d.velop

DMS-App-API

Table of Contents

1. DMS-App-API

1.1. Introduction

This topic provides you with general information about the DMSApp API, its scope of functions and useful information to know when programming functions and customizations.

1.1.1. About the DMSApp API

In this documentation, you can learn how to use the programming interface of DMSApp for your own developments.

This documentation is designed for developers of applications that are to be run in the d.ecs architecture system environment. DMSApp is a core component in the d.ecs architecture system environment and provides functions related to documents and dossiers within the framework of a d.3ecm system.

JSON objects, URLs, and query parameters can be enhanced at any time while the API is still valid. Only the extension points, properties, objects and parameters for API programming described in this documentation are released. Each chapter indicates whether the URL for retrieving the HTML page, the JSON representation or the extension point is released. The HTML page is used exclusively for UI integration, must not be evaluated with programs and may be changed at any time without prior notification. Any properties that are not described in this documentation are not permitted for use.

This documentation is available to development partners of d.velop AG online on the Service Portal. Passing this documentation or parts of it on to third parties is not permitted. For requests as part of the development partnership, only the online documentation applies.

Please note that your software accesses data stored and configured in d.3ecm by your customers via this interface and also affects the procedures in the d.3ecm system. Please handle it with care and make sure that your application is part of an existing collaboration of several applications. The improper use of this interface may lead to changed application procedures and the loss of data.

1.1.2. Scope of functions of DMSApp

DMS app is the core HTTP interface to essential functionality related to documents and dossiers in the d.3ecm environment. You can use the DMS app, for example, to create mappings, create, edit and update documents. You can also navigate through dossier structures and display detailed information using the DMS app. In addition, the DMS API offers lots of extension points for your own features and functions.

1.1.3. Prerequisites

This documentation is designed for administrators of a d.3ecm system and for developers creating apps and extensions.

You should be familiar with the following subjects to use the API:

- You should have a good knowledge of the d.3ecm architecture.
- You should be familiar with the authentication using the d.ecs identity provider API in the d.3ecm context.
- You should have knowledge in the development of web-based apps and be confident in dealing with the following detailed topics:
  - Hypertext Transfer Protocol (HTTP) (RFC 7230)
  - RESTful HTTP interfaces
  - JavaScript Object Notation JSON) (RFC 7159)
  - Hypertext Application Language in connection with the JavaScript Object Notation (HAL+JSON)
  - UriTemplate (RFC 6570)

- If you want to provide your own user interface, you should be familiar with HTML, CSS and responsive web design. To facilitate a consistent look & feel, it is recommended to know how to use the framework Material Design. For more recommendations about how to design see the Developer Portal.
- Additionally, knowledge in administering a web server is a benefit.

1.2. Notes about support for API programming

Please note that your software accesses stored and configured data in d.3ecm owned by your customers via this interface and also affects the procedures in the d.3ecm system. Please handle it with care and make sure that your application is part of an existing collaboration of several applications. The improper use of this interface can lead to changed application procedures and the loss of data.

Software developments with this programming interface are individual developments. The program code generated by you is not subject to the maintenance and support conditions of the products of d.velop AG. Our support will be pleased to help you, but your requests are subject to additional charges, if it is not lead back to an error in our products.

For any questions about the prerequisites and the software development with d.3ecm, please contact the Technology Partners Management of d.velop AG.

1.3. Technical background information

It is recommended to be acquainted with the system landscape, the interoperability and the architecture of d.3ecm.

In addition, the authentication mechanism in the d.3ecm system landscape is of high importance. You can find general information about authentication under Basic knowledge about authentication.

You can find general information about using the DMSApp mapping feature under Basic information about mapping.

You can find general information about writing access to the DMSApp under Basic information about write access.

1.3.1. Basic information about mapping

The mapping feature in DMSApp lets you establish a connection between any third-party application, such as an e-mail application or ERP system, and a d.3 repository. This mapping feature is the basis for creating extensions and functions that make it easier for you to program functions, e.g. saving elements that come from another ERP system.

For example, if you would like to store your e-mails in a d.3 repository by default, it is useful to directly map the item "E-mail" to a d.3 category and the related d.3 properties. You can save your users from manually mapping properties and categories if you define the mapping as an administrator. Thus, your source system is the e-mail application, the destination system is a d.3 repository. By default, any e-mail has specific properties, among them are, for example, the sender and the recipient as well as the subject of an e-mail. You can use these typical properties for an e-mail to map them to a d.3 category and the d.3 properties. This mapping automatically maps the e-mail to the correct category, for example. The typical e-mail properties are then automatically written into d.3 properties.

To use the mapping feature effectively, you can make the data model of a third-party application available as a source to connect it to the data model of the d.3ecm target (DMSApp). The data model description is designated as the source.

For instance, you may want to store your data from an e-mail application (e-mails, attachments) in a d.3 repository and save your users from manually mapping d.3 categories and d.3 properties. In this case, the e-mail application is your source system. The e-mail is a source of this source system, which has certain categories and properties by default. These typical categories and properties for the source must be “translated” into the d.3ecm data model. This method is the only way for you to ensure that the fields are mapped correctly by default when you save the items from the e-mail application.

An e-mail application provides you with a variety of metadata. This may include the sender, recipient and subject line, for example. You can connect this source metadata to the metadata in a d.3 repository, where the items from the source are saved. For example, in a d.3 repository, you can map an e-mail to the document type **Correspondence** and the subject (source property **Mail_Subject**) to the d.3 property **Subject**.

Understanding the relationship between the possible categories and properties of a source and the categories and properties in d.3ecm is important for mapping.

1.3.2. Basic knowledge about authentication

The app d.ecs identity provider is responsible for authentication in the architecture of a d.3ecm environment. Almost all the REST resources of the DMSApp must be called with authentication. In the following descriptions of the API functions, you are informed if you can call a REST resource anonymously.

For information about how the identity provider app works, see the identity provider API documentation.

1.3.3. Basic information about write access

When calling the API via the HTTP verbs **POST**, **PUT**, **DELETE** and **PATCH**, you must specify the **Origin** header. The header value must correspond to the calling URL without path. The header is not required for read access (e.g. **GET**).

It is necessary to specify a value for **Origin** in order to prevent CSRF (Cross-Site Request Forgery) attacks.

Request

```
PUT /dms/sampleuri
Origin: https://samplehost
```

If the header is missing, the DMSApp will reply with **HTTP 403 Forbidden**.

1.4. Using the API functions

In the following chapters, you can learn more about the various options for using the DMSApp user interfaces to meet your requirements.

- General information about API functions (DMSApp)
- Calling the web interface with a URL (DMSApp)
- Connecting external systems (DMSApp)
- Adding your own functions (DMSApp)

1.4.1. General information about API functions (DMSApp)

In this chapter, you can find all the topics that apply to all types of use of the API functions in DMSApp.

Determining a repository

Released: JSON representation

To implement your own functions, you always need the ID of the repository. With DMSApp, you have the option of accessing different repositories if you have configured multiple repositories in your company or organization.

If, for example, you want to start a search, you must first select a repository. To specify the repository, execute a **HTTP GET** request for the REST resource **/dms**.

The repository ID is determined in two steps:

- Determining the link relation for retrieving the list of repositories
- Retrieving the list of repositories

Determining the link relation for retrieving the list of repositories

The URL for a repository is available as a link relation in the response to the **HTTP GET** request.

Request

```
GET /dms
Accept: application/hal+json
```

Response

```
{
        _links: {
                repo: {
                        href: "/dms/r/{repositoryid}",
                        templated: true
                }
        }
}
```

Retrieving the list of repositories

To call repository-specific functions, you require the repository ID.

Replace the **{repositoryid}** placeholder in the URL **"/dms/r/{repositoryid}"** with the repository ID. If you do not know the repository ID, open the URL **/dms/r** as follows:

Request

```
GET /dms/r
Accept: application/hal+json
```

In the response, you receive an array of repositories in which the repository ID is listed as the **id** property and the repository's display name is listed as the **name** property.

Response

```
{
        repositories: [
                {
                        id: "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27",
```

```
                        name: "Contoso (A)"
                },
                ...
        ]
}
```

If you already know the repository ID, then you can replace the **{repositoryid}** placeholder in the URL **/dms/r/{repositoryid}** with the repository ID. When you open the URL with the repository ID, you receive the following result:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object in the response is the same object as the object in the list of JSON objects from the request for the URL **/dms/r**.

Response

```
{
        id: "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27",
        name: "Contoso (A)"
}
```

Overview of formats for errors

Released: JSON representation

In this chapter, you will learn more about the basics of error handling and the format in which errors are returned.

Depending on the result of the processing of an **HTTP** request, the request is answered with different **HTTP** status codes. Descriptive information can be returned as JSON as an option.

Example for a response upon a failed request:

Response

```
HTTP/1.1 400 BadRequest

{
    "reason": "10019: Missing value for a mandatory property.",
    "severity": 1,
}
```

Description of parameters for response upon a failed request:

Property	Description
reason	An optional short description of why the error has occurred. This text is used as the error message's title.
hint	An optional message for the user containing troubleshooting tips.
details	Optional detailed information about the error.

severity	Optional error severity. Possible values are:
	`Success = 0,`
	`Information = 1,`
	`Warning = 2,`
	`Error = 3`

1.4.2. Calling the web interface with a URL (DMSApp)

In this section, you can learn how to reach the HTML user interfaces of the DMSApp using URLs. You can also use these URLs to integrate the HTML user interfaces into your own application.

Displaying the storage dialog

Released: HTML page

The storage dialog for the **Storage** feature allows you to save and update DMS objects.

After you open the storage dialog in the browser and enter values, you are shown the link for calling the displayed dialog, including your entered values, in the URL. You can use this link as a bookmark in the browser, for example, so that you can always display the storage dialog with the entered values. This chapter explains how this link is structured.

If you want to connect an external system, you can find more information about storing new DMS objects using the storage dialog in Storing new DMS objects with user interaction. For information on how to prepare and display the storage dialog for updating from an external system, see chapter Updating DMS objects with user interaction.

Determining the link relation for displaying the storage dialog

If you call the root resource **/dms**, you receive in response a JSON object with the link relation **new**, which you can use to display the storage dialog, as the response.

The request for the root resource **/dms**:

Request

```
GET /dms
Accept: application/hal+json
```

The JSON object of the root resource contains the link relation **new**:

Response

```
{
        _links: {
                new: {
                        href: "/dms/new/"
                }
        }
}
```

Specifying response control parameters

You control the behavior of the storage dialog using the following parameters. You must encode the URL parameters (e.g. spaces in **%20**). The length of the encoded query parameter is limited to 2000 characters.

Parameter	Description
type	Specifies the category (storage type) in the storage dialog. The only possible value is **dossier**. **type** opens the storage dialog for creating dossiers.

Parameter	Description
repositoryid	Defines the repository ID.
objectdefinitionid	Specifies the category to be selected in the storage dialog. Specify the ID of a category. If you do not make a specification, no category is selected. Examples (not encoded): • **objectdefinitionid=RECH**: Selects the category "Invoice" (**"RECH"**) for storage.
properties	A JSON object that you can use to set the following properties: • Document number: **property_document_number** • Variant number: **property_variant_number** • Document status: **property_state** (the possible values are **Be** for Processing (Bearbeitung), **Pr** for Verification (Prüfung), **Fr** for Release (Freigabe) and **AR** for Archive (Archiv). • Editor: **property_editor** (d.3 user ID) • Remark 1: **property_remark1** • Remark 2: **property_remark2** • Remark 3: **property_remark3** • Remark 4: **property_remark4** • Color code: **property_colorcode** (is an integer value between 1 and 24 that corresponds to the number of the color code you want to assign). You can also set values for advanced properties. The ID of the property corresponds to the identifier (RID) of the property. You can find the ID in the URL in the address bar of your browser. After you open the storage dialog in the browser and enter values, you are shown the link for calling the displayed dialog, including your entered values, in the URL. You can set values for advanced properties as follows. • **properties={"227"=["KND001"]}** For multi-values, you must also set the row number: • **properties={"232":{"1":"Name1@contoso.com", "2":"Name2@contoso.com"}}**

Opening the URL for the storage dialog

You have generated a URL for the storage dialog. If you open the URL in the browser, the storage dialog is loaded with the transferred parameters.

Note

Use cases for various calls of the storage dialog:

- **Storage in a category**: Add the part **objectdefinitionid=<category ID>** to the URL.

https://<Base address>/dms/new/?repositoryid=<RepositoryID>&objectdefinitionid=RECH

- **Storage with an alphanumeric property**: Add **properties={"227":"KND001"}** (not encoded) to the **properties** parameter in the URL to propose the property field with the RID 227 (customer number) and the value **"KND001"** as the customer number.

https://<Base address>/dms/new/?repositoryid=<RepositoryID>&objectdefinitionid=RECH&properties=%7B%22227%22%3A%22KND001%22%7D

- **Storage with multiple properties**: Add **properties={"227":"KND001","231":"-100"}** (not encoded) to the **properties** parameter in the URL to propose the property field with the RID 227 (customer number) with the value **"KND001"** and the property field with the RID 231 (invoice amount) with the value -100.

https://<Base address>/dms/new/?repositoryid=<RepositoryID>&objectdefinitionid=RECH&properties=%7B%22227%22%3A%22KND001%22%2C%22231%22%3A%22-100%22%7D

- **Storage with multi-values**: Add **properties={"232":{"1":"Name1@contoso.com", "2":"Name2@contoso.com"}}** (not encoded) to the **properties** parameter in the URL to propose the property field with the RID 232 (e-mail) with the values "**Name1@contoso.com**" and "**Name2@contoso.com**".

https://<Base address>/dms/new/?repositoryid=<RepositoryID>&objectdefinitionid=RECH&properties=%7B%22227%22%3A%22KND001%22%2C%22231%22%3A%22-100%22%2C%22232%22%3A%7B%221%22%3A%22Name1%40contoso.com%22%2C%20%222%22%3A%22Name2%40contoso.com%22%7D%7D

Viewing the details of a DMS object

Released: HTML page

The detail view offers you a number of different perspectives for the requested item, such as **Properties**, **View** and **Notes**.

To find our how to view the detail view for an item, see the Retrieving and viewing the details of a DMS object chapter.

Displaying the preview of a DMS object

Released: HTML page

The **View** perspective offers you a preview for the selected item. You can use the content of the perspective for an item directly without loading the other perspectives.

To display only the preview of a DMS object, you must perform the following steps:

- Determining the URL for a repository
- Determine the link relation for displaying the preview
- Open the URL for displaying the preview

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine the link relation for displaying the preview

You open the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for a repository contains the link relation **dmsobjectpreview** with the placeholder **dmsobjectid**.

Response

```
{
        _links: {
                dmsobjectpreview: {
                        href: "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/o2/{dmsobjectid}/preview{?isReadonly}",
                        templated: true
                }
        },
        id: "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Replace the **{dmsobjectid}** placeholder with the ID of the DMS object that you want to view directly in the preview.

Specifying response control parameters

Parameter	Description
isReadonly	Specifies whether the preview is displayed in read mode (default value: **false**). In read mode, it is also not possible to apply visual annotations to PDF documents, for example. Use the parameter **isReadonly=true** if the preview is displayed in the inner supply (**InnerSupply**) of the operating concept.

Open the URL for displaying the preview

You open the preview as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/P0123456789/preview
Accept: text/html
```

The preview of the DMS object is then loaded as a result.

Searching for DMS objects

You can integrate the HTML pages of the **Search** feature in your own HTML user interface to simplify tasks that recur frequently for users, for example.

You can integrate the following tasks:

- Defining search operations
- Viewing the results of a search operation

Parameters that you can use both for search operations and to display the result list are explained for you in the chapter Defining the parameters for a search operation and a result list.

Defining search operations

Released: HTML page

You can use the **Search** feature to find saved items in the repository, provided that you have the appropriate authorizations. Based on their area of responsibility, users frequently search for the same items, which they can find quickly by using selected properties and categories. You can simplify the search process by providing users with a predefined search operation. In the search operation, you have already defined the values that your users otherwise have to enter repeatedly.

To make a predefined search operation available, transfer the parameters of the URL.

You can call the search operation in two ways:

- Call the search operation without specifying a specific repository. In this case, the last selected repository is used for the search operation. However, the user can change the repository at any time.
- You call the search operation for a specific repository, which means that the user can no longer change the repository.

