%if]
    [/%if]
</div>
```

**Example 2 - extending the component **textImage** from MTK**

In this example we reuse the component **textImage** which is provided by the **MTK module**. The template script from the MTK component can be used as it is, but on the dialog, we have to change a few bits. To achieve this we use **inherit** and override the dialog from mtk, and the template definition must point to our new dialog.

**Dialog**
Note that the dialog above inherits all the properties from the `mtk:components/textImage` dialog defined in the MTK module, but overrides the `image` field on the first tab. Read Reusing configuration or YAML inherit and include for further information.

**Reference a field by its name**

Instead of defining the type of a field using the `class` property, you can reference the type using the `fieldType` property, which takes the field's short and easy-to-remember alias name as the value.

For example, use `fieldType: text` instead of `class: info.magnolia.ui.form.field.definition.TextFieldDefinition`. To view the field alias names, look in the Definitions app. See also Referencing fields.

**Template definition**

```yaml
cumulus-templates-examples/templates/components/cumulusTextImage.yaml

# Reference a field by its name

Instead of defining the type of a field using the `class` property, you can reference the type using the `fieldType` property, which takes the field's short and easy-to-remember alias name as the value.

For example, use `fieldType: text` instead of `class: info.magnolia.ui.form.field.definition.TextFieldDefinition`. To view the field alias names, look in the Definitions app. See also Referencing fields.
```

**Get the examples from git**

You can clone the examples from our git repository. The repository is a light module. In addition to the two components, it contains a page template with an area where you can use the two components.

Clone URL: https://git.magnolia-cms.com/scm/documentation/cumulus-templates-examples.git

**Resources**

- Canto Cumulus home page
- Magnolia Cumulus Connector Git repository
- Magnolia Cumulus Connector JIRA
- Cumulus templating examples