

CANTO DAM CONNECTOR FOR MAGNOLIA CMS

Version: 1.3.5

Magnolia: 6.4.x

DAM: 4.0.x

Java: 17+

TABLE OF CONTENTS

1. Overview
2. Installation
3. Configuration
4. REST Client Configuration
5. DAM App Integration
6. cantoLinkField — Canto-exclusive input field
7. cantoCropField — Image cropping for Canto assets
8. Usage in Templates
9. MDC Integration (Media Delivery Cloud)
10. Post-Installation Tests
11. Troubleshooting
12. Release Notes



OVERVIEW

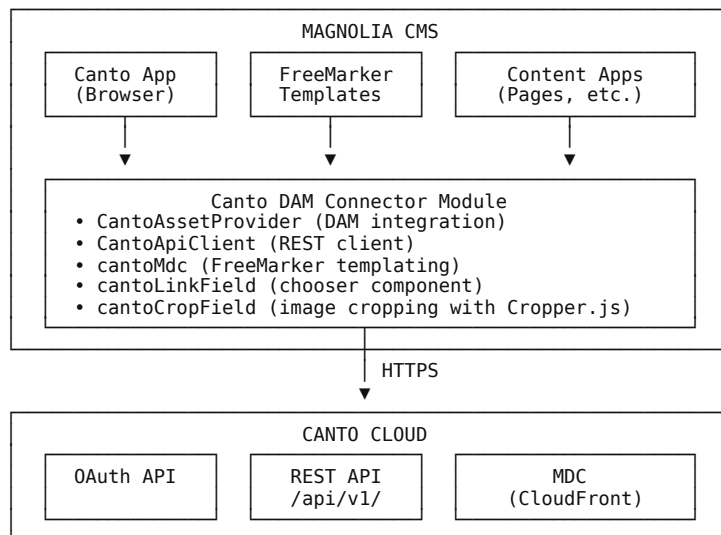
The Canto DAM Connector seamlessly integrates the Canto Digital Asset Management system into Magnolia CMS.

KEY FEATURES

Feature	Description
Asset Browser	Dedicated Magnolia app for browsing Canto
DAM App Integration	Canto as a separate tab in the standard DAM app (since v1.3.0)
Keyword Search	Full-text search across Canto assets with partial-word matching (since v1.3.1)
Actionbar Actions	"Open in Canto", "Download" and "Asset Details" (since v1.3.2/v1.3.3)
Asset Details Dialog	Metadata display with preview, basic data and Canto metadata (since v1.3.3)
cantoLinkField	Input component for dialogs that offers only Canto assets for selection (since v1.3.4)
cantoCropField	Image cropping for Canto assets using Cropper.js (since v1.3.5)
Custom App Icon	Dedicated SVG icon for the Canto app in the App Launcher (since v1.3.2)
OAuth2 Authentication	Secure connection to the Canto API
MDC Integration	CDN-based image delivery with transformations
Templating	FreeMarker functions for responsive images
REST Client	Declarative YAML configuration for API access
Refresh Button	Refreshes Canto data without a restart



ARCHITECTURE OVERVIEW





INSTALLATION

There are two installation options:

OPTION A: MAVEN DEPENDENCY

If you have access to the Neo Reply Maven repository:

1. Add the repository (if not already present)

```
<repository>
  <id>neoreply</id>
  <url>https://nexus.neoreply.de/repository/maven-releases/</url>
</repository>
```

2. Add the dependency

```
<dependency>
  <groupId>de.neoreply.magnolia.canto</groupId>
  <artifactId>canto-dam-connector</artifactId>
  <version>1.3.5</version>
</dependency>
```

OPTION B: JAR FILE FROM NEO REPLY

If you do not have access to the Maven repository, Neo Reply provides the JAR file directly:

1. Request the JAR file

Contact Neo Reply to obtain the current version of the JAR file:

- File: canto-dam-connector-1.3.5.jar
- Contact: your Neo Reply contact person

2. Add the JAR to your project

Option A: Local Maven repository

```
mvn install:install-file \
  -Dfile=canto-dam-connector-1.3.5.jar \
  -DgroupId=de.neoreply.magnolia.canto \
  -DartifactId=canto-dam-connector \
  -Dversion=1.3.5 \
  -Dpackaging=jar
```

Then in the pom.xml:

```
<dependency>
  <groupId>de.neoreply.magnolia.canto</groupId>
  <artifactId>canto-dam-connector</artifactId>
  <version>1.3.5</version>
</dependency>
```

Option B: Directly in WEB-INF/lib

Copy the JAR file into the WEB-INF/lib directory of your Magnolia webapp.