Searching without specifying a repository

If you want to define the search operation without specifying a repository, proceed as follows:

When you open the URL **/dms**, you are notified of the URL for the search operation, including the available parameters, with the following HTTP response:

Request

```
GET /dms
Accept: application/hal+json
```

The JSON object for the root resource contains the **search** link relation with placeholders for the values used to perform the search for DMS objects.

Response

```
{
        _links: {
                search: {
                        href: "/dms/s/{?
objectdefinitionids,fulltext,properties,showresultlist,repositoryid}"
                        templated: true
                }
        }
}
```

Searching in a predefined repository

If you want to define a search operation for a specific repository, you must first determine the URL for the repository. In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Once you have determined the URL for the repository (for example: **/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27**) and called the URL using **HTTP GET**, you receive the following result, which contains the repository-specific URL for the search operation:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for the repository contains the **search** link relation with placeholders for the values used to perform the search for DMS objects.

Response

```
{
        _links: {
                search: {

href: "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/s/{?
objectdefinitionids,fulltext,properties,showresultlist}"
                        templated: true
                }
        },
        id: "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Specifying response control parameters

You control the system response for searching and displaying using the following parameters:

Parameter	Description
showresultlist	If values are available for a search operation, this specifies whether the search operation is executed directly and the results of the search query are displayed (default value: **false**).
repositoryid	Defines the repository ID.

For descriptions of the **objectdefinitionids**, **fulltext** and **properties** parameters, see the Defining the parameters for a search operation chapter.

> **Note**
> You can also define multiple values for a parameter, provided that facets are displayed for the corresponding property. If no facets for viewing have been defined for the property, the last value is always adopted for the search operation.

Opening the URL for the search operation

Once you have created a URL for the search operation, you can then call the results using the corresponding requests.

Request without a repository ID:

Request

```
GET
/dms/s/?objectdefinitionids=["RECH"]&fulltext=Mustermann&properties={"227":
["KND001"]}&repoid=dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: text/html
```

Request with a repository ID:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/s/?
objectdefinitionids=["RECH"]&fulltext=Mustermann&properties={"227":
```

```
["KND001"]}
Accept: text/html
```

The search operation in the **Search** feature is then loaded with the transferred parameters as a result.

> **Note**
> You must encode the URL parameters (e.g. spaces in **%20**). The length of the encoded query parameter is limited to 2000 characters.
>
> Use cases for various search queries (not encoded):
>
> - **Searching across a category:** Add the part **objectdefinitionids=["<category ID>"]** to the URL.
>
> **https://**<Base address>**/dms/r/**<RepositoryID>/s/?fulltext=5353&objectdefinitionids=["RECH"]
>
> - **Searching across several categories**: Add the part **objectdefinitionids=[**"<category ID>","**<category ID>"**] to the URL.
>
> **https://**<base address>**/dms/r/<RepositoryID>**/s/?fulltext=5353&objectdefinitionids=["RECH","AUFT"]
>
> - **Searching for PDF documents restricted to the file type**: Add the part **properties={"property_filetype":["<File type>"]}** to the URL.
>
> **https://**<base address>**/dms/r/<RepositoryID>**/s/?fulltext=test&properties={"property_filetype":["pdf"]}
>
> - **Searching for an alphanumeric property**: Add **properties={"227":["KND001"]}** (not encoded) to the **properties** parameter in the URL to find the property field with the RID 227 (customer number) and the value **"KND001"** as the customer number.
>
> **https://**<base address>**/dms/r/<RepositoryID>**/s/?objectdefinitionids=["RECH"]&fulltext=&properties={"227":["KND001"]}
>
> - **Searching for multiple properties**: Add **properties={"227":["KND001"],"231":["|-100"]}** (not encoded) to the **properties** parameter in the URL to find the property field with the RID 227 (customer number) with the value **"KND001"** and the property field with the RID 231 (invoice amount) with a value lower than or equal to 100.
>
> **https://**<base address>**/dms/r/<RepositoryID>**/s/?objectdefinitionids=["RECH"]&fulltext=&properties={"227":["KND001"],"231":["|-100"]}

Viewing the results of a search operation

Released: HTML page

When defining search operations, you can specify that only the result list is to be displayed. Your users can then view the results of a search directly and no longer have to specify the individual properties. The result list makes it easy for your users to continue their work. However, the users can no longer change the search criteria that led to the result.

To display only the results of a search operation, you must perform the following steps:

- Determining the URL for a repository
- Determine the link relation for retrieving the results of a search operation
- Specify response control parameters

- Open the URL for the results

Once you have created a mapping for a source, you can also display the results of a search operation in other ways. For additional information see chapter Retrieving and viewing the results of a search operation.

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determining the link relation for retrieving the results of a search operation

You open the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for the repository contains the **searchresult** link relation with placeholders for the values used to perform the search for DMS objects.

Response

```
{
        _links: {
                searchresult: {

href: "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/sr/{?
objectdefinitionids,fulltext,properties,propertysort,ascending,showdetails}"
                        templated: true
                }
        },
        id: "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Specifying response control parameters

Use the following parameters to control the system response when viewing the results:

Parameter	Description
showdetails	If the search operation finds a single document, this parameter specifies whether this document is displayed directly in the detail view (default value: **false**).

Parameter	Description
property-sort	Specifies the ID of the property that is used to sort the result. If no sort property is specified, the default sort order using the **Last modified** sort criterion is applied. • Caption/Title: **property_caption** • Owner: **property_owner** • File type: **property_filetype** • Remark: **property_remark** • Last access: **property_access_date** • Editor: **property_editor** • Document ID: **property_document_id** • Document number: **property_document_number** • File size: **property_size** • Document status: **property_state** • Last modified: **property_last_modified_date** • Last file update: **property_last_alteration_date** • Color code: **property_colorcode** • Category: **property_category** • Created on: **property_creation_date** • Last access: **property_access_date** You can also sort the result list based on an advanced property. The ID of the property corresponds to the ID (RID) of a property that you can determine for each advanced property in d.3 admin.
ascending	Specifies the direction of the sort order. • **ascending=true**: results in an ascending sort order (from small to large (A-Z) and from old to young). • **ascending=false**: results in an descending sort order (from large to small (Z-A) and from young to old). If the **ascending** parameter is not explicitly specified, the ascending sort order is applied. This is not true for the default sort order: If the **Last modified** criteria is used for sorting and the sort order is not specified, the sort order will be descending. In addition to this, dossiers are displayed prior to documents in the result list. Within documents and dossiers, the items are sorted by the sorting criterion.
children_of	Specifies the document ID for which the linked direct child items are searched.

For descriptions of the **objectdefinitionids**, **fulltext** and **properties** parameters, see the Defining the parameters for a search operation chapter.

Opening the URL for the results

Once you have created a URL, you can then open the results as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/sr/?
objectdefinitionids=["RECH"]&fulltext=Mustermann&properties={"227":
["KND001"]}
Accept: text/html
```

The result of the search operation is then loaded.

Note

You must encode the URL parameters (e.g. spaces in **%20**). The length of the encoded query parameter is limited to 2000 characters.

Use cases for various result lists (not encoded):

- **Searching across a category:** Add the part **objectdefinitionids=["<category ID>"]** to the URL.

https://<Base address>**/dms/r/**<RepositoryID>/sr/?fulltext=5353&objectdefinitionids=["RECH"]

- **Searching across several categories**: Add the part **objectdefinitionids=[**"<category ID>","**<category ID>"**] to the URL.

https://<base address>**/dms/r/<RepositoryID>**/sr/?fulltext=5353&objectdefinitionids=["RECH","AUFT"]

- **Searching for PDF documents restricted to the file type**: Add the part **properties={"property_filetype":["<File type>"]}** to the URL.

https://<base address>**/dms/r/<RepositoryID>**/sr/?fulltext=test&properties={"property_filetype":["pdf"]}

- **Searching for an alphanumeric property**: Add **properties={"227":["KND001"]}** (not encoded) to the **properties** parameter in the URL to find the property field with the RID 227 (customer number) and the value **"KND001"** as the customer number.

https://<base address>**/dms/r/<RepositoryID>**/sr/?objectdefinitionids=["RECH"]&fulltext=&properties={"227":["KND001"]}

- **Searching for multiple properties**: Add **properties={"227":["KND001"],"231":["|-100"]}** (not encoded) to the **properties** parameter in the URL to find the property field with the RID 227 (customer number) with the value **"KND001"** and the property field with the RID 231 (invoice amount) with a value lower than or equal to 100.

https://<base address>**/dms/r/<RepositoryID>**/sr/?objectdefinitionids=["RECH"]&fulltext=&properties={"227":["KND001"],"231":["|-100"]}

- **Defining the sort order of the result list based on the title**: Complete the URL by the part **propertysort=property_caption.**

https://<base address>/dms/r/<RepositoryID>/sr/?fulltext=&objectdefinitionids=["RECH"]&propertysort=property_caption

As response you will receive the HTML page with the result list. If there is an error, you will receive an HTML page that describes the error in more detail. If errors occur, you must correct the incorrect request.

Defining the parameters for a search operation and a result list

If, for example, you want to provide your users with a predefined search operation or result list, you can specify a variety of parameters. Usually, the full-text search in a d.3 repository is not specific enough. Therefore, you can restrict the search query by using query parameters.

You must encode the URL parameters (e.g. spaces in **%20**). The length of the encoded query parameter is limited to 2000 characters.

Parameter	Description
objectdefinitionids	Sets the categories used for searching items. You can define at least one category. Specify the ID of a category. If this value is not specified, the search is done across all categories in a d.3 repository. Examples (not encoded): • **objectdefinitionids=["RECH"]**: for performing a search across the category "Invoice" (**"RECH"**). • **objectdefinitionids=["RECH","AUFT"]**: for performing a search across the categories "Invoice" (**"RECH"**) and "Orders" (**"AUFT"**).
fulltext	Specifies a full-text keyword.
properties	Specifies the search restriction based on the properties of documents and dossiers. You can use the following criteria to restrict a search operation: • Document ID: **property_document_id** • File type: **property_filetype** • Document number: **property_document_number** • Document status: **property_state** (allowed values are **Be** for Processing (Bearbeitung), **Pr** for Verification (Prüfung), **Fr** for Release (Freigabe) and **AR** for the status Archive (Archiv). • Editor: **property_editor** (d.3 user ID) • File name: **property_filename** • Import date: **property_creation_date** • File size: **property_size** (when searching for the file size, the size must be specified in bytes and as an integer. A search within a specific range can be done using the separator pipe and minus character (|-): Search for documents whose document file is smaller than or equal to 1024 bytes with the expression **{"property_size":["|-1024"]}**. • Last modified: **property_last_modified_date** • Last file update: **property_last_alteration_date** • Access date: **property_access_date** • Remark: **property_remark** • Color code: **property_colorcode** (is an integer value between 1 and 24 that corresponds to the number of the color code you want to assign). • Variant number: **property_variant_number** You can define at least one value for each property. Examples (not encoded): • **properties={"property_filetype":["docx"]}**: for searching the file type **"docx"**. • **properties={"property_filetype":["docx","pdf"]}**: for searching for items with the file type **"docx"** or **"pdf"**. Restriction in relation to the definition of a search operation: You can also define multiple values for a parameter, provided that facets have been configured for the corresponding property. If no facets for viewing have been configured for the property, the last value is always adopted for the search operation. You can also limit the result list and search operation based on an advanced property. The ID of the property corresponds to the ID (RID) of a property that you can determine for each advanced property in d.3 admin. Examples (not encoded): • **properties={"227":["KND001"]}**: for searching for items that have the advanced property with the ID **"227"** and this property has the value **"KND001"**. • **properties={"227":["KND001","KND002"]}**: for searching for items that have the advanced property with the ID **"227"** and this property has the value **"KND001"** or **"KND002"**. You can also use several properties as search restrictions simultaneously. Examples (not encoded): • **properties={"227":["KND001"],"231":["|-100"]}**: for searching for items with the customer number (advanced property with the ID **"227"**) **"KND001"** and an invoice amount (advanced property with the ID **"231"**) lower than or equal to 100. • **properties={"227":["KND001"],"property_filetype":["pdf"]}**: for searching for items with the customer number (advanced property with the ID **"227"**) **"KND001"** and the file type **"pdf"**.

Note

Specific values for the **properties** parameter related to the various options for restricting the result in a targeted way:

1. **Search for a numeric value or a money value:**
   Specify the value without using a thousand separator. The decimal separator is the period (.). Example: For the value 1,000.20 EUR specify 1000.20.
2. **Search for a date and time:**
   Specify the date in the format YYYY-MM-DD. Example: Enter 2014-12-05 for the date 12/05/2014 (MM/DD/YYYY).
   Time is specified in the format YYYY-MM-DDTHH:mm:ss+01:00. You must encode the plus character (+) with **%2b**. Example: 2015-02-18T23:59:59%2b01:00 for 02/18/2015 at 11:59 p.m. and 59 seconds in the time zone UTC+1 for standard time in Germany.
3. **Search for items that are located in a specific range:**
   To search in a range, use a combination of the pipe and minus character (|-) as separators. Examples for a numeric field with the ID **"231"**:

- Values greater than or equal to 100: **{"231":["100|-"]}**
- Values smaller than or equal to 100: **{"231":["|-100"]}**
- Values between 100 and 200: **{"231":["100|-200"]}**

1.4.3. Connecting external systems (DMSApp)

In this chapter, you can learn how to connect external systems (third-party applications) to DMSApp to create mapping in DMSApp. From the perspective of DMSApp, an external system represents a source system. In the Basic information about mapping chapter, you can find general information for helping you to create mappings.

- Defining a source system
- Retrieving the default source system for a d.3 repository
- Retrieving and viewing the results of a search operation
- Retrieving and viewing the details of a DMS object
- Storage of DMS objects
- Deleting the current version of a DMS object without user interaction
- Retrieving, saving and editing mapping
- Retrieving and saving the notes of a DMS object
- Retrieving and displaying the versions of a DMS object
- Linking DMS objects
- Removing a DMS object link

Defining a source system

Released: Extension point

If you have an system that provides output data (e.g. an e-mail application or an ERP system), you must prepare the items of the d.3 repository source system for the creation of mapping. An item in a source system is referred to as a source. A source contains source properties and source categories. By defining a source, you give DMSApp the opportunity to connect the identifier for the source metadata to the identifiers for the d.3 repository metadata.

Preparing your app

To ensure that DMSApp can find source systems, your app must provide this user interface. To find apps that act as source systems, DMSApp uses the concept of an **HTTP GET** request for the root resource

(**systemBaseUri** path with the app name) for the apps. All apps registered to the d.ecs http gateway will be requested. Make sure that the administrator did not actively exclude your app.

Providing the URL for the sources

The root resources for the apps are requested cyclically. The response of an app is then checked to determine whether a link relation with the name **sources** is included. This link relation serves as the signal that the app, acting as the source system, offers at least one source for DMSApp. DMSApp executes an **HTTP GET** request to the specified link and waits for a standardized JSON object with the HTTP status code 200 from the responding app.

Request

```
GET https://host/myapp/ HTTP/1.1
Accept: application/hal+json
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
    "_links" : {
        "sources" : {
            "href" : "/myapp/sources"
        }
    }
}
```

> **Note**
> The root resources are queried anonymously (without authentication) and this process is executed as an asynchronous background process within DMSApp.

Providing sources

Once the URLs of the sources (**sources**) have been determined, DMSApp opens the sources of the apps using **HTTP GET**.

The example shows how to arrange the HTTP response of the **sources** link relation in order to identify sources for DMSApp.

Request

```
GET https://host/myapp/sources HTTP/1.1
Accept: application/hal+json
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
        "sources" : [{
                "id" : "/myapp/sources/mysource",
                "displayName" : "My Source",
                "categories": [{
                        "key": "mycategory1_ID",
                        "displayName": "My category 1"
```

```
                },
                {
                        "key": "mycategory2_ID",
                        "displayName": "My category 2"
                }],
                "properties" : [{
                        "key" : "myprop1_ID",
                        "displayName" : "My property 1"
                },
                {
                        "key" : "myprop2_ID",
                        "displayName" : "My property 2"
                },
                {
                        "key" : "myprop3_ID",
                        "displayName" : "My property 3"
                }]
        }]
}
```

Structure of a source

Property	Property of a contained object	Description
sources	-	The source system can deliver multiple sources in this array (e.g. e-mail, attachments, etc.) in order to distinguish between different mapping types.
	id	Indicates the unique identifier of the source. The ID must be a relative URI. The relative URI should start with the name of the app that provides the source system in order to ensure uniqueness (e.g. **/myapp/sources/mysource**).
	displayName	Indicates the display name as displayed in the **Source** field in the **Mappings** administrator interface. For internationalization purposes, DMSApp works with the HTTP header **Accept-Language**. This HTTP header must be evaluated by the source system, and the display name for the source must be displayed in the corresponding language.
	categories	Indicates the array of the categories for the source that are provided for managing and processing mapping.
	properties	Indicates the array of the properties for the source that are provided for managing and processing mapping.