STARTING MAGNOLIA

Once Magnolia starts, the module is installed automatically:

1. Start Magnolia
2. Log in to AdminCentral
3. The module installs itself automatically on first startup

Verifying the installation

After installation you should see:



- App: "Canto DAM" in the app overview (with its own Canto icon)
- DAM app tab: "Canto" as the second tab in the standard DAM app
- Module: /modules/canto-dam-connector in the JCR
- Provider: canto in /modules/dam/config/providers
- REST clients: cantoAuth and cantoApi (after configuration)



Configuration

Module configuration in the JCR

Path: /modules/canto-dam-connector/config

Property	Type	Description	Example
tenant	String	Canto tenant (subdomain)	mycompany
appId	String	OAuth app ID	abc123def456...
appSecret	String	OAuth app secret	secret789xyz...
userId	String	User for API access	canto.api@company.com
oauthBaseUrl	String	OAuth endpoint	https://oauth.canto.de/oauth/api/oauth2
connectionTimeoutSeconds	int	Connection timeout	30
readTimeoutSeconds	int	Read timeout	60
maxRetries	int	Max. retry attempts	3

Example configuration

```
tenant: mycompany
appId: abc123def456ghi789jkl012mno345pq
appSecret: secret789xyz012abc345def678ghi901jkl234mno567pqr890stu123vwx456
userId: canto.api@company.com
oauthBaseUrl: https://oauth.canto.de/oauth/api/oauth2
connectionTimeoutSeconds: 30
readTimeoutSeconds: 60
maxRetries: 3
```

Note: You can obtain the OAuth credentials (appId, appSecret) from your Canto administrator. They must be created in Canto under "Settings > API".



REST Client Configuration

The module includes preconfigured YAML files for the Magnolia REST client. These enable declarative API usage with automatic token management.

REST client overview

Client	File	Description
cantoAuth	cantoAuth.yaml	OAuth2 token retrieval
cantoApi	cantoApi.yaml	All Canto API endpoints

Setup with the Password Manager

1. Create passwords

Open the Passwords app in Magnolia and create three entries:

Name	Value	Description
canto-app-id	Your app ID	OAuth app ID from Canto
canto-app-secret	Your app secret	OAuth app secret from Canto
canto-user-id	api@example.com	User email for API access

2. Enter the UUIDs in cantoAuth.yaml

After creating the passwords, copy the UUID (not the path!) of each entry.

Replace the placeholders in the cantoAuth.yaml file:

```
body: "app_id={@password:UUID-1}&app_secret={@password:UUID-2}&grant_type=client_credentials&user_id={@password:UUID-3}"
```

3. Adjust the tenant (if necessary)

By default, reply.canto.de is configured in cantoApi.yaml. If you have a different tenant:

```
baseUrl: https://MY-TENANT.canto.de/api/v1
```

Available API endpoints

Navigation

REST call	Path	Description
getTree	/tree	Complete folder/album hierarchy
browseFolderContent	/browse/folder/{folderId}	Subfolders and albums of a folder

Content

REST call	Path	Description
getFolderContent	/folder/{folderId}	Assets in a folder



getAlbumContent	/album/{albumId}	Assets in an album
search	/search	Global asset search

Assets

REST call	Path	Description
getAsset	/ {scheme} / {assetId}	Asset by scheme and ID
getImage	/image/{assetId}	Image asset
getVideo	/video/{assetId}	Video asset
getDocument	/document/{assetId}	Document asset
getAudio	/audio/{assetId}	Audio asset
getPresentation	/presentation/{assetId}	Presentation asset
getOther	/other/{assetId}	Other asset