Structure of a source category

Property	Description
key	Indicates the unique identifier of the source category.
display-Name	Indicates the display name as displayed in the **Source** field in the **Categories** area on the **Mappings** administration user interface. For internationalization purposes, DMSApp works with the HTTP header **Accept-Language**. This HTTP header must be evaluated by the source system, and the display name for the category must be displayed in the corresponding language.

Structure of a source property

Property	Description
key	Indicates the unique identifier of the source property.

Property	Description
display-Name	Indicates the display name as displayed in the **Source** field in the **Source** area on the **Mappings** administration user interface. For internationalization, DMSApp works with the HTTP header **Accept-Language**. This HTTP header must be evaluated by the source system, and the display name for the property must be displayed in the corresponding language.

> **Note**
> Make sure that: The sources must be returned quickly in order to ensure that the response time of the **Mappings** feature is not negatively affected.

Retrieving the default source system for a d.3 repository

Released: JSON representation

The default source system is a predefined source system for each d.3 repository which is provided by default by the DMS app. When you provide an extension to DMS functions and do not want to define your own source system, you can use the properties and categories of the d.3 repository with the default source system.

The definition of a default source system and the associated mappings are set by the DMS app. You cannot change mappings for a default source system.

This chapter explains how to retrieve the source system definition for each d.3 repository.

To retrieve the source system definition of a d.3 repository, you must perform the following steps:

- Determining the URL for a repository
- Determine the link relation for retrieving the source system definition of a d.3 repository
- Open the URL for the source system definition of a d.3 repository

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine the link relation for retrieving the source system definition of a d.3 repository

The JSON object for a d.3 repository contains the link relation **source** which you can use to retrieve the source system definition of the d.3 repository.

Response

```
{
        "_links": {
                "source": {
                        "href": "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/source"
                }
        },
        "id": "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Open the URL for the source system definition of a d.3 repository

You need to ensure that you can authenticate yourself for the d.3 repository. Only then you can retrieve the URL. The default source system contains all categories and properties of the d.3 repository regardless of the permissions of the user.

Retrieve the source system definition of the d.3 repository with the previously determined URL as follows:

Request

```
GET https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/source HTTP/1.1
Accept: application/hal+json
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
        "id": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/source",
        "displayName": "D3RepositoryName",
        "properties": [
                {
                        "key": "Key0",
                        "type": "String",
                        "displayName": "Sample property1"
                }
        ],
        "categories":[
                {
                        "key": "mycategory1_ID",
                        "displayName": "Sample category1"
                }
        ]
}
```

Structure of a source

Property	Description
id	Specifies the unique identifier of the default source system. You use this ID as a value for the parameter **sourceId** for more API functions.
displayName	Specifies the display name of a d.3 repository.
categories	Specifies the array of categories of the queried source system.
properties	Specifies the array of properties of the queried source system.

Structure of a category

Property	Description
key	Specifies the unique identifier of the category in the source system.
displayName	Specifies the display name of the category. For internationalization purposes, DMSApp works with the HTTP header **Accept-Language**. This HTTP header ensures that the display name of the category is displayed in the corresponding language.

Structure of a property

Property	Description
key	Specifies the unique identifier of the property in the source system.
type	Specifies the type of the property. The type of the property is defined by the administrator when creating the property. Possible values are: **String**, **ColorCode**, **Date**, **DateTime**, **Double**, **Money**.

Property	Description
displayName	Specifies the display name of the property. For internationalization, DMSApp works with the HTTP header **Accept-Language**. This HTTP header ensures that the display name of the property is displayed in the corresponding language.

Retrieving and viewing the results of a search operation

Released: JSON representation, HTML page

When retrieving and viewing the results of a search operation, you can restrict the number of results displayed to you by specifying a particular source plus the associated source categories and source properties. You can also restrict your search results using a full-text search term. The Defining a source system chapter explains how to create a source for mapping.

To retrieve or display only the results of a search operation, you must perform the following steps:

- Determining the URL for a repository
- Determine the link relation for retrieving the results of a search operation
- Specifying response control parameters
- Open the URL for the results of a search operation

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determining the link relation for retrieving or viewing the results of a search operation

You open the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for the repository contains the **searchresultwithmapping** link relation with placeholders for the values used to retrieve or view the results of the search operation.

Response

```
{
        _links: {
                searchresultwithmapping: {
                        href: "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/
srm{?
sourceid,sourceproperties,sourcecategories,sourcepropertysort,ascending,full
text,page,pagesize}",
                        templated: true
                }
        },
        id: "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Specifying response control parameters

Use the following parameters to control the system response when retrieving or viewing the results of a search operation: You must encode the URL parameters (e.g. spaces in **%20**). The length of the encoded query parameter is limited to 2000 characters.

Parameter	Description
sourceid	Defines the source to which the mapping used to retrieve the results of the search operation belongs. Only the source properties and categories that have been mapped to d.3 properties and d.3 categories are used for the individual items in the result. If no source is specified, only the IDs and the link relations for the DMS object will be returned for the individual items in the result.
sourceproperties	Specifies the search restriction based on the properties of documents and dossiers from the perspective of the source. Species the ID of the mapped source properties in order to restrict a search procedure to certain criteria. The search procedure is executed based on the mapping. Regular expressions specified by the administrator during mapping are ignored. You must specify the search restrictions as a JSON object. You can define at least one value for each property. Examples (not encoded): • **sourceproperties={"myprop1_ID":["Test e-mail 1"]}**: for a search for items with the d.3 property with the value **"Test e-mail 1"** that has been mapped to the source property with the ID **"myprop1_ID"**. • **sourceproperties={"myprop1_ID":["Test e-mail 1", "Test e-mail 2"]}**: for a search for items with the d.3 property with the value **"Test e-mail 1"** or **"Test e-mail 2"** that has been mapped to the source property with the ID **"myprop1_ID"**. Restriction in relation to the definition of a search operation: You can also define multiple values for a parameter, provided that facets have been configured for the corresponding d.3 property. If no facets for viewing have been configured for the d.3 property, the last value is always adopted for the search operation. You cannot use source properties, which are mapped to the common properties for comment fields (Comment 1 - 4), for the Search feature, since searching for individual comment fields in the d.3 repository is not supported. You can also use several properties as search restrictions simultaneously. Examples (not encoded): • **sourceproperties={"myprop1_ID":["Test e-mail 1"],"myprop2_ID":["John Doe"]}**: for a search for items with the d.3 property with the value **"Test e-mail 1"**, that has been mapped to the source property with the ID **"myprop1_ID"** and the d.3 property with the value **"John Doe"** what has been mapped to the source property with the ID **"myprop2_ID"**.
sourcecategories	Sets the categories used for searching items. You can define at least one category. Enter the ID of the source category, otherwise the search will be performed across all the categories in a d.3 repository. Sets the source categories used for searching items. Enter the ID of the source category. The search procedure is executed based on the mapping. You must specify the search restrictions as a JSON array. You can specify one or more source categories. If you do not specify a source category, the search will be performed across all the categories in a d.3 repository. Examples (not encoded): • **sourcecategories=["mycategory1_ID"]**: for a search in the d.3 category that has been mapped to the source category with the ID **"mycategory1_ID"**. • **sourcecategories=["mycategory1_ID","mycategory2_ID"]**: for a search in the d.3 categories that have been mapped to the source category with the ID **"mycategory1_ID"** or **"mycategory2_ID"**.
sourceproperty-sort	Specifies the ID of the mapped source property that is used to sort the result. If no sort property is specified, the default sort order using the **Last modified** sort criteria is applied.
ascending	Specifies the direction of the sort order. • **ascending=true**: results in an ascending sort order (from small to large (A-Z) and from old to young). • **ascending=false**: results in an descending sort order (from large to small (Z-A) and from young to old). If the **ascending** parameter is not explicitly specified, the ascending sort order is applied. This is not true for the default sort order: If the **Last modified** criteria is used for sorting and the sort order is not specified, the sort order will be descending. In addition to this, dossiers are displayed prior to documents in the result list. Within documents and dossiers, the items are sorted by the specified sorting criterion.
fulltext	Specifies a full-text keyword.
page	Specifies which page of the result list is requested. If this parameter is not transferred, Page 1 will be requested.

Parameter	Description
pagesize	Specifies how many items are displayed per page. If this parameter is not transferred, 25 items per page will be requested.

Note

Specific values for the **sourceproperties** parameter related to the various options for restricting the result in a targeted way:

1. **Search for a numeric value or a money value:**
   Specify the value without using a thousand separator. The decimal separator is the period (.). Example: For the value 1,000.20 EUR specify 1000.20.
2. **Search for a date and time:**
   Specify the date in the format YYYY-MM-DD. Example: Enter 2014-12-05 for the date 12/05/2014 (MM/DD/YYYY).
   Time is specified in the format YYYY-MM-DDTHH:mm:ss+01:00. You must encode the plus character (+) with **%2b**. Example: 2015-02-18T23:59:59%2b01:00 for 02/18/2015 at 11:59 p.m. and 59 seconds in the time zone UTC+1 for standard time in Germany.
3. **Search for items that are located in a specific range:**
   To search in a range, use a combination of the pipe and minus character (|-) as separators. Examples for a numeric field with the ID **"231"**:

- Values greater than or equal to 100: **{"231":["100|-"]}**
- Values smaller than or equal to 100: **{"231":["|-100"]}**
- Values between 100 and 200: **{"231":["100|-200"]}**

Opening the URL for the results of a search operation (JSON representation)

Once you have created a URL, you can then open the results of the search operation as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/srm?sourceid=/myapp/sources/
mysource&sourceproperties={"myprop1_ID":["Test E-Mail
1"]}&sourcecategories=["mycategory1_ID"]&sourcepropertysort=myprop1_ID&ascen
ding=true&fulltext=test&page=1&pagesize=50
Accept: application/json
```

The following JSON object will then be returned as a result:

Response

```
{
  "_links": {
    "next": {
      "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/srm?...page=2"
    },
    "self": {
      "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/srm?...page=1"
    }
  },
  "page": 1,
  "items": [
    {
      "_links": {
```

```
        "self": { "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123" },
                "previewReadonly": { href": "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/o2/D000000123/preview?isReadonly=true" }
      },
      "id": "D000000123",
      "sourceProperties": [
        {
          "key": "myprop1_ID",
          "value": "Test E-Mail 1",
          "isMultiValue": false
        },
        {
          "key": "myprop2_ID",
          "value": "Max Mustermann",
          "isMultiValue": true
        },
        "..."
      ],
      "sourceCategories": [ "mycategory1_ID" ]
    },
    "..."
  ]
}
```

Property	Description
_links	Contains the link relations for the item. **self**: Self-link. **previewReadonly**: Relative URL for retrieving the preview in read mode. Use the link relation when the preview is displayed in the inner supply (**InnerSupply**) of the operating concept. **next**: Relative URL for opening the next page of the result list. Is only specified when there are more results to display. **prev**: Relative URL for opening the previous page of the result list. Is only specified when there are previous results to display.
page	Specifies the page number of the result list.
items	Specifies the array with search operation result items for the requested page.

Structure of an item in the result list

Property	Description
_links	Contains the link relations for the item. **self**: Self-link. **delete** or **deleteWithReason**: If available, the item can be deleted with an HTTP DELETE request. For more information see Deleting the current version of a DMS object without user interaction. **update** or **updateWithContent**: If available, the item can be updated with an HTTP PUT request. For more information see Storing a new version without user interaction.
id	Specifies the document ID of the item.
sourceProperties	Specifies the array with source properties that are available for the item. If the same source property has been mapped to multiple d.3 properties that the item possesses, this source property will be returned multiple times with the respective values of the d.3 property. Regular expressions specified by the administrator during mapping are ignored.
sourceCategories	Specifies the array with the IDs of the source categories that are available for the item. The system will only return multiple categories if multiple source categories have been mapped to the d.3 category in which the item is located.

Structure of a source property

Property	Description
key	Indicates the unique identifier of the source property.
value	Specifies the value of the mapped d.3 property.
displayValue	Specifies the display value for the mapped d.3 property. Is returned only if the value (**value**) and display value (**displayValue**) are different.
isMultiValue	Specifies whether the mapped d.3 property is a multi-value property. If the d.3 property is a multi-value property, **value** will return the first value or first input value of the property (depending on the d.3 repository configuration). To retrieve all multi-values, use the request for Retrieving the details of a DMS object.

Opening the URL for the results of a search operation (HTML page)

If you want to open the HTML view of the results, you must create the URL in the same way as described for retrieving the JSON representation. Enter the URL in the browser to view the HTML page. This HTML page contains the identifiers for the d.3 properties and d.3 categories.

Examples (not encoded):

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/srm?sourceid=/myapp/sources/
mysource&sourceproperties={"myprop1_ID":["Test E-Mail
1"]}&sourcecategories=["mycategory1_ID"]&sourcepropertysort=myprop1_ID&ascen
ding=true&fulltext=test&page=1&pagesize=50
Accept: text/html
```

Retrieving and viewing the details of a DMS object

Released: JSON representation, HTML page

You can retrieve the details of a DMS object as a JSON representation or view the detail view for a DMS object. When retrieving the details as a JSON representation, you can enter a specific source from a source system to define which source properties and source categories are determined. If you do not specify a source, only the ID and the link relation for the DMS object will be returned. The Defining a source system chapter explains how to create a source.

To retrieve or display the details of a DMS object, you must perform the following steps:

- Determining the URL for a repository
- Determine the link relation for retrieving the details of a DMS object
- Specifying response control parameters
- Open the URL for the details of a DMS object

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determining the link relation for retrieving or viewing the details of a DMS object

You open the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for the repository contains the **dmsobjectwithmapping** link relation with placeholders for the values used to retrieve or view the details of the DMS object.

Response

```
{
        _links: {
                dmsobjectwithmapping: {
                        href: "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/o2m/{dmsobjectid}{?sourceid}",
                        templated: true
                }
        },
        id: "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Specifying response control parameters

Use the following parameters to control the system response when retrieving or viewing the details of a DMS object:

Parameter	Description
dmsobjectid	Specifies the document ID of the DMS object for which you want to request or view the details.
sourceid	Defines the source to which the mapping used to retrieve the details of the DMS object belongs. For information about providing a source system for mapping, see Defining a source system. If you want to use the default source system, see Retrieving the default source system for a d.3 repository for more information. **Retrieve (JSON representation)**: Only the source properties that are mapped to the d.3 properties are returned. If no source is specified, only the ID and the link relation for the DMS object will be returned. **View (HTML page)**: You do not need to enter this parameter if you wish to view the details of a DMS object. The parameter will not be evaluated.

Opening the URL for the details of a DMS object (JSON representation)

Once you have created a URL, you can then retrieve the details of the DMS object as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/D000000123?sourceid=/
myapp/sources/mysource
Accept: application/json
```

The following JSON object will then be returned as a result:

Response

```
{
  "_links": {
    "self": {
                "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123"
     },
    "mainblobcontent": {
        "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/v/current/b/main/c"
     },
```

```
    "editinoffice": {
          "href": "{ms-
word:ofe|u|{+clientOrigin}/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/dav/D000000123%20(D000000123).DOCX}",
      "templated": true
    },
        "pdfblobcontent": {
      "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/D000000123/v/
current/b/p1/c"
    },
        "notes":{
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/n"
        },
        "children":{
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/srm/?
children_of=D000000123"
        },
        "versions":{
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/v/"
        }
  },
  "id": "D000000123",
  "sourceProperties": [
    {
      "key": "myprop1_ID",
      "value": "value of property 1"
    },
    {
      "key": "myprop2_ID",
      "value": "value of property 2 in row 2",
      "values": {
            "2": "value of property 2 in row 2",
        "4": "value of property 2 in row 4"
          }
    }
  ],
  "sourceCategories": ["mycategory2_ID"]
}
```


Property	Description
_links	Contains the link relations for the DMS object. **mainblobcontent**: Relative download URL for the main document of the current version of the DMS object. **editinoffice**: URL with placeholders for editing the document in Microsoft Office. You will only receive this URL if the administrator has activated the function for editing Microsoft Office documents. The URL is unavailable if you are working with Microsoft Office 365 in the cloud. **pdfblobcontent**: Relative download URL for the generated (“dependent”) PDF document of the current version of the DMS object. You receive this URL only if a generated PDF document has been created for the DMS object. **notes**: Relative URL for opening the notes of the DMS object. You will only receive this URL if any notes have already been saved for the DMS object. **children**: Relative URL for the child DMS objects. You will only receive this URL if the DMS object has child items. **versions**: Relative URL for retrieving and displaying the versions of a DMS object. **self**: Self-link.

Property	Description
id	Specifies the document ID of the DMS object.
sourceProperties	Specifies the array with source properties that are available for the requested DMS object. If the same source property has been mapped to multiple d.3 properties that the requested DMS object possesses, this source property will be returned multiple times with the respective values of the d.3 property.
sourceCategories	Specifies the array with the IDs of the source categories that are possible for the requested DMS object. The system will only return multiple categories if multiple source categories have been mapped to the d.3 category in which the requested DMS object is located.

Structure of a source property

Property	Description
key	Indicates the unique identifier of the source property.
value	Specifies the value of the mapped d.3 property. If the d.3 property is a multi-value property, **value** will return the first value or first input value of the property (depending on the d.3 repository configuration).
values	Specifies the values of the mapped d.3 property. Only returned if the d.3 property is a multi-value property. **values** is an object comprised of name-value pairs (key value): Name: Row number (starting with 1). Value: Value of the property in the corresponding row.
displayValue	Specifies the display value for the mapped d.3 property. Is returned only if the value (**value**) and display value (**displayValue**) are different.

Opening the URL for the details of a DMS object (HTML page)

If you want to open the HTML view of the results, you must create the URL in the same way as described for querying the JSON representation. Enter the URL in the browser to view the HTML page. This HTML page contains the identifiers for the d.3 properties and d.3 categories.

Example:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/D000000123
Accept: text/html
```

Storage of DMS objects

When storing DMS objects, you can determine the storage metadata by specifying a particular source plus the associated source categories and source properties. The Defining a source system chapter explains how to create a source for mapping. In the Basic information about mapping chapter, you can find general information for helping you to create mappings.

In this topic, you will learn how to store data from a source using the DMSApp.

- Storing a new DMS object without user interaction
- Updating a DMS object without user interaction
- Storing new DMS objects with user interaction
- Updating DMS objects with user interaction
- Changing the current editor and document status

Please note the following limitations (only applies to the d.velop documents cloud offer):

- max. 3000 updates of DMS objects per client (client IP) in a period of 5 minutes
- max. 8000 saves of new DMS objects per client (client IP) in a period of 5 minutes

Storing a new DMS object without user interaction

You can store a new DMS object automatically in the d.3 repository without the need for action by the user. No **Validate** hooks are executed in d.3 server during storing without user interaction. Mappings are required before the action can be performed. In the Basic information about mapping chapter, you can find general information for helping you to create mapping.

Perform the following steps to store a DMS object:

- Determining the URL for a repository
- Determine the link relation for storing a new DMS object
- Provide the file to be stored (optional)
- Open the URL for storing a new DMS object

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determining the link relation for storing a new DMS object

You open the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for a repository contains the link relation **dmsobjectwithmapping**.

Response

```
{
        "_links": {
                "dmsobjectwithmapping": {
                        "href": "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/o2m/{dmsobjectid}{?sourceid}",
                        "templated":true
                }
        },
        "id": "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

The values **dmsobjectid** and **sourceid** are not relevant for storage. They do not need to be filled when executing the template.

Provide the file to be stored (optional)

The Provision of files chapter explains how to obtain the **contentUri** or **contentLocationUri**. These parameters are required for storing a new DMS object.

If you want to create a dossier, you do not need to provide the file.

Open the URL for storing a new DMS object

Perform an **HTTP POST** request to this URL with the required properties as **Body**.

Request

```
POST /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m
Origin: https://baseuri
```

```
Accept: application/hal+json
Content-Type: application/hal+json

{
        "filename": "myfile.txt",
        "sourceCategory": "mycategory1_ID",
        "sourceId": "/myapp/sources/mysource",
        "contentLocationUri": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/
blob/chunk/2018-01-01_temp_master_file_user1_44f7-95a6-58b8400ecf43",
        "sourceProperties": {
                "properties": [{
                                "key": "myprop1_ID",
                                "values": ["Please verify the XYZ invoice"]
                        },
                        {
                                "key": "myprop2_ID",
                                "values":
["Name1@contoso.com","Name2@samplecompany.de"]
                        }
                ],
        }
}
```

For information about the JSON object parameters, see the chapter Defining the parameters for storing.

If no assignment is configured for the d.3 document status, the document status **Release** (**Release**) is used by default for the DMS object. The **Status** property must be assigned by the administrator if you want to use a different document status during storage.

If the call is successful, the URL is returned to the DMS object in the **Location** header:

If linking an item to a dossier (**parentId**) fails, you will receive the URL to the DMS object in the **Location** header, as well as an additional notification that the linking process has failed.

Response

```
HTTP/1.1 201 Created
Location: /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/D0001234?
sourceId=%2Fmyapp%2Fsources%2Fmysourc

{
        "reason": "You do not have the right to link the document with the
dossier manually.
Nevertheless, the document was successfully saved. The document may be
automatically linked on the server."
        "severity": 1
}
```

If the call was successful and there is detailed information on the processing result of the storage operation, the detailed information is returned in the response. With the help of the information, you can decide whether further processing steps are necessary for the DMS object. For more information see Format of the response for successful requests with detailed information.

If storing the DMS object fails, an appropriate answer is displayed. For more information see Response format for errors.

Updating a DMS object without user interaction

You can update an existing DMS object in a d.3 repository without the need for action by the user. No **Validate** hooks are executed in d.3 server while updating the DMS object without user interaction. Mappings are required before the action can be performed. In the Basic information about mapping chapter, you can find general information for helping you to create mapping.

To update a DMS object, perform the following steps:

- Determining the URL for a repository
- Determine the link relation for the existing DMS object
- Provide the file to be stored (optional)
- Open the URL to update the DMS object

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine the link relation for the existing DMS object

To determine a URL for an existing DMS object, perform a search and evaluate the **update** or **updateWithContent** link relation for an item in the result list. The chapter Retrieving and viewing the results of a search operation provides you with more information about performing a search and the description of a result list item. If the d.3 document status of the existing DMS object is **Processing**, the authenticated user must be the processor of the DMS object.

Provide the file to be stored (optional)

If you have determined an **updateWithContent** link relation, you can provide a file with the parameters **contentUri** or **contentLocationUri**. In the chapter Provision of files, you can learn how to provide a file.

If you want to update only the properties with the **update** link relation, you do not need to provide a file.

If you have received only an **update** link relation, an update with a file is not possible.

Open the URL to update the DMS object

Send an **HTTP PUT** request with the required properties as the **Body** to the URL of the existing DMS object that you received in the **update** or **updateWithContent** link relation. If you want to modify only parameters, do not specify the **contentLocationUri** and **filename** parameters.

Request

```
PUT /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/A00000001
Origin: https://baseuri
Accept: application/hal+json
Content-Type: application/hal+json

{
        "filename": "myfile.txt",
        "alterationText": "updated file",
        "sourceCategory": "mycategory1_ID",
        "sourceId": "/myapp/sources/mysource",
        "contentLocationUri": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/
blob/chunk/2018-01-01_temp_master_file_user1_44f7-95a6-58b8400ecf43",
        "sourceProperties": {
                "properties": [{
                                "key": "myprop1_ID",
                                "values": ["Please verify the XYZ invoice"]
```

```
                        },
                        {
                                "key": "myprop2_ID",
                                "values":
["Name1@contoso.com","Name2@samplecompany.de"]
                        }
                ]
        }
}
```

For information about the JSON object parameters, see the chapter Defining the parameters for storing.

You cannot transfer the status of the document using the URL described above. Use the URL of the current version of the document to transfer status. For more information, see Change the current editor and document status.

If the call is successful, **HTTP 200 OK** is returned:

Response

```
HTTP/1.1 200 OK
```

If the call was successful and there is detailed information on the processing result of the storage operation, the detailed information is returned in the response. With the help of the information, you can decide whether further processing steps are necessary for the DMS object. For more information see Format of the response for successful requests with detailed information.

If the version fails to save, a corresponding answer (response) is returned. For more information see Response format for errors.

If the call was successful and there is detailed information about the processing result of the storage operation, this detailed information is returned in the response. You can use this information, for example, to make decisions about whether further processing steps are necessary for the DMS object. For more information see Format of the response for successful requests with detailed information.

Storing new DMS objects with user interaction

Released: HTML page

You can open the **Storage** feature to store new DMS objects with properties and files. Mappings are required before the action can be performed. In the Basic information about mapping chapter, you can find general information for helping you to create mappings.

Perform the following steps to store DMS objects:

- Determine the link relation for the storage dialog
- Provide the file to be stored (optional)
- Create a storage dialog with predefined properties and files
- Display the storage dialog with predefined properties and files
- Perform more actions after storing (optional)

Determine the link relation for the storage dialog

There are two link relations for the **Storage** feature:

- Link relation without reference to a d.3 repository: Storage without reference to a d.3 repository allows the user to select the d.3 repository. In this case, however, you cannot temporarily upload the file. For more information about providing files see Provision of files.
- Link relation with reference to a d.3 repository: When storing with reference to a d.3 repository, the user is no longer able to change the d.3 repository.

To determine the link relation to a d.3 repository, execute an **HTTP GET** request for the REST resource **/dms**.

Request

```
GET /dms
Accept: application/hal+json
```

The JSON object contains the link relation **new**.

Response

```
{
    "_links": {
        "new": {
            "href": "/dms/new/"
        }
    }
}
```

You must know the repository URL in order to determine the link relation with reference to a d.3 repository. In the chapter Determining a repository, you can learn how to determine the URL for a repository. Then execute an **HTTP Get** request on the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for a repository contains the link relation **new**.

Response

```
{
        "_links": {
                "new": {
                        "href": "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/new/"
                }
        },
        "id": "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Provide the file to be stored (optional)

The Provision of files chapter explains how to obtain the **contentUri** or **contentLocationUri**. This parameter is required to create a storage dialog for a new DMS object.

If you want to create a dossier, you do not need to provide the file.

Create a storage dialog with predefined properties and files

Send an **HTTP POST** request with the required properties as the **Body** to the URL that you received in the **self** link relation.

Request

```
POST /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/new
Origin: https://baseuri
```

```
Accept: application/hal+json
Content-Type: application/hal+json

{
    "storeObjects": [{
        "displayValue": "Please verify the XYZ invoice",
        "filename": "myfile.txt",
        "contentLocationUri": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/
blob/chunk/2018-01-01_temp_master_file_user1_44f7-95a6-58b8400ecf43",
        "sourceId": "/myapp/sources/mysourcel",
                "sourceCategory": "mycategory1",
                "sourcePropertiesUri": "/myapp/sources/mysourcel/properties/
myfile.haljson",
                "successCallbackUri": "/myapp/sources/mysourcel/success/
myfile"
    },
    {
        ...
    }]
}
```

The JSON object that is transferred to **POST** is described as follows:

Property	Description
storeObjects	Specifies the array of item properties and files that should be used to define the storage dialog. For information about an item of the array, see Defining the parameters for storing.

To store a single DMS object, you can also specify the JSON object of the item properties without the **storeObjects** parent array.

If the call is successful, the URL is returned to the storage dialog in the **Location** header:

Response

```
HTTP/1.1 201 Created
Location: /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/new/
a09dd457-5a21-4b90-8134-e562092b50ea
```

If creating the storage dialog fails, an appropriate response is displayed. For more information see Response format for errors.

Displaying the storage dialog with predefined properties and files

Open the URL that you received in the **Location** header in the browser to show the created storage dialog to the user. The user can edit the properties and complete the storage of the DMS objects.

Perform more actions after storing (optional)

The URL is opened if the **successCallbackUri** parameter is specified while creating the storage dialog and the user has successfully completed the storage process. You can perform additional steps after performing this action. For more information, see Feedback using "SuccessCallback" and "Userdata".

Updating DMS objects with user interaction

Released: HTML page

You can select the **Storage** feature to update the properties and file of a DMS object. Mappings are required before the action can be performed. In the Basic information about mapping chapter, you can find general information for helping you to create mappings.

To update a DMS object, perform the following steps:

- Determine the ID for the existing DMS object
- Provide the file to be stored (optional)
- Determine the link relation for the storage dialog
- Create a storage dialog with predefined properties and files
- Displaying the storage dialog with predefined properties and files
- Perform more actions after storing (optional)

Determine the ID for the existing DMS object

In the chapter Searching for DMS objects, you can learn how to determine the ID for an existing DMS object. If the d.3 document status of the existing DMS object is **Processing**, the authenticated user must be the processor of the DMS object.

Provide the file to be stored (optional)

In the chapter Provision of files, you can learn how to provide a file.

Determine the link relation for the storage dialog

In the chapter Determining a repository, you can learn how to determine the URL for a repository. Then, execute an **HTTP Get** request on the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for a repository contains the link relation **new**.

Response

```
{
        "_links": {
                "new": {
                        "href": "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/new/"
                }
        },
        "id": "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Create a storage dialog with predefined properties and files

Send an **HTTP POST** request with the required properties as the **Body** to the URL that you received in the **self** link relation.

Request

```
POST /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/new
Origin: https://baseuri
Accept: application/hal+json
Content-Type: application/hal+json

{
    "storeObjects": [{
        "dmsObjectId": "D123456789",
        "displayValue": "Please verify the XYZ invoice",
```

```
        "filename": "myfile.txt",
        "contentLocationUri": "/dms/blob/chunk/2016-
May-24-14-20-26-393-1720.part",
        "sourceId": "/myapp/sources/mysourcel",
        "sourcePropertiesUri": "/myapp/sources/mysourcel/properties/
myfile.haljson",
        "successCallbackUri": "/myapp/sources/mysourcel/success/myfile"
    },
    {
        ...
    }]
}
```

The JSON object that is transferred to **POST** is described as follows:

Property	Description
storeObjects	Specifies the array of item properties and files that should be used to define the storage dialog. For information about an item of the array, see Defining the parameters for storing.

To update a single DMS object, you can also specify the JSON object of the item properties without the **storeObjects** parent array.

If the call is successful, the URL is returned to the storage dialog in the **Location** header:

Response

```
HTTP/1.1 201 Created
Location: /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/new/
a09dd457-5a21-4b90-8134-e562092b50ea
```

If creating the storage dialog fails, an appropriate response is displayed. For more information see Response format for errors.

Displaying the storage dialog with predefined properties and files

Open the storage dialog URL that you received in the **Location** header in the browser to display the storage dialog to the user. The user can edit the properties and finish updating the DMS objects.

Perform more actions after storing (optional)

The URL is opened if the **successCallbackUri** parameter is specified while creating the storage dialog and the user has successfully completed the storage process. You can perform additional steps after performing this action. For more information, see Feedback using "SuccessCallback" and "Userdata".

Changing the current editor and document status

You can update the editor and document status of an existing DMS object in a d.3 repository without the need for action by the user. Mappings are required before the action can be performed. In the Basic information about mapping chapter, you can find general information for helping you to create mappings.

To change the editor or document status of a DMS object, you must perform the following steps:

- Determining the URL for a repository
- Determine the link relation for the existing DMS object
- Open the URL for changing the editor or document status of a DMS object

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine the link relation for the existing DMS object

To determine a URL for an existing DMS object, perform a search and evaluate the **displayVersion** link relation for an item in the result list. The chapter Retrieving and viewing the results of a search operation provides you with more information about performing a search and the description of a result list item.

Opening the URL for changing the DMS object status and editor

Send an **HTTP PUT** request with the required properties as the **Body** to the URL of the current available version of the DMS object that you received in the **displayVersion** link relation.