Token handling

The OAuth2 token is managed automatically:

1. On the first API call, a token is obtained via `cantoAuth.authenticate`
2. The token is cached until `expiresIn` seconds have elapsed
3. When it expires, a new token is obtained automatically
4. On 401 errors, the token is invalidated and requested again



DAM App Integration

Overview (since v1.3.0)

As of version 1.3.0, Canto automatically appears as a separate tab in Magnolia's standard DAM app. Editors find Canto assets right next to the Magnolia assets, without having to open a separate app.

How it works

- Canto tab in the DAM app: After installation, a "Canto" tab appears next to the standard "Assets" tab
- Canto assets only: The Canto tab displays only Canto assets (no Magnolia assets)
- Full asset rendering: The title column with icons, preview images and thumbnail view work just as they do for Magnolia assets
- Tree, list and thumbnail view: All three views are supported
- Keyword search: Canto assets can be searched via the search field (since v1.3.1)
- Actionbar actions: "Refresh", "Open in Canto", "Download" and "Asset Details" in the actionbar (since v1.3.2/v1.3.3)

Actionbar actions

The actionbar contains four Canto-specific actions:

Action	Icon	Description
Refresh	Redo	Reloads the Canto folder tree from the API
Open in Canto	Open New Window	Opens the selected asset/folder/album in the Canto web interface
Download	Download	Downloads the selected asset directly
Asset Details	Info	Shows a dialog with a preview image and all of the asset's metadata (since v1.3.3)

"Open in Canto" generates the appropriate URL depending on the type:

- Folder: `https://{tenant}.canto.de/folder/{id}`
- Album: `https://{tenant}.canto.de/album/{id}`
- Asset: `https://{tenant}.canto.de/allfiles?viewIndex=0&column={scheme}&id={id}&origin=classic`

Asset Details dialog (since v1.3.3)

The "Asset Details" dialog shows all available metadata of a Canto asset in a modal window:

- Preview image: Preview of the asset (centered, max. 300px)
- Basic data: Name, type (scheme), MIME type, file size (human-readable), dimensions, created/modified, Canto ID
- Description & copyright: Only if present
- Canto default metadata: All key-value pairs from the default metadata block (keywords, author, etc.)
- Custom metadata: All project-specific metadata from the metadata block
- Empty or missing fields are hidden automatically

Keyword search (since v1.3.1)

The search field in the content browser enables full-text search across all Canto assets:

- Input: Search is triggered automatically from 3 characters onwards



- Partial-word matching: Automatic wildcard expansion ("doo" finds "door", "outdoor")
- Flat result list: Search results are displayed without folder structure
- Assets only: Folders and albums are filtered out of the search results
- Back to the tree: Clearing the search field restores the normal tree view

Technical details

The integration uses Magnolia's decoration mechanism — a YAML file adds the Canto sub-app to the existing DAM app:

```
canto-dam-connector/decorations/dam/apps/dam.subApps.canto.yaml
```

Key components in the datasource-canto scope:

Component	Function
CantoTreePresenter	Extends AssetTreePresenter for the Canto-filtered tree
CantoListPresenter	Extends AssetListPresenter for the Canto-filtered list
CantoAssetDataProvider	Filters on the Canto provider + keyword search via the Canto Search API
CantoDatasourceDefinition	Custom datasource type (cantoDatasource) with assetProviderId
AssetSelectFieldSupport	Converts between item and string (ItemKey) for link fields (since v1.3.4)



cantoLinkField

Overview (since v1.3.4)

The `cantoLinkField` is an input component for Magnolia dialogs that offers only Canto assets for selection. Unlike the standard `damLinkField` (which shows all DAM providers), the `cantoLinkField` opens its own chooser dialog that contains only the Canto asset tree.