You can find descriptions of the JSON object transferred with the **PUT** request below:

Property	Description
property_state	Target state of the document. Possible values: • **Processing** • **Verification** • **Release** • **Archive**
property_editor	Mandatory field for the target status **Processing** and optional for the target status **Verification**. The parameter is ignored in other target statuses. Possible values: • Target status **Processing**: User or group ID to which the document is to be assigned. • Target status **Verification**: Group ID to which the document is to be assigned. You can use the IDs from both d.ecs identity provider and the DMS system.
alterationText	Mandatory field for the target status **Release**. Text (max. 120 bytes) that is stored when the status changes for the document version. You can provide information about the extent of the change, for example.

Example 1: Changing a document status to “Processing” or changing the editor with your own mapping

Ensure that mapping exists in which the value **myprop1_ID** is set to **property_state** and **myprop2_ID** is set to **property_editor**.

Request

```
PUT/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/A00000001/v/current
Origin: https://baseuri
Accept: application/hal+json
Content-Type: application/hal+json

{
   "sourceId": "/myapp/sources/mysource",
   "sourceProperties": {
      "properties": [
         {
            "key": "myprop1_ID",
            "values": [
               "Processing"
            ]
         },
         {
            "key": "myprop2_ID",
            "values": [
               "97273358-d124-497c-97ce-977f72b32a33"
            ]
         }
```

```
      ]
   }
}
```

If the call is successful, **HTTP 200 OK** is returned:

Response

```
HTTP/1.1 200 OK
```

Example 2: Changing a document status to “Processing” or changing the editor with standard mapping

Request

```
PUT/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/A00000001/v/current
Origin: https://baseuri
Accept: application/hal+json
Content-Type: application/hal+json

{
   "sourceId": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/source",
   "sourceProperties": {
      "properties": [
         {
            "key": "property_state",
            "values": [
               "Processing"
            ]
         },
         {
            "key": " property_editor",
            "values": [
               "97273358-d124-497c-97ce-977f72b32a33"
            ]
         }
      ]
   }
}
```

If the call is successful, **HTTP 200 OK** is returned:

Response

```
HTTP/1.1 200 OK
```

Example 3: Changing a document status to “Release” with standard mapping

Request

```
PUT/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/A00000001/v/current
Origin: https://baseuri
Accept: application/hal+json
Content-Type: application/hal+json

{
   "sourceId": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/source",
   "alterationText": "updated file",
   "sourceProperties": {
```

```
      "properties": [
         {
            "key": "property_state",
            "values": [
               "Release"
            ]
         }
      ]
   }
}
```

If the call is successful, **HTTP 200 OK** is returned:

Response

```
HTTP/1.1 200 OK
```

If the change of editor or document status fails, an appropriate response is returned. For more information see Response format for errors.

General information about the storage process

In this topic, you can learn more details about storing DMS objects. However, before you begin with this topic, you should familiarize yourself with the chapters on saving and updating DMS objects.

- Defining the parameters for storing
- Provision of files
- Response format for errors
- Feedback using "SuccessCallback" and "Userdata"

Defining the parameters for storing

In this chapter, you learn more about the parameters you can use to store DMS objects. The parameters are applicable for storing with or without user action.

Setting a JSON object for storing

The JSON object that is sent with the **HTTP** requests for storing a DMS object includes the following parameters:

Property	Description
displayValue	Specifies the display name of the item that is stored. The display name will become visible to the user in the storage dialog. This parameter is optional.
filename	Specifies the name of the file to be stored with the file extension. If you set this parameter, you need to map the category which you specified under **sourceCategory** in the mapping processing for a document type. If you do not set this property, you need to map the category which you specified under **sourceCategory** the in mapping processing for a dossier type. • The property is mandatory if you are resaving a document. • This property is optional if you are updating a document. If the parameter is not set, the old file name will be used. • The property is ignored if you update a document without **contentLocationUri** or **contentUri** . • The property must not be set when you create or update a new dossier.
dmsObjectId	Specifies the ID of the DMS object to be updated. This parameter is taken into account when updating a DMS object with user interaction.
alterationText	Specifies the reason for change for the update of a DMS object. In d.3 admin, you define whether the parameter has to be entered. This parameter is taken into account only when updating a DMS object.

Property	Description
sourceCategory	Specifies which category should be used for mapping processing while storing. Enter the ID of the source category. This property must be specified except under the following circumstances: • The property is ignored if the parameter **parentId** is set and an associated default category has been defined in d.3 admin. • The property is ignored when you update a DMS object. • The property is optional if an item is stored via user action. If you store multiple DMS objects via user action, then only the source category of the first item in the **storeObjects** array in the storage dialog is taken into account. This source category is displayed to the user for all additional files to be stored.
sourceId	Defines the source to which the mapping that is to be used for storing belongs. The source must exist and the source system must be registered with d.ecs http gateway so that the properties under **sourcePropertiesUri** or **sourceProperties** can be applied. For more information about providing a source see Defining the parameters for storing. If you do not have your own source system, you can also use the default source system of the d.3 repository. For information on the standard source system, see Retrieving the default source system for a d.3 repository.
sourcePropertiesUri	Specifies the URL through which the DMSApp should download the item properties for the mapping processing and duplicate check. Before storing the DMS object, the DMSApp sends an **HTTP GET** request to the URL in order to determine the source properties for the storage process. If the **sourcePropertiesUri** is a relative URL, the user information of the registered user is also transmitted in the **HTTP** request (**AuthSessionID**). This makes it possible to take the current user name into account or perform a permission check, for example. In the case of an absolute URL, the source properties are requested anonymously. The JSON object to be provided is described under "**Setting the source properties**." Alternatively, you can use the **sourceProperties** property.
sourceProperties	Specifies a JSON array with source properties that are used for mapping processing and duplicate check. The JSON object to be provided is described under "**Setting the source properties**." Alternatively, you can use the **sourcePropertiesUri** property.
contentLocationUri	Specifies the URL for the file that was temporarily uploaded. For more information, see Providing a file via temporary upload. Alternatively, you can use the **contentUri** property. If you set this parameter, you need to map the category which you specified under **sourceCategory** the in mapping processing for a document type. If you do not specify neither the **contentLocationUri** nor the **contentUri** property, you need to map the category which you specified under **sourceCategory** in the mapping processing for a dossier type. This property is optional if you want to update a DMS object or create a dossier.
contentUri	Specifies the relative URL through which the DMSApp downloads the file to be stored. For more information, see Providing a file via temporary upload. Alternatively, you can use the **contentLocationUri** property. If you set this parameter, you need to map the category which you specified under **sourceCategory** the in mapping processing for a document type. If you do not specify neither the **contentLocationUri** nor the **contentUri** property, you need to map the category which you specified under **sourceCategory** in the mapping processing for a dossier type. This property is optional if you want to update a DMS object or create a dossier.
parentId	Specifies the DMS object ID of the dossier with which the item to be stored should be linked. If the storage process is successful, the stored item is linked to the corresponding dossier. If the item to be stored is recognized as a duplicate, the already existing item is linked to the dossier. However no duplicate check performed, when you want to link a dossier to another dossier. This property is ignored when you update a DMS object.
successCallbackUri	Specifies the URL that is opened via **HTTP POST** when the storage process was performed successfully. The URL must be relative. For more information, see Feedback using "SuccessCallback" and "Userdata". This property is optional. You can use the property only when storing with user interaction.

Setting the source properties

You can define the source properties directly using the **sourceProperties** parameter or provide them using the URL in **sourcePropertiesUri**.

The JSON object is described as follows:

Property	Description
properties	Specifies an array of the source properties that are provided for mapping processing. Only the properties that were configured by the administrator in the mapping are taken into consideration when storing.
uniqueTag	Specifies a unique hashcode that is used by the source for the item to be stored and by the d.3 server for duplicate recognition. Setting this property is optional.
userdata	Specifies an array of options that should also be displayed in the storage dialog. Specifying this property is optional. You can use the property only when storing with user interaction. For more information see Feedback using "SuccessCallback" and "Userdata".

Property	Description

checkHash	Indicates the hash of the file to be stored. The hash of the file to be stored helps you to validate the file saving process in a d.3 repository. If the hashes do not match, the document saving process is terminated with an error message.
	
	While saving the file, a hash of the file is generated by the server, compared with the hash that you generate and validated. In this case, the parameter must have the following structure:
	
	`HASHALGORITHM:base64encode(hashfunction(PAYLOAD))`
	
	Example of a file with the content "Example":
	
	`SHA512:xrCRnH/`
	`mKK6QVpksSpF+XcA1qWFdSX9usr0UBj6q0+ZQjvyGgv7IKCPKPz3jEYaKcpkJRhZkKfAbOPnzPZymEA==`
	
	The following hash algorithms are supported:
	
	• RIPEMD256
	• SHA256
	• SHA384
	• SHA512
	• MD5
	
	This parameter is optional.
	
	Always perform Base64 encoding at the byte level. Encoding a textual representation results in invalid hash values.
	
	Examples of generating a **checkHash** value using the classes from System.Security.Cryptography in C#:
	
	`private string CreateM5Hash(string filename)`
	`{`
	`using (var md5 = MD5.Create())`
	`{`
	`using (var stream = File.OpenRead(filename))`
	`{`
	`var retVal = md5.ComputeHash(stream);`
	`//Bytes in retVal für eine Datei mit dem Inhalt "Example":`
	`//45 155 209 108 21 173 26 160 179 111 217 119 62 110 20 172`
	
	`return $"MD5:{Convert.ToBase64String(retVal)}";`
	`//Hash base64-encodiert:`
	`//MD5:ClJzBZf7T/oB/BF9nnHjqQ==`
	`}`
	`}`
	`}`
	
	`private string CreateSHA256Hash(string filename)`
	`{`
	`using (var sha256 = SHA256.Create())`
	`{`
	`using (var stream = File.OpenRead(filename))`
	`{`
	`var retVal = sha256.ComputeHash(stream);`
	`//Bytes in retVal für eine Datei mit dem Inhalt "Example":`
	`//53 239 154 101 221 213 246 176 70 65 37 206 150 167 54 152`
	`//63 65 154 140 96 227 200 169 71 112 149 213 153 156 17 27`
	
	`return $"SHA256:{Convert.ToBase64String(retVal)}";`
	`//Hash base64-encodiert:`
	`//SHA256:0Cn4fj2A+P2bG+Z8dCa0zB/0e0qdCoRhyCalnYxets0=`
	`}`
	`}`
	`}`
	
	`private string CreateSHA512Hash(string filename)`
	`{`
	`using (var sha512 = SHA512.Create())`
	`{`
	`using (var stream = File.OpenRead(filename))`
	`{`
	`var retVal = sha512.ComputeHash(stream);`
	`//Bytes in retVal für eine Datei mit dem Inhalt "Example":`
	`//72 237 177 122 242 252 39 57 63 98 44 167 119 228 162 20 106`
	`//147 201 76 246 223 209 126 97 224 198 30 40 137 121 216 147`
	`//21 15 16 1 103 84 101 60 241 95 82 14 126 211 137 170 235`

Property	Description

	180
	//206 18 112 247 228 229 185 208 62 192 244 131 41
	
	return $"SHA512:{Convert.ToBase64String(retVal)}";
	//Hash base64-encodiert:
	//SHA512:xrCRnH/
	mKK6QVpksSpF+XcA1qWFdSX9usr0UBj6q0+ZQjvyGgv7IKCPKPz3jEYaKcpkJRhZkKfAbOPnzPZymEA==
	}
	}
	}

Structure of an object in the "property" array

Property	Description
key	Specifies the ID of the source property. Only the properties that you specified and that were configured by the administrator in the mapping are taken into account. When updating a DMS object, only the properties that you specified are updated. All the other values for the existing DMS object properties are retained. When you want to specify the creation date (**property_creation_date**) of the DMS object, the specification of the property is only considered when a new DMS object is saved without any user interaction. For more information, see Storing a new DMS object without user interaction.
values	Specifies an array with the corresponding values of the item property. Even if you only want to transmit one value, you must transmit this value as a JSON array. You can set the d.3 document status for a DMS object when storing a new DMS object. The values available are **Processing**, **Verification** and **Release**. During a DMS object update, mappings for the d.3 document status, d.3 variant number, d.3 document ID and d.3 document number are ignored. Properties with the tag "read-only" are not considered while storing the item. During an update without user interaction, the authenticated user must be the processor of the DMS object if the existing DMS object has the **Processing** d.3 document status. Specify the numerical value without using a thousand separator. The decimal separator is the period (.). Example: For the value 1,000.20 EUR specify 1000.20. Specify the date values in the format YYYY-MM-DD. Example: Enter 2014-12-05 for the date 12/05/2014 (MM/DD/YYYY). Enter date and time values in the format YYYY-MM-DDTHH:mm:ss+01:00. Example: 2015-02-18T23:59:59+01:00 for 2/18/2015 at 11:59 p.m. and 59 seconds in the time zone UTC+1 for standard time in Germany.

Example 1: Providing the file and the properties via a URL

A simple JSON object could appear as follows:

```
{
    "filename": "myfile.txt",
        "alterationText": "updated file",
    "sourceCategory": "mycategory1_ID",
    "sourceId": "/myapp/sources/mysource",
    "contentUri": "/myapp/sources/mysource/myfile.txt",
    "sourcePropertiesUri": "/myapp/sources/mysource/myfile.haljson"
}
```

Example 2: The properties, including the "userdata" and "successCallbackUri" parameters, are directly specified.

The following JSON object contains **userdata** and the properties (**properties**). **uniqueTag** is set at the same time for the duplicate check.The dossier ID is set in the **parentId** parameter in order to link the DMS object to be recreated:

```
{
    "displayValue": "Please verify the XYZ invoice",
    "filename": "myfile.txt",
        "alterationText": "updated file",
    "sourceCategory": "mycategory1_ID",
    "sourceId": "/myapp/sources/mysource",
    "parentId": "P123456789",
    "contentLocationUri": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/blob/
chunk/2018-01-01_temp_master_file_user1_44f7-95a6-58b8400ecf43",
    "successCallbackUri": "/myapp/sources/mysource/myfile/success",
    "sourceProperties": {
        "properties": [{
                "key": "myprop1_ID",
                "values": ["Please verify the XYZ invoice"]
            },
            {
                "key": "myprop2_ID",
                "values": ["Name1@contoso.com","Name2@samplecompany.de"]
            }
        ],
                "userdata": [{
                    "key": "postProcessingOption",
                        "display": "My post processing options",
                "values": [
                        {"value":"1", "display": "Action 1",
"default":"false"},
                        {"value":"2", "display": "Default action 2",
"default":"true"},
                        {"value":"3", "display": "Action 3",
"default":"false"}
                ]
                }
        ],
                "uniqueTag": "123456789",
                "checkHash": "MD5:ClJzBZf7T/oB/BF9nnHjqQ=="
    }
}
```


Provision of files

Storing DMS objects saves their properties and categories. In order for files to be stored, they must be provided to the DMSApp and the source category must be mapped to a document type in the mapping processing. You can provide the files in two ways:

- Providing a file with a URL
- Providing a file via temporary upload

Providing a file with a URL

This chapter explains how to provide a file using a URL. When storing DMS objects, provide the URL leading to a file in the **contentUri** property. Before the DMS object is stored, the DMSApp will download the file via the specified URL (**contentUri**). The URL must be a relative URL. The respective user's information is also transmitted in the **HTTP** request during the download (**AuthSessionID**). The **AuthSessionID** can be used to identify the user, e.g. to perform a permission check.

Since the file is downloaded in a request, this procedure is not suited for large files. Use the temporary file upload procedure for larger files. For additional information see Providing a file via temporary upload.

Downloading the provided file

The DMSApp performs an **HTTP GET** request to the URL specified in the **contentUri** parameter. Here is an example for the DMSApp 's **HTTP GET** request for your source system, provided the **contentUri** contains the following URL **/myapp/sources/mysource/myfile.txt**:

Request

```
GET /myapp/sources/mysource/myfile.txt
Accept: application/octet-stream, */*
AuthSessionID: SampleAuthsessionId
```

Your source system's response must look as follows:

Response

```
HTTP/1.1 200 OK
Content-Length: 3495

<binary content>
```

Providing a file via temporary upload

This chapter explains how to make a file temporarily available via temporary upload. Temporary upload is especially useful for large files. Using the (**contentLocationUri**) URL for the file that was temporarily uploaded, you can then store the DMS object.

Perform the following steps to upload a file temporarily:

- Determining the URL for a repository
- Determine the link relation used to temporarily upload the file
- Open the URL to temporarily upload the file
- Open the URL to temporarily upload the file (optional)

It is not possible to upload a file temporarily without specifying an ID for a repository.

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine the link relation used to temporarily upload the file

You open the URL for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for a repository contains the **chunkedupload** link relation.

Response

```
{
        "_links": {
                "chunkedupload": {
                        "href": "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/blob/chunk/"
                }
        },
        "id": "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Open the URL to temporarily upload the file

Send an **HTTP POST** request with the binary data as the **Body** to the URL that you received in the **chunkedupload** link relation. If successful, you will receive the **HTTP** status code **201** and the URL (**contentLocationUri**) in the **Location** HTTP header.

Request

```
POST /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/blob/chunk/
Origin: https://baseuri
Content-Type: application/octet-stream

<binary content>
```

Response

```
HTTP/1.1 201 Created
Location: /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/blob/chunk/
2018-01-01_temp_master_file_user1_44f7-95a6-58b8400ecf43
```

In the **HTTP Post** request, The whole body is considered as binary data of the file to be uploaded. Currently, the upload using **multipart/form-data** is not supported.

The maximum size of the body is limited to 100 MB. We recommend a body size of 25 MB. If you want to upload larger files, you can split the file into different chunks. For additional information, see the following section.

Open the URL to temporarily upload the file (optional)

You can divide the upload among multiple **HTTP** requests in order to avoid a timeout for larger files.

To do so, follow the steps below:

- Split the file to be uploaded into different chunks.
- Upload the first chunk as described in "**Opening the URL to temporarily upload the file**".
- Upload the next chunk by executing an **HTTP POST** request on the URL that you received in the header of the response to the first chunk under **Location**.
- Repeat the previous step successively for all chunks in the correct order.

Request

```
POST /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/blob/chunk/
2018-01-01_temp_master_file_user1_44f7-95a6-58b8400ecf43
Content-Type: application/octet-stream

<binary content for chunk>
```

Response

```
HTTP/1.1 200 OK
```

Note the following when uploading multiple chunks:

- The chunks can only be uploaded sequentially.
- The order of the **HTTP POST** request for the chunks must be strictly adhered to.
- If an error occurs during an **HTTP POST** request (e.g. **HTTP** status code is not **200**), then you must re-upload the file in all the chunks. Individual chunks cannot be reuploaded.

You receive the following possible responses to the requests:

Status code	Description
200 OK	A chunk was successfully uploaded.
201 Created	The first chunk was successfully uploaded.
404 Not Found	The resource was not found. An unknown repository ID was specified, for example.
500 Internal Server Error	An internal error occurred while processing.

Response format for errors

Released: JSON representation

In this chapter, you will learn about the output format for errors. Depending on the result of the storage process, the **HTTP** request is answered with different **HTTP** status codes. Descriptive information can be returned as an option.

Example for a response upon a failed request:

Response

```
HTTP/1.1 400 BadRequest

{
    "reason": "10019: Missing value for a mandatory property.",
    "severity": 1,
    "errorCode": 10019
}
```

Description of parameters for response upon a failed request:

Property	Description
reason	An optional short description of why the error has occurred. This text is used as the error message's title.
hint	An optional message for the user containing troubleshooting tips.
details	Optional detailed information about the error.

severity	Optional error severity. Possible values are:
	`Success = 0,`
	`Information = 1,`
	`Warning = 2,`
	`Error = 3`

errorCode	An optional error code, as returned by d.3 server.
requestId	ID of the associated request During additional requests, the ID is transferred to other apps and serves tracing purposes when an action is processed.

Additional parameters, if the DMS object to be stored is recognized as a duplicate:

Property	Description
dmsObjectId	Contains the ID of the already existing DMS object.
_links	Contains the link relation **dmsobject** referencing the already existing DMS object.

The exclusive processing of a DMS object is required so users in Microsoft Office 365 can process an element. If processing via Office 365 is not complete, the request fails, and you receive the status code **403 Forbidden** with the corresponding error information as the response. Please repeat the request at a later time.

Format of the response for successful requests with detailed information

Released: JSON representation

In this chapter, you will learn about the output format for detailed information on successful requests. Depending on the result of the processing, the HTTP request is returned with descriptive detailed information.

Description of the parameters for the response to the successful request with detailed information:

Property	Description
responseDetails	Array with detailed information about the request

Description of the parameters of a **responseDetails** object:

Property	Description
code	Code of the type **String** defining the detailed information: • **DmsApp-Mapping-IgnoredDestinationSystemPropertyDueToUpdateMode** • **DmsApp-Mapping-IgnoredNotModifiableDestinationProperty** • **DmsApp-Mapping-MappingForSourceNotFound** • **DmsApp-Mapping-NoDestinationCategoryFoundWithDestinationProperty** • **DmsApp-Mapping-NoMappingFoundForSourceCategory** • **DmsApp-Mapping-NoMappingFoundForSourceProperty** • **DmsApp-Mapping-NoSourceProperties** • **DmsApp-Mapping-NoValuesGivenForSourceProperty** • **DmsApp-Mapping-PropertyStatusIgnored** • **DmsApp-Mapping-UnknownDestinationCategory**
title	Title that briefly states what detailed information is involved.
detail	Detailed description for detailed information.
hint	Optional further reference with possible information on actions.
data	Optional parameter that contains further data on the detailed information, e.g. for programmatic evaluation of data.

Description of the parameters of a **ResponseDetailData** object:

Property	Description
destinationItemValue	Current value of the target item.
sourceItemId	ID of the source item to which the detailed information refers (e.g. source, source property, source category).
destinationItemId	ID of the target item assigned to the source item (e.g. target property, target category).
sourceItemValue	Value of the source item to be written into the target item.

Examples of a response for a successful request with detailed information on the respective **Response-Detail** codes:

Example 1: **DmsApp-Mapping-IgnoredNotModifiableDestinationProperty**

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-
IgnoredNotModifiableDestinationProperty",
            "title": "Ignored source property",
            "detail": "The source property with the ID 'myprop1_ID' was
ignored because the mapped destination property with the ID '5' ('Name')
is not modifiable. The current value for the property is 'Value of Name'
and is not overwritten by the given value 'Value for myprop1'.",
            "hint": "Please adjust the d.3 configuration if you still want
to write the values or contact your administrator.",
            "data": {
```

```
                "sourceItemId": "myprop1_ID",
                "sourceItemValue": "Value for myprop1",
                "destinationItemId": "5",
                "destinationItemValue": "Value of Name"
            }
        }
    ]
}
```

Example 2: DmsApp-Mapping-NoMappingFoundForSourceCategory

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-NoMappingFoundForSourceCategory",
            "title": "Ignored source category",
            "detail": "The source category with the ID 'CAT1' was ignored
because no mapping to a destination category was found.",
            "hint": "Please configure a mapping for the given source
category or contact your administrator.",
            "data": {
                "sourceItemId": "CAT1"
            }
        }
    ]
}
```

Example 3: DmsApp-Mapping-IgnoredDestinationSystemPropertyDueToUpdateMode

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-
IgnoredDestinationSystemPropertyDueToUpdateMode",
            "title": "Ignored source property",
            "detail": "The source property with the ID 'PROP2'
was ignored because the mapped destination property with the ID
'property_document_number', which is a system property, cannot be changed
in update mode.",
            "data": {
                "sourceItemId": "PROP2",
                "destinationItemId": "property_document_number"
            }
        }
    ]
}
```

Example 4: DmsApp-Mapping-MappingForSourceNotFound

```
HTTP/1.1 200 OK

{
    "responseDetails": [
```

```
        {
            "code": "DmsApp-Mapping-MappingForSourceNotFound",
            "title": "Mapping configuration not found",
            "detail": "No mapping configuration was found for the source
with the ID '/myapp/sources/mysourcex'. For this reason, no mapping is
applied and any given values for properties or categories are ignored.",
            "hint": "Please configure a mapping for the given source or
contact your administrator.",
            "data": {
                "sourceItemId": "/myapp/sources/mysourcex"
            }
        }
    ]
}
```

Example 5: DmsApp-Mapping-NoDestinationCategoryFoundWithDestinationProperty

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-
NoDestinationCategoryFoundWithDestinationProperty",
            "title": "Ignored source property",
            "detail": "The source property with the ID 'myprop1_ID' was
ignored because the mapped destination property with the ID '5' was not
found in any destination category.",
            "data": {
                "sourceItemId": "myprop1_ID",
                "destinationItemId": "5"
            }
        }
    ]
}
```

Example 6: DmsApp-Mapping-NoMappingFoundForSourceProperty

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-NoMappingFoundForSourceProperty",
            "title": "Ignored source property",
            "detail": "The source property with the ID 'PROP3' was ignored
because no mapping to a destination property was found.",
            "hint": "Please configure a mapping for the given source
property or contact your administrator.",
            "data": {
                "sourceItemId": "PROP3"
            }
        }
    ]
}
```

Example 7: DmsApp-Mapping-NoSourceProperties

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-NoSourceProperties",
            "title": "No source properties given",
            "detail": "No source properties were given that could be
mapped."
        }
    ]
}
```

Example 8: DmsApp-Mapping-NoValuesGivenForSourceProperty

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-NoValuesGivenForSourceProperty",
            "title": "Ignored source property",
            "detail": "The source property with the ID 'myprop1_ID' was
ignored because no values were given.",
            "data": {
                "sourceItemId": "myprop1_ID"
            }
        }
    ]
}
```

Example 9: DmsApp-Mapping-PropertyStatusIgnored

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-PropertyStatusIgnored",
            "title": "Ignored source property",
            "detail": "The source property with the ID 'myprop1_ID' was
ignored because it is mapped to the destination property 'status' (ID:
'property_state') and other source properties were given. A status transfer
is only possible if no other properties are changed.",
            "hint": "For a status transfer please send only the property
‘status’. Change the other properties in another request.",
            "data": {
                "sourceItemId": "myprop1_ID",
                "destinationItemId": "property_state"
            }
        }
    ]
}
```

Example 10: DmsApp-Mapping-UnknownDestinationCategory

```
HTTP/1.1 200 OK

{
    "responseDetails": [
        {
            "code": "DmsApp-Mapping-UnknownDestinationCategory",
            "title": "Ignored source category",
            "detail": "The source category with the ID 'mycat1_ID'
was ignored because the mapped destination category with the id
'destcat1_ID' is unknown or the user does not have permission for the
category.",
            "data": {
                "sourceItemId": "mycat1_ID",
                "destinationItemId": "destcat1_ID"
            }
        }
    ]
}
```

Feedback using "SuccessCallback" and "Userdata"

This chapter explains how to receive feedback about the saving process. In addition, you can provide the user with various options in the storage dialog. If the saving process was successful, you can analyze the options that were selected by a user in your source system. However, you can also receive feedback without having provided options to the user.

Providing options for the user in the storage dialog

In addition to the mapping process data, you can display various options to your users in the storage dialog. The user can select one of the options. In order to make these options available, add the **userdata** array to the JSON object **sourceProperties**. For more information, see Defining the parameters for storage.

You can suggest a possible selection to the user. If you do not suggest an option, the user must select an option him/herself prior to saving the DMS object. Once saving has been successfully performed, the option selected by the user is transferred to the **(successCallbackUri**) URL.

The JSON object **sourceProperties** with the **userdata** array looks as follows:

```
{
    "properties": [
       ...
    ],
        "userdata": [{
       "key": "postProcessingOption",
       "display": "My post processing options",
       "values": [
                    {"value":"1", "display": "Action 1", "default":"false"},
                    {"value":"2", "display": "Default action 2",
"default":"true"},
                    {"value":"3", "display": "Action 3", "default":"false"}
                ]
    }]
}
```

Feedback after successfully saving

If you would like to receive feedback, specify the **successCallbackUri** parameter as soon as the DMS object was successfully saved. You will receive an **HTTP POST** request for this URL with a JSON object in the request.

The request to send a DMSApp could appear as follows:

Request

```
POST /myapp/sources/mysource/myfile/success
Origin: https://baseuri
Accept: application/hal+json
Content-Type: application/hal+json

{
    "_links": {
        "dmsobjectwithmapping": {
            "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000001234?sourceId=%2Fmyapp%2Fsources%2Fmysourc"
        }
    },
        "userdata": [
        {
            "postProcessingOption": ["2"]
        }
    ]
}
```

Description of the JSON object that is transferred for **POST**:

Property	Description
_links	Contains the **dmsobjectwithmapping** link relation. The link relation is a relative URL for the details of the stored DMS object.
userdata	Specifies the array with the results of which options a user had selected as name-value pair (key value). The **userdata** property only exists if you have provided options to the user.

Provide a corresponding HTTP status code as a response to the request of the DMSApp. Based on your status code, the storage dialog displays a corresponding success or error message. You can transmit additional information in your response as a JSON object.

Example of your response for a request:

Response

```
HTTP/1.1 400 BadRequest

{
    "reason": "My error message",
    "severity": 3
}
```

Description of the parameters for the response to the request:

Property	Description
reason	An optional short description of why the error has occurred. This text is used as the error message's title.
hint	An optional message for the user containing troubleshooting tips.
details	Optional detailed information about the error.

Property	Description

severity	Optional error severity. Possible values are:
	`Success = 0,`
	`Information = 1,`
	`Warning = 2,`
	`Error = 3`

Deleting the current version of a DMS object without user interaction

Released: JSON representation

You can delete the current version of an existing DMS object in a d.3 repository without the need for action by the user.

To delete the current version of a DMS object, you must perform the following steps:

- Determining the URL for a repository
- Determine the link relation for the existing DMS object
- Determine the link relation for deleting the current version of an existing DMS object
- Open the URL to delete the current version of the DMS object

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine the link relation for the existing DMS object

To determine a URL for an existing DMS object, perform a search and evaluate the **self** link relation for an item in the result list. The chapter Retrieving and viewing the results of a search operation provides you with more information about performing a search and the description of a result list item.

Determining the link relation for deleting the current version of the existing DMS object

To determine the URL for deleting the current version of the DMS object, evaluate the **delete** and **deleteWithReason** link relations for an item in the result list. If it is essential for a reason for deletion to be specified, for example, when deleting a version that has already been released, the **deleteWithReason** link relation is available. Otherwise, the **delete** link relation contains the URL for deletion. If both link relations are missing, please check the authorizations for deleting the version of the DMS object. If the existing DMS object is being processed, the authenticated user must be the one processing the DMS object.

Open the URL to delete the current version of the DMS object

Send an **HTTP DELETE**request for the **delete** or **deleteWithReason** link relations determined for the existing DMS object. If you want to a reason for deletion to be communicated, this must be entered in the body of the request. The reason for deletion must contain at least 3 characters and have a maximum of 80 characters.

Request

```
DELETE /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/A00000001
Origin: https://baseuri
Accept: application/hal+json
Content-Type: application/hal+json

{"reason":"Created accidentally."}
```

If the call is successful, **HTTP 200 OK** is returned: If the body is empty, the entire DMS object was deleted. If the DMS object contains more versions, the body contains additional link relations. If it

contains the **self** link relation, it means there are more versions for the DMS object and you can call up more details for the DMS object using this URL. You can delete the next version of the DMS object using the **delete** or **deleteWithReason** link relations. If these link relations are not available, it means that the user is not authorized to delete the next version.

Response

```
HTTP/1.1 200 OK

{
        "_links": {
        "self": {
                "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
A00000001"
        },
                "delete": {
                "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
A00000001"
        }
        }
}
```

If deletion of the version fails, an appropriate response is displayed. For more information, see Overview of formats for errors.

The exclusive processing of a DMS object is required so users in Microsoft Office 365 can process an element. If processing via Office 365 is not complete, the request fails, and you receive the status code **403 Forbidden** with the corresponding error information as the response. Please repeat the request at a later time.

Retrieving, saving and editing mapping

Released: JSON representation

In this chapter, you can learn how to create and manage mapping for properties and for the categories of a source system (for instance, an e-mail application) for a d.3 repository. This mapping is used for storage, for retrieving the details of an element and for retrieving and displaying the results of searches in d.3one integrations. When saving a mapping, you choose which external data (for instance, the properties of an e-mail) is assigned to which d.3 document property.

There can only be one unique mapping for each source in a source system.

The Defining a source system chapter explains how to create a source for mapping. In the Basic information about mapping chapter, you can find general information for helping you to create mappings. For useful information about write access, see Basic information about write access.

Determining the link relation for managing mapping

In the chapter Determining a repository, you can learn how to determine the URL for a repository. Then, execute an **HTTP Get** request for the REST resource for a repository as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27
Accept: application/hal+json
```

The JSON object for a repository contains the link relation **mappingconfig**.