Comparison: `damLinkField` vs. `cantoLinkField`

Property	<code>damLinkField</code>	<code>cantoLinkField</code>
Providers	All DAM providers (JCR + Canto)	Canto only
Chooser	Standard DAM chooser	Dedicated Canto chooser
Stored value	<code>jcr:uuid</code> or <code>canto:scheme:id</code>	Always <code>canto:scheme:id</code>
Template usage	<code>damfn</code> or <code>cantoMdc</code>	<code>cantoMdc</code> (recommended)

Usage in dialog YAML

```
form:
  tabs:
    - name: content
      fields:
        heroImage:
          $type: cantoLinkField
          label: Hero Image (Canto)

        teaserImage:
          $type: cantoLinkField
          label: Teaser Image

        # Standard damLinkField for comparison (shows all providers)
        otherImage:
          $type: damLinkField
          label: Any Image
```

Stored format

The field stores the asset identifier as a string in the JCR:

```
canto:image:r9u9rvii4t2hjem9ovrp kuri7c
```

Format: `canto:{scheme}:{id}`, where:

- `canto` = provider ID
- `scheme` = asset type (image, video, document, audio, presentation, other)
- `id` = Canto asset ID

Resolution in the template

The `cantoMdc` templating functions accept the stored string value directly:



```
[#-- Simple: image with max. 800px --]
[#assign assetKey = content.heroImage!]
[#if assetKey?has_content]
  
[/#if]

[#-- Exact dimensions --]


[#-- WebP with quality --]


[#-- Responsive srcset --]


[#-- Smart cropping --]



[#-- Builder for complex transformations --]
[#assign builder = cantoMdc.builder(assetKey)]
[#if builder??]
  
[/#if]
```

Technical details

The cantoLinkField consists of two parts:

1. CantoLinkFieldDefinition (@FieldType("cantoLinkField")): Extends Magnolia's DamLinkFieldDefinition, sets the chooserId to a dedicated Canto chooser and the datasource to CantoDatasourceDefinition
2. Chooser dialog (canto-dam-connector/dialogs/chooser.yaml): SingleItemWorkbenchChooserDefinition with cantoDatasource; shows only the Canto asset tree in tree view



cantoCropField

Overview (since v1.3.5)

The `cantoCropField` enables image cropping for Canto assets directly in the Magnolia dialog. It uses Cropper.js for a visual cropping dialog and stores the crop parameters as an MDC URL suffix (e.g. -C800x600,100,50) in the JCR. In FreeMarker templates, the crop parameters are appended to the MDC base URL.

Usage in dialog YAML

The `cantoCropField` is placed next to a `cantoLinkField` in the same form:

```
form:
  properties:
    heroImage:
      $type: cantoLinkField
      label: Hero Image
    heroImageCrop:
      $type: cantoCropField
      label: Crop
      linkedField: Hero Image
```

Note: The `linkedField` value must match the label (not the property name) of the `cantoLinkField`, e.g. `linkedField: Hero Image` (not `linkedField: heroImage`). If `linkedField` is omitted, the field automatically detects the first `cantoLinkField` in the form.

Optional configuration parameters

```
heroImageCrop:
  $type: cantoCropField
  label: Crop
  linkedField: Hero Image
  aspectRatio: "16:9"      # Fixed aspect ratio (optional, empty = free)
  minWidth: 800            # Minimum width in pixels (optional, 0 = none)
  minHeight: 450          # Minimum height in pixels (optional, 0 = none)
```

Parameter	Type	Default	Description
<code>linkedField</code>	String	(auto-detect)	Label of the associated <code>cantoLinkField</code>
<code>aspectRatio</code>	String	(free)	Aspect ratio, e.g. "16:9", "4:3", "1:1"
<code>minWidth</code>	int	0	Minimum crop width in pixels
<code>minHeight</code>	int	0	Minimum crop height in pixels

Stored format

The field stores the MDC crop suffix as a string in the JCR:

```
-C1600x900,200,100
```

Format: -C{width}x{height},{x},{y}, where:

- width/height = crop dimensions in original pixels
- x/y = offset from the top-left corner of the image in original pixels

For "No crop", an empty string is stored.

Multiplicator concept

In the dialog, the image is loaded at a reduced size via MDC (-B3000, max. 3000px bounding box). The Cropper.js coordinates are scaled up to the original pixel values using a multiplicator:



```
multiplier = originalWidth / displayWidth  
MDC coordinates = Cropper.js coordinates × multiplier
```

This corresponds to the proven concept from the FirstSpirit Canto module.

Usage in FreeMarker templates

```
[!-- Cropped image with bounding box (recommended) --]  
  
  
[!-- Cropped image without resizing --]  
  
  
[!-- Fallback: if no crop is set, the image is delivered uncropped --]  
[#assign crop = content.heroImageCrop!]  
[#if crop?has_content]  
    
[#else]  
    
[#/if]
```

The `cantoMdc.cropped()` method:

- Appends the crop parameters to the MDC base URL
- Optionally adds a bounding box (-B{size}) after the crop
- Automatically falls back to `cantoMdc.url()` if no crop is set

Technical details

Component	Description
CantoCropFieldDefinition	@FieldType("cantoCropField") — field definition with linkedField, aspectRatio, minWidth/minHeight
CantoCropFieldFactory	AbstractFieldFactory — creates the CantoCropField, injects CantoMdcTemplatingFunctions + AssetProviderRegistry
CantoCropField	CustomField<String> — UI with buttons; traverses the Vaadin component tree to find the LinkField
CropperDialog	Modal window — contains the CropperComponent, info label, Apply/Cancel buttons, minimum-size validation
CropperComponent	AbstractJavaScriptComponent — Vaadin server component; communicates with the JS connector via SharedState
CropperState	JavaScriptComponentState — shared state (imageUrl, originalWidth/Height, aspectRatio, initialCrop)
cropper_connector.js	JS connector — initializes Cropper.js, calculates the multiplier, sends crop coordinates to the server
cropper.min.js	Cropper.js v1.6.2 (MIT license) — bundled offline, no CDN
cropper.min.css	Cropper.js styles — bundled offline



CORS handling: Cropper.js is configured with `checkCrossOrigin: false` and `checkOrientation: false` because the Canto MDC CDN does not send CORS headers. This is safe because only crop coordinates are read from events (no canvas export).

MdcUrlBuilder improvement: The `MdcUrlBuilder` automatically detects and removes existing transformation suffixes (e.g. `/-FJPG`) from the Canto MDC base URL before appending new operations. The new `rawOperation()` method allows inserting arbitrary MDC operation strings.



Usage in Templates

Standard DAM functions (damfn)

The Canto provider integrates fully with the Magnolia DAM API. All standard damfn functions work:

Retrieving an asset from content

```
[!-- Retrieve an asset from a content property --]
[#assign asset = damfn.getAsset(content.image)!]

[#if asset?/]
  
[/#if]
```

Canto-specific properties

```
[!-- Check whether it is a Canto asset --]
[#if asset.class.simpleName == "CantoAsset"]
  <p>Preview URL: ${asset.previewUrl!}</p>
  <p>Download URL: ${asset.downloadUrl!}</p>

  [#if asset.hasMdc()]
    <p>MDC Base URL: ${asset.mdcBaseUrl!}</p>
  [/#if]
[/#if]
```



MDC Integration

What is MDC?

MDC (Media Delivery Cloud) is Canto's CDN solution based on AWS CloudFront:

- Worldwide edge locations for fast delivery
- On-the-fly image transformations
- Automatic caching
- WebP/AVIF support

cantoMdc templating functions

After installation, the `cantoMdc` object is available in FreeMarker templates.

Responsive images (bounding box)

```
[!-- Scale image to max. 800px (preserves aspect ratio) --]


[!-- With format and quality --]

```

Exact dimensions

```

```

Responsive images with srcset

```

```

Image with crop parameters (since v1.3.5)

```
[!-- Cropped image with bounding box --]


[!-- Cropped image without resizing --]

```

Fallback behavior

If MDC is not available:

1. Automatic fallback: `directUrlPreview` is used as the default preview URL
2. Null handling: Missing MDC metadata is correctly treated as null