Response

```
{
        "_links": {
                "mappingconfig": {
                        "href": "/dms/r/dee1f3d3-
eae8-5d9d-84d8-2d758c5ddc27/m/"
                }
        },
        "id": "dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27"
}
```

Storing mapping

Send an **HTTP POST** request with the mapping for your source system as the **Body** for the URL that you received in the **mappingconfig** link relation. If the mapping is stored successfully, you receive the HTTP status code **201 Created**.

Request

```
POST https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/m HTTP/1.1
Origin: https://host
Accept: application/hal+json
Content-Type: application/hal+json

{
        name: "My Source",
        sourceId: "/myapp/sources/mysource",
        mappingItems: [{
                destination: "RECH",
                source: "mycategory1_ID",
                type: 1
        },
        {
                destination: "3",
                source: "myprop1_ID",
                type: 0
        },
        {
                destination: "property_caption",
                source: "myprop2_ID",
                type: 0
        }]
}
```

The JSON object that is transferred to **POST** is described as follows:

Property	Property of a contained object	Description
name	-	Specifies the name of the mapping.
sourceId	-	Indicates the unique identifier of the source. The ID must be a relative URI. The relative URI should start with the name of the app that provides the source system in order to ensure uniqueness (e.g. **/myapp/sources/mysource**).
mappingItems	-	Specifies the array with the mapping elements. This array maps the properties and categories from the source system to the properties and categories of the d.3 repository.

Property	Property of a contained object	Description
	type	Specifies the mapping element type. Possible values are: When the value is 0, the mapping element relates to a property. When the value is 1, the mapping element relates to a category.
	source	Specifies the ID of the property in the source system.
	destination	When you map a category, **destination** specifies the ID of the category in the d.3 repository. If you map a property, **destination** contains one of the following values: • The advanced property ID as it is defined in the d.3 repository. • **property_last_modified_date** • **property_last_alteration_date** • **property_editor** • **property_remark1** • **property_remark2** • **property_remark3** • **property_remark4** • **property_owner** • **property_caption** • **property_filename** • **property_filetype** • **property_document_number** • **property_variant_number** • **property_creation_date** • **property_size** • **property_state** • **property_access_date** • **property_colorcode** When you want to specify the creation date (**property_creation_date**) of the DMS object, the specification of the property is only considered when a new DMS object is saved without any user interaction. For more information, see Storing a new DMS object without user interaction. Please note that you can only specify d.3 properties as target that are assigned to at least one d.3 category as property.

Retrieving stored mapping

To retrieve the mapping, execute an **HTTP GET** request for the URL **/dms/r/<RepositoryID>/m?sourceId=<SourceID>**. The value of the **sourceID** property indicates the unique identifier of the source (for example, **/myapp/sources/mysource**).

Request

```
GET https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/m?
sourceId=%2Fmyapp%2Fsources%2Fmysource HTTP/1.1
Accept: application/hal+json
```

As the response, you receive the object with the stored mapping.

Response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
        mappings: [{
                _links: {
                        self: {
                                href: "/dms/r/dee1f3d3-
```

```
eae8-5d9d-84d8-2d758c5ddc27/m/<MappingContainerID>"
                        }
                }
                name: "My Source",
                sourceId: "/myapp/sources/mysource",
                mappingItems: [{
                        destination: "RECH",
                        source: "mycategory1_ID",
                        type: 1
                },
                {
                        destination: "3",
                        source: "myprop1_ID",
                        type: 0
                },
                {
                        destination: "property_caption",
                        source: "myprop2_ID",
                        type: 0
                }]
        }]
}
```

The properties of the objects are the same properties as when the mappings are stored.

Deleting stored mapping

When you retrieve mapping for your source, you receive a link relation for each stored mapping (**_links.self.href**). For this URL, you can execute an **HTTP Delete** request to delete the mapping.

Request

```
DELETE https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/m/
<MappingContainerID> HTTP/1.1
Origin: https://host
Accept: application/hal+json
```

Editing stored mapping

When you retrieve mapping for your source, you receive a link relation for each stored mapping (**_links.self.href**). For this URL, you can execute an **HTTP PUT** request to update the map. The object that you transfer is the same as the object for storing mapping.

Request

```
PUT https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/m/
<MappingContainerID> HTTP/1.1
Origin: https://host
Accept: application/hal+json
Content-Type: application/hal+json

{
        name: "My Source",
        sourceId: "/myapp/sources/mysource",
        mappingItems: [{
                destination: "RECH",
                source: "mycategory1_ID",
                type: 1
```

```
        },
        {
                destination: "3",
                source: "myprop1_ID",
                type: 0
        },
        {
                destination: "property_caption",
                source: "myprop2_ID",
                type: 0
        }]
}
```

Retrieving and saving the notes of a DMS object

Released: JSON representation

This chapter explains how to retrieve and save the notes of a DMS object.

To retrieve or save the notes of a DMS object, you must perform the following steps:

- Determining the URL for a repository
- Determine and open the link relation for retrieving the details of a DMS object
- Determine the link relation for retrieving the notes of a DMS object
- Open the URL for the notes of a DMS object

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine and open the link relation for retrieving the details of a DMS object

In the chapter Retrieving and viewing the details of a DMS object, you can learn how to determine and open the URL for retrieving the details of a DMS object.

Determine the link relation for retrieving the notes of a DMS object

The JSON object for the details of a DMS object contains the link relation notes, which you can use to retrieve the notes of the DMS object.

Response

```
{
  "_links": {
         "notes":{
           "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/n"
         }
  },
  "id": "D000000123"
}
```

Retrieving the notes of a DMS object

Retrieve the notes of the DMS object with the previously determined URL as follows:

Request

```
GET https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/n HTTP/1.1
Accept: application/hal+json
```

As the response, you receive the object with the notes of the DMS object:

Response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
    "_links": {
        "self": {
            "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/n"
        }
    },
    "notes": [
        {
            "creator": {
                "id": "MargaS",
                "displayName": "Marga Schilling"
            },
            "text": "This is a sample text.",
            "created": "2019-09-03T09:09:09.453+02:00"
        }
    ],
    "description": "1 note available",
    "canAddNote": true
}
```

Property	Description
_links	**self**: self-referencing link (self-link)
notes	Specifies the array with the note elements of the DMS object.
description	Specifies how many notes there are.
canAddNote	Specifies whether notes can be added.

Structure of a note element

Property	Description
creator	Returns an object with information on the user who has written the note.
text	Contains the content of the note.
created	Specifies the unique timestamp indicating the time the note was written. The timestamp is returned in ISO format.

Structure of a creator object

Property	Description
id	Returns the unique ID of the d.3 user.
displayName	Returns the display name of the d.3 user.

Saving the notes of a DMS object

With a **HTTP POST** request for the previously determined URL you can create a note for a DMS object. If you want to create multiple notes, repeat the process. Save a note as follows:

Request

```
POST https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/n HTTP/1.1
Content-Type: application/json
Content-Length: 42

{
    "text": "This is a sample text."
}
```

Response

```
HTTP/1.1 200 OK
```

Structure of the note object

Property	Description
text	Contains the content of the note.

Retrieving and displaying the versions of a DMS object

Released: JSON representation, HTML page

You can retrieve the versions of a DMS object as JSON representations or display the versions of a DMS object.

To retrieve or display the versions of a DMS object, you must perform the following steps:

- Determine the link relation for retrieving and displaying the versions of a DMS object
- Open the URL for the versions of a DMS object

Determine the link relation for retrieving and displaying the versions of a DMS object

In the chapter Retrieving and viewing the details of a DMS object, you can learn how to determine the link relation for the versions of a DMS object.

Retrieving and displaying the versions of a DMS object (JSON representation)

Once you have the URL of the versions from the link relation, then you can retrieve the versions of a DMS object as follows:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/D000000123/v/
Accept: application/json
Accept-Language: en
```

The following JSON object will then be returned as a result:

Response

```
{
  "_links": {
    "self": {
      "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/v/"
    }
  },
```

```
  "versions": [{
      "_links": {
        "mainblobcontent": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/v/6_3/b/main/c"
        },
        "pdfblobcontent": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/v/6_3/b/p1/c"
        },
        "self": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/v/6_3"
        }
      },
      "alterationText": "",
      "caption": "3 Processing",
      "creationDate":"2019-10-25T12:22:56.000+02:00",
      "id": "6_3",
      "mimeType":"application/msword",
      "state":"Processing"
    },
    {
      "_links": {
        "mainblobcontent": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/v/3_2/b/main/c"
        },
        "pdfblobcontent": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/v/3_2/b/p1/c"
        },
        "self": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/v/3_2"
        }
      },
      "alterationText": "Reasons for changes",
      "caption": "2 Release",
      "creationDate":"2019-10-22T16:29:32.000+02:00",
      "id": "3_2",
      "mimeType":"application/msword",
      "state":"Released"
    },
    {
      "_links": {
        "mainblobcontent": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/v/1_1/b/main/c"
        },
        "pdfblobcontent": {
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2/
D000000123/v/1_1/b/p1/c"
        },
        "self": {
```

```
          "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/v/1_1"
        }
      },
      "alterationText": "Reasons for changes",
      "caption": "1 Archive",
      "creationDate":"2019-10-12T11:49:41.000+02:00",
      "id": "1_1",
      "mimeType":"application/msword",
      "state":"Archived"
    }
  ]
}
```

Property	Description
_links	Contains the link relations: **self**: Self-link
versions	Specifies the array with the versions.

Structure of a version

Property	Description
_links	Contains the link relations for the version of a DMS object. **mainblobcontent**: Relative download URL for the main document of the DMS object version. **pdfblobcontent:** Relative download URL for the dependent PDF document of the version of the DMS object. You will only receive this URL if a dependent PDF document has been created for the DMS object. **self**: Self-link.
alterationText	Specifies the alteration text for the release of the version.
caption	Specifies the display name of the version. This value is output for the specific language by evaluating the **Accept-Language** HTTP header.
creationDate	Specifies the creation date of the version.
id	Specifies the ID of the version.
mimeType	Specifies the type of the main document.
state	Specifies the status of the version. The values available are **Processing**, **Verification**, **Released** and **Archived**.

Retrieving and displaying the versions of a DMS object (HTML page)

If you want to call the HTML view of the versions, determine the link relation for the versions of a DMS object. Enter the URL in the browser to view the HTML page. This HTML page contains the list of the versions of a DMS object.

Example:

Request

```
GET /dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/D000000123/v/
Accept: text/html
```

Linking DMS objects

Released: JSON representation

In this chapter, you can learn how to link DMS objects with other DMS objects hierarchically independent from their type.

To link DMS objects to each other, perform the following steps:

- Determining the URL for a repository
- Determine and open the link relation for retrieving the details of a DMS object
- Determine the link relation for linking DMS objects
- Linking DMS objects

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine and open the link relation for retrieving the details of a DMS object

In the chapter Retrieving and viewing the details of a DMS object, you can learn how to determine and open the URL for retrieving the details of a DMS object.

Determine the link relation for linking DMS objects

The JSON object for the details of a DMS object contains the link relation **linkDmsObjects**, which you can use to link DMS objects.

Response

```
{
    "_links": {
        "linkDmsObjects":{
            "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/children"
        }
    },
    "id": "D000000123"
}
```

Linking DMS objects

To link a DMS object to the transferred DMS objects, execute an **HTTP POST** request with the list of IDs of the DMS objects to be linked as the **body** to the URL determined above as follows:

Request

```
POST https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/D000000123/
children HTTP/1.1
Accept: application/hal+json

Content-Type: application/hal+json

{
    "dmsObjectIds": [
        "D000000089",
        "D000000127",
        "D000004567",
    ]
}
```

The JSON object that is transferred to **POST** is described as follows:

Property	Description
dmsObjectIds	Specifies the array with the IDs of the DMS objects (of the **String** type) that are to be linked to the DMS object.

As the response, you receive the HTTP status code 200 (OK) if the link was successful. If the link was not successful or was successful only for individual DMS objects, you receive the HTTP status code 207 (multi-status) and a list with detailed information about the individual link operations.

Example of a response to a partly failed request:

Response

```
HTTP/1.1 207 Multi-Status
Content-Type: application/hal+json

{
 "requestId": "XyErwIKPhyGaMg9dxcGksgAAA@A",
 "linkDocumentErrorPageModels": [
   {
     "dmsObjectId": "D000000089",
     "errorPageModel": {
       "reason": "These documents are already linked to each other!
[0000071] ",
       "severity": 1,
       "errorCode": 71
     }
   },
   {
     "dmsObjectId": "D000000127",
     "errorPageModel": {
       "reason": "These documents are already linked to each other!
[0000071] ",
       "severity": 1,
       "errorCode": 71
     }
   },
   {
     "dmsObjectId": "D000004567",
     "errorPageModel": {
       "reason": "These documents are already linked to each other!
[0000071] ",
       "severity": 1,
       "errorCode": 71
     }
   }
 ]
}
```

Description of parameters for response upon a failed request:

Property	Description
requestId	ID of the associated request During additional requests, the ID is transferred to other apps and serves tracing purposes when an action is processed.
linkDocumentErrorPageModels	An array with error messages for a link operation.

Structure of a response object for a link operation

Property	Description
dmsObjectId	The ID of the DMS object to be linked.

Property	Description
errorPageModel	An object with a description of whether the link was successful.

Response format for errors

Property	Description
errorCode	An optional error code, as returned by d.3 server.
reason	An optional short description of why the error has occurred. This text is used as the error message's title.

severity	Optional error severity. Possible values are:
	`Success = 0,`
	`Information = 1,`
	`Warning = 2,`
	`Error = 3`

Removing a DMS object link

This function is available only for on-premises installations.

Released: JSON representation

This chapter explains how to remove the link between two DMS objects. You can only remove links of DMS objects individually.

To remove the link between two DMS objects, you must perform the following steps:

- Determining the URL for a repository
- Determine and open the link relation for retrieving the details of a DMS object
- Determine the link relation for removing the link between two DMS objects
- Open the URL for removing the link between two DMS objects

Determining the URL for a repository

In the chapter Determining a repository, you can learn how to determine the URL for a repository.

Determine and open the link relation for retrieving the details of a DMS object

In the chapter Retrieving and viewing the details of a DMS object, you can learn how to determine and open the URL for retrieving the details of a DMS object.

Determine the link relation for removing the link between two DMS objects

The JSON object for the details of a DMS object contains the link relation **unlinkDmsObject** with a placeholder for the ID of the parent DMS object for which you want to remove the link to the dossier.

Response

```
{
    "_links": {
        "unlinkDmsObject": {
            "href": "/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
{parentDmsObjectId}/children/D000004567"
        }
    },
    "id": "D000000123"
}
```

Open the URL for removing the link between two DMS objects

Execute an **HTTP DELETE** request with the ID of the DMS object as a parameter in the URL determined above as follows to remove the link between the DMS object and the dossier:

Request

```
DELETE https://host/dms/r/dee1f3d3-eae8-5d9d-84d8-2d758c5ddc27/o2m/
D000000123/children/D000004567 HTTP/1.1
Accept: application/hal+json
```

As the response, you receive the **HTTP** status code 200 (OK) if the removal of the link was successful. If the removal of the link was unsuccessful, you receive the **HTTP** status code 400 (bad request) and detailed information about the error.

Example for a response upon a failed request:

Response

```
HTTP/1.1 400 Bad Request
Content-Type: application/hal+json

{
  "reason": "These two documents are not linked with each other. [0000073]
",
  "severity": 1,
  "errorCode": 73,
  "requestId": "XyEWT4KPhyGaMg9dxcGNiAAAA@c"
}
```

If you want to remove multiple DMS objects, repeat this step.

Description of parameters for response upon a failed request:

Response format for errors

Property	Description
reason	An optional short description of why the error has occurred. This text is used as the error message's title.

severity	Optional error severity. Possible values are:
	`Success = 0,`
	`Information = 1,`
	`Warning = 2,`
	`Error = 3`

errorCode	An optional error code, as returned by d.3 server.
requestId	ID of the associated request During additional requests, the ID is transferred to other apps and serves tracing purposes when an action is processed.

1.4.4. Adding your own functions (DMSApp)

In this section, you can learn how to add your own functions to the DMSApp.

- Using extension points

Using extension points

Extension points allow you to enhance the DMSApp with your own functions. The DMSApp provides extension points at the following positions:

- Replacing the content of the "View" perspective in the detail view
- Adding context actions to lists
- Adding context actions to detail view

There are two ways you can extend the extension points in the DMSApp:

- Providing via URL
- Creating via the API in the DMSApp

Preparing your app

To ensure that you can find extensions for context actions or for displaying documents from the DMSApp, your app must provide this user interface. To find apps that provide extensions, DMSApp uses the concept of an **HTTP GET** request for the root resource (**systemBaseUri** path with the app name) for the apps. All apps registered to the d.ecs http gateway will be requested. Make sure that the administrator did not actively exclude your app.

Providing the URL for the extension points

The root resources for these apps are requested cyclically. The response of an app is then checked to determine whether a link relation with the name **dmsobjectextensions** is included. This link relation serves as the signal that the app provides extensions for DMSApp. DMSApp executes an **HTTP GET** to the specified link and waits for a standardized JSON object with the HTTP status code 200 from the responding app.

Request

```
GET https://host/myapp/ HTTP/1.1
Accept: application/hal+json
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
    "_links" : {
        "dmsobjectextensions" : {
            "href" : "/myapp/dmsobjectextensions"
        }
    }
}
```

> **Note**
> The root resources are queried anonymously (without authentication) and this process is executed as an asynchronous background process within DMSApp.

Creating via the API in the DMSApp

With a **HTTP POST** request for the URL **/dms/extensions** you can create an extension in the DMSApp. If you want to create multiple extensions, repeat the process. The request must be made by a user with administration rights only. If the request response is successful, the extension is stored in the DMSApp and you get a **location URL** in the response header. The structure of the standardized JSON object is described in the following pages.

Request

```
POST /dms/extensions
Origin: https://baseuri
Content-Type: application/json
Content-Length: 619
```

```
{
    "id": "myapp.openExternalApp",
    "activationConditions": [{
            "propertyId": "repository.id",
            "operator": "or",
            "values": ["e632f767-5cfa-538d-ab55-6756c36a74c9"]
        }
    ],
    "captions": [{
            "culture": "de",
            "caption": "Externe Applikation öffnen"
        }, {
            "culture": "en",
            "caption": "Open external application"
        }
    ],
    "context": "DmsObjectDetailsContextAction",
    "uriTemplate": "/myapp/dosomething?id={dmsobject.property_document_id}",
    "iconUri": "/myapp/images/goto.svg"
}
```

Response

```
HTTP/1.1 201 Created
Location: /dms/extensions/dmsapp-custom-myapp.openExternalApp
```

Via the **location URL**, a user with administration rights can also delete the extension in the DMSApp. To do this, an **HTTP DELETE request**must be sent to the **Location URL**value.

Request

```
DELETE /dms/extensions/dmsapp-custom-myapp.openExternalApp
Origin: https://baseuri
```

Response

```
HTTP/1.1 200 OK
```

Displaying extension points

Released: HTML page

You can display the extension points registered with DMSApp. Simply enter the URL **/dms/extensions** in the browser.

Request

```
GET /dms/extensions
Accept: text/html
```

Adding context actions to lists

Released: Extension point

If you add context actions to lists with items (e.g. documents and dossiers), they are displayed in the following sections:

- Results in the **Search** feature
- Content of a dossier

- Your lists of favorites and watched items in the **Personal Area** feature

If the user clicks a context action in a list, the list always first switches to selection mode. In selection mode, the user can select the desired items and execute the context action.

The app that you want to provide such extensions must return an HTTP response in JSON format under the **dmsobjectextensions** link relation. This response must provide the following information for each context action:

- Extension point context: **DmsObjectListContextAction**
- Activation condition for displaying the context action.
- Display name for the context action in the available translations.
- Link to the context action icon.
- The action to be performed once the user has completed the selection. Specify a relative link that is called from DMSApp using **HTTP GET**.

Any context action can also be created via **HTTP POST request**for the URL **/dms/extensions** in the DMSApp. The request must be made by a user with administration rights only. If the request response is successful, the extension is stored in the DMSApp and you get a **location URL**in the response header. Via the **location URL**, a user with administration rights can also delete the extension in the DMSApp. To do this, an **HTTP DELETE request**must be sent to the **Location URL**value.

Example

The example shows how you arrange the HTTP response of the **dmsobjectextensions** link relation to add a context action using the **DmsObjectListContextAction** extension point. You define different properties for each context action.

```
{
        "extensions": [
        {
                "id": "myapp.exportList",
                "activationConditions": [{
                        "propertyId": "repository.id",
                        "operator": "or",
                        "values": ["e632f767-5cfa-538d-ab55-6756c36a74c9"]
                }],
                "captions": [{
                        "culture": "de",
                        "caption": "Exportieren"
                },
                {
                        "culture": "en",
                        "caption": "Export"
                },
        {
            "culture": "en-GB",
            "caption": "Exportin Great Britain"
        }],
                "descriptions": [{
                        "culture": "de",
                        "description": "Liste als CSV exportieren"
                },
                {
                        "culture": "en",
                        "description": "Export list as CSV"
                },
```

```
        {
            "culture": "en-GB",
            "description": "Export list as CSV in Great Britain"
        }],
            "context": "DmsObjectListContextAction",
                "uriTemplate": "/myapp/dosomething?url={dmsobjectlist.url}",
                "iconUri": "/myapp/images/export-list.svg",
        "target": "dapi_navigate"
        }]
}
```

Property	Properties of a contained object	Description
id	-	Specifies the unique technical name used to differentiate the extension from other extensions.
activationConditions		For each extension, the application notifies you of which activation conditions are used to display the context action. These activation conditions are reported by the application in advance. If the activation conditions were not reported in advance, DMSApp would have to query other apps with a network request at a later time when the user is viewing an item. The waiting time for the user would then increase significantly if an app only responds to this request with a delay. A context action is displayed if all the sub-conditions are met. If the list of activation conditions does not contain an entry, the extension is generally active. You specify the activation conditions as an array.
	propertyId	Specifies the ID of the property that is tested for the activation condition. The available values are described in more detail below.
	operator	The operator specifies how a sub-condition is evaluated. The following operator is available: **or**: An **or** condition is fulfilled if the current value of the property corresponds to one of the values. Capitalization is not taken into account. **notOr**: A **notOr** condition is fulfilled if the current value of the property does not correspond to any of the values. Capitalization is not taken into account.
	values	Specifies the values in the form of an array that is compared with the value of the **propertyId** property.

Property	Properties of a contained object	Description
captions		Each context action notifies you of the name under which it is displayed. You can also include different languages. In this case, the language-dependent names are specified as an array. To ensure the language packages are fully compatible, specify the languages for the following **Cultures** names: • cs (Czech) • da (Danish) • de (German) • en (English) • es (Spanish) • fr (French) • hr (Croatian) • it (Italian) • nl (Dutch) • pl (Polish) • sr (Serbian) • sk (Slovakian) • zh-CN (Chinese (simplified)) If the user requests a language for which the extension app does not have any specification for the localized (language-specific) name, an alternative language is determined for the view based on the following rules: • If the language ID with the regional code (e.g. en-GB) is not found, the system checks whether the higher-level language without the regional code is available (en). In the example above, "Export List" is displayed for a request of the "en-GB" language because "en" is used as an alternative. • If the higher-level language is also unavailable, English "en" without the regional code is always used as the alternative. • If "en" is also not specified as the alternative, the first specified language is displayed as the alternative.
	culture	Specifies the language ID for which the name of the context action is defined. The specification includes the language code (e.g. en) and optionally an additional regional code (e.g. en-GB).
	caption	Specifies the language-dependent name of the context action.
descriptions		(Optional) Each context action can indicate which description is displayed for that context action. This description is displayed as a tooltip for the context action. Different languages are specified as in the captions property.
	culture	Specifies the language ID for which the description of the context action is defined. The specification includes the language code (e.g. en) and optionally an additional regional code (e.g. en-GB).
	description	Specifies the language-dependent description of the context action.
context	-	Specifies the extension point to which you want to add the context action. Specify the following value for context actions for the detail view: **DmsObjectListContextAction**
uriTemplate	-	In the **uriTemplate** property, you define the URL that you want to be called when the user has completed the selection. You can define placeholders to receive more information about the current context. The placeholders are replaced by the actual properties when the extension is called. The properties are transferred to the URI with URL encoding. You can find the available placeholders below. **Note** Using a placeholder in the host part of the URL is not permitted for security reasons and leads to an error.
iconUri	-	Specifies the link to the icon that is displayed for the context action. The icon file must be available in SVG format. The color of the SVG's fill must not be specified, so that the fill color can be adjusted by using the theming. If you want to use a custom SVG file for a symbol, we recommend saving the SVG file to the installation directory **d.3one/dms/Client/Custom**.

Property	Properties of a contained object	Description
target	-	(Optional) Specifies where the content of the context action should be displayed. Possible values: • **dapi_navigate** (default value): Navigates to a new main resource with the content within the Shell app. • **dapi_inner_supply**: Opens an Inner Supply (display area below the app bar) with the content. • **dapi_outer_supply**: Opens an outer supply (display area above the app bar) with the content. • **_blank**: Opens a new tab in the browser with the content.

When defining activation conditions for context actions for lists, use the following values for the **propertyId** property:

Condition context	propertyId	Description
Repository	**repository.id**	ID of the repository as specified in the d.ecs repo app. You can find the repository ID in the detail section of the **d.3 repositories** feature (**https://<Base address>/repo/repositories/**). You can obtain this ID either by using the URL or clicking **All Programs > d.velop > d.3one > Repository Configuration** in the Start menu. In the chapter Determining a repository, you can learn how to determine the ID for a repository with the program.
User	**user.d3.group_id**	Activation condition: ID of a d.3 user group (maximum of eight characters) of which the user that is currently logged in is a member.
	user.idp.group_id	Activation condition: GUID of an identity provider app user group of which the user that is currently logged in is a member.

For a context action of the **DmsObjectListContextAction** type, you can use the following properties as placeholders in the **uriTemplate** property:

Property	Description
dmsobjectlist.url	A URL-encoded URL under which the list of items selected by the user (documents and dossiers) can be queried. As an HTTP response, you receive this URL as a **selectionList** JSON array that returns the following information for each item: • **_links.self.href**: Relative URL for the item that you can use to retrieve more detailed information about the item. • **id**: Document ID of the item. • **caption**: Title for the item. Example: **selectionList** `{ "selectionList": [{ "_links": { "self": { "href": "/dms/r/e632f767-5cfa-538d-ab55-6756c36a74c9/o2/A0000000001" } }, "id": "A0000000001", "caption": "Title of the item" }, ... ] }`

Property	Description
repository.id	ID of the repository as specified in the d.ecs repo app. You can find the repository ID in the detail section of the **d.3 repositories** feature (**https://<Base address>/repo/repositories/**). You can obtain this ID either by using the URL or clicking **All Programs > d.velop > d.3one > Repository Configuration** in the Start menu. In the chapter Determining a repository, you can learn how to determine the ID for a repository with the program.
user.d3.group_id	Placeholder **uriTemplate**: List of IDs of the d.3 user groups (each a maximum of eight characters) of which the user currently logged in is a member as an array in JSON format.
user.idp.group_id	Placeholder **uriTemplate**: List of GUIDs of the identity provider app user groups of which the user currently logged in is a member as an array in JSON format.

Adding context actions to detail view

Released: Extension point

You can add context actions to the detail view for an item (DMS object, document or dossier). These context actions are also summarized in the menu for context actions (three stacked dots) for an item. The app that you want to provide such an extension must return an HTTP response in JSON format under the link relation **dmsobjectextensions**. The response must provide the following information for each context action:

- Extension point context: **DmsObjectDetailsContextAction**
- Activation condition for displaying the context action.
- Display name for the context action in the available translations.
- Link to the context action icon.
- The action to be performed once the user has clicked on the context action. Specify a relative link that is called from DMSApp using **HTTP GET**.

Any context action can also be created via **HTTP POST request**for the URL **/dms/extensions** in the DMSApp. The request must be made by a user with administration rights only. If the request response is successful, the extension is stored in the DMSApp and you get a **location URL**in the response header. Via the **location URL**, a user with administration rights can also delete the extension in the DMSApp. To do this, an **HTTP DELETE request**must be sent to the **Location URL**value.

Example

The example shows how you arrange the HTTP response of the **dmsobjectextensions** link relation to add a context action using the **DmsObjectDetailsContextAction** extension point. You define different properties for each context action.

```
{
        "extensions": [
        {
                "id": "myapp.openExternalApp",
                "activationConditions": [{
                        "propertyId": "repository.id",
                        "operator": "or",
                        "values": ["e632f767-5cfa-538d-ab55-6756c36a74c9"]
                }],
                "captions": [{
                        "culture": "de",
                        "caption": "Externe Applikation öffnen"
                },
                {
                        "culture": "en",
                        "caption": "Open external application"
                }],
                "context": "DmsObjectDetailsContextAction",
```

```
                "uriTemplate": "/myapp/dosomething?
id={dmsobject.property_document_id}",
                "iconUri": "/myapp/images/goto.svg",
        "target": "dapi_navigate"
        }]
}
```

Property	Properties of a contained object	Description
id	-	Specifies the unique technical name used to differentiate the extension from other extensions.
activationConditions		For each extension, the application notifies you of which activation conditions are used to display the context action. These activation conditions are reported by the application in advance. If the activation conditions were not reported in advance, DMSApp would have to query other apps with a network request at a later time when the user is viewing a document. The waiting time for the user would then increase significantly if an app only responds to this request with a delay. A context action is displayed if all the sub-conditions are met. If the list of activation conditions does not contain an entry, the extension is generally active. You specify the activation conditions as an array.
	propertyId	Specifies the ID of the property that is tested for the activation condition. The available values are described in more detail below.
	operator	The operator specifies how a sub-condition is evaluated. The following operator is available: **or**: An **or** condition is fulfilled if the current value of the property corresponds to one of the values. Capitalization is not taken into account. **notOr**: A **notOr** condition is fulfilled if the current value of the property does not correspond to any of the values. Capitalization is not taken into account.
	values	Specifies the values in the form of an array that is compared with the value of the **propertyId** property.
captions		Each context action notifies you of the name under which it is displayed. You can also include different languages. In this case, the language-dependent names are specified as an array. To ensure the language packages are fully compatible, specify the languages for the following **Cultures** names: • cs (Czech) • da (Danish) • de (German) • en (English) • es (Spanish) • fr (French) • hr (Croatian) • it (Italian) • nl (Dutch) • pl (Polish) • sr (Serbian) • sk (Slovakian) • zh-CN (Chinese (simplified)) If the user requests a language for which the extension app does not have any specification for the localized (language-specific) name, an alternative language is determined for the view based on the following rules: • If the language identifier with the regional code (e.g. en-GB) is not found directly, the system checks whether the higher-level language (language without regional code) is available (en). In the example above, "Export List" is displayed for a request of the "en-GB" language because "en" is used as an alternative. • If the higher-level language is also unavailable, English "en" without the regional code is always used as the alternative. • If "en" is also not specified as the alternative, the first specified language is displayed as the alternative.

Property	Properties of a contained object	Description
	culture	Specifies the language ID for which the name of the context action is defined. The specification includes the language code (e.g. en) and optionally an additional regional code (e.g. en-GB).
	caption	Specifies the language-dependent name of the context action.
context	-	Specifies the extension point to which you want to add the context action. Specify the following value for context actions for the detail view: **DmsObjectDetailsContextAction**
uriTemplate	-	In the **uriTemplate** property, you define the URL that is to be called when the user clicks on the context action. You can define placeholders to receive more information about the current context. The placeholders are replaced by the actual properties when the extension is called. The properties are transferred to the URI with URL encoding. You can find the available placeholders below.
iconUri	-	Specifies the link to the icon that is displayed for the context action. The icon file must be available in SVG format. The color of the SVG's fill must not be specified, so that the fill color can be adjusted by using the theming. If you want to use a custom SVG file for a symbol, we recommend saving the SVG file to the installation directory **d.3one/dms/Client/Custom**.
target	-	(Optional) Specifies where the content of the context action should be displayed. Possible values: • **dapi_navigate