Post-Installation Tests

Test 1: Verify the module installation

1. Open the Configuration app
2. Navigate to `/modules/canto-dam-connector`
3. Check whether `/modules/canto-dam-connector/config` and `/modules/canto-dam-connector/apps/canto` exist

Test 2: Open the Canto app

1. Click the App Launcher icon (9 dots)
2. Look for "Canto DAM" — the app shows its own Canto SVG icon
3. Open the app

Test 3: Check the Canto tab in the DAM app

1. Open the standard Assets app (DAM)
2. Check whether a "Canto" tab appears next to the "Assets" tab
3. Switch to the Canto tab
4. Check: only Canto folders and assets are displayed

Test 4: Test the keyword search

1. In the Canto tab: enter a search term (at least 3 characters)
2. Check: a flat list of assets appears, partial-word search works
3. Clearing the search field restores the tree view

Test 5: Test the actionbar actions

1. Select an asset
2. "Open in Canto" — the asset opens in a new tab in the Canto web interface
3. "Download" — the asset download starts
4. "Asset Details" — a modal dialog with preview image and metadata opens
5. "Refresh" — the tree is reloaded

Test 6: Test the `cantoLinkField` (since v1.3.4)

1. Create a test dialog with `$type: cantoLinkField`
2. Click "Select" — a chooser containing only Canto assets opens
3. Navigate through the folder structure and select an asset
4. Check: a preview of the asset appears in the field
5. Check: the stored value has the format `canto:scheme:id`
6. Reopen the dialog — the previously selected asset is displayed correctly
7. Clear the field — the value is removed

Test 7: Test the `cantoCropField` (since v1.3.5)

1. Create a test dialog with `cantoLinkField` + `cantoCropField` (with `linkedField` set to the label of the `cantoLinkField`)
2. Select a Canto asset in the `cantoLinkField`



3. Click "Crop" — a modal cropping dialog opens with the image
4. Drag and resize the crop area — the info line shows the current dimensions
5. Click "Apply" — the crop value appears in the field (e.g. -C1600x900,200,100)
6. Reopen the dialog — the previous crop is preselected
7. "Reset" — the crop value is deleted, the display changes to "No crop"
8. Optional: test aspectRatio: "16:9" — the crop box keeps a fixed aspect ratio
9. Optional: test minWidth/minHeight — a crop that is too small shows a warning, "Apply" validates
10. In the FreeMarker template: `cantoMdc.cropped()` generates the correct MDC URL with crop parameters

Test 8: Embed an asset in content

1. Open the Pages app and edit a page with an image component
2. Click "Choose asset" and switch to the "Canto" tab
3. Select an asset



Troubleshooting

Problem: No connection to the Canto API

Symptoms: The Canto app shows no data; log: "Failed to obtain access token"

Solution: Check the OAuth credentials, test whether `https://{tenant}.canto.de` is reachable, check the firewall.

Problem: Canto tab shows two trees

Solution (fixed in v1.3.0): Update to version 1.3.0+

Problem: "null" is displayed as the image URL

Solution (fixed in v1.2.0): Update to version 1.2.0+

Problem: MDC URLs not available

Causes: MDC not licensed, asset not released for MDC, additional field empty.

Problem: cantoMdc not available in the template

Solution: Check whether `/modules/rendering/renderers/freemarker/contextAttributes/cantoMdc` exists, restart Magnolia.

Problem: REST client not available

Solution: Check that `magnolia-rest-client` is on the classpath (groupId: `info.magnolia.restclient`), check the YAML files and the Password Manager UUIDs.

Problem: Search returns no results

Solution: Enter at least 3 characters, check the log for Canto Search API calls, test API reachability.

Problem: cantoLinkField chooser shows an error

Solution: Check whether `SelectFieldSupport` is registered in the `datasource-canto` scope (in `canto-dam-connector.xml`). Update to version 1.3.4+.

Problem: cantoCropField — "Please select a Canto asset first"

Causes: No asset selected in the linked `cantoLinkField`, or `linkedField` points to an incorrect label.

Solution: Check that: 1. the `cantoLinkField` contains a selected asset, 2. `linkedField` uses the label text (not the property name) of the `cantoLinkField`, 3. alternatively: omit `linkedField` for auto-detection.

Problem: cantoCropField — image does not load in the cropper dialog

Causes: CORS issues with the Canto MDC CDN.

Solution: Update to version 1.3.5+ (includes a CORS-safe Cropper.js configuration with `checkCrossOrigin: false` and `checkOrientation: false`).



Release Notes

v1.3.5 — cantoCropField Image Cropper (current)

Visual image cropping for Canto assets with Cropper.js directly in the Magnolia dialog.

- `cantoCropField`: New field type (\$type: `cantoCropField`) for image cropping alongside a `cantoLinkField`
- Cropper.js dialog: Visual cropping with drag & resize, aspect-ratio constraint and minimum-size validation
- MDC crop parameters: Stores the MDC suffix (e.g. -C1600x900,200,100) in the JCR
- Multiplicator concept: Scales display coordinates up to original pixel coordinates (as in FirstSpirit)
- `cantoMdc.cropped()`: New FreeMarker method for cropped images with an optional bounding box
- `MdcUrlBuilder.rawOperation()`: New method for inserting arbitrary MDC operation strings
- `MdcUrlBuilder` URL fix: Automatic removal of default transformation suffixes from Canto MDC base URLs
- CORS-safe: Cropper.js configured without CORS-dependent features
- Cropper.js v1.6.2: Bundled offline (MIT license), no CDN access at runtime

v1.3.4 — cantoLinkField Chooser Component

Dedicated input component for template dialogs with Canto-exclusive asset selection.

- `cantoLinkField`: New field type (\$type: `cantoLinkField`) for Magnolia dialogs
- Dedicated chooser dialog: Opens a Canto-exclusive asset browser
- Seamless template integration: The stored value works directly with the `cantoMdc` functions
- `SelectFieldSupport`: Item-to-string conversion in the `datasource-canto` scope

v1.3.3 — Asset Details Dialog

Metadata display for Canto assets directly from Magnolia.

- "Asset Details" action: New actionbar button shows a modal dialog with metadata
- Preview image: Preview of the asset in the dialog (centered, max. 300px)
- Basic data: Name, type, MIME type, file size, dimensions, created/modified, ID
- Canto default and custom metadata: All key-value pairs displayed dynamically

v1.3.2 — Actionbar Actions & Custom App Icon

Context actions for Canto assets and a dedicated app icon.

- "Open in Canto" action: Opens the asset/album/folder in the Canto web interface
- "Download" action: Downloads the asset directly from Canto
- Custom SVG app icon: Dedicated Canto icon in the App Launcher
- Correct Canto URLs and Magnolia icons

v1.3.1 — Keyword Search

Full-text search across Canto assets directly in the Magnolia content browser.

- Keyword search: The search field searches all Canto assets
- Partial-word matching: Automatic wildcard expansion
- Flat result list: Search results without folder structure
- Assets only: Folders and albums are filtered out

v1.3.0 — DAM App Integration

Canto as a fully fledged tab in Magnolia's standard DAM app.



- Canto tab in the DAM app: Added as a separate tab via the decoration mechanism
- Full asset rendering: Title column, preview images, thumbnail view
- Filtered view: Canto assets only
- Jakarta migration: Switch to jakarta.inject for Magnolia 6.3.x

v1.2.0 — MDC Integration & REST Client

Canto Media Delivery Cloud and declarative REST client configuration.

- MDC integration: cantoMdc FreeMarker object with image transformations
- REST client YAML: Declarative API configuration with OAuth2 token management
- Password Manager: Secure credential storage
- Refresh button and null fix

v1.1.0 — Folder Structure & Thumbnails

- Folder/album structure: Correct Canto hierarchy
- Thumbnail display: Asset preview images
- Asset referencing: Canto assets can be stored in Magnolia content

v1.0.0 — Initial Release

- Asset provider: Canto registered as a DAM provider
- OAuth2 authentication: Secure API connection
- Browser app: Dedicated Magnolia app for Canto assets

Documentation prepared for Canto DAM Connector v1.3.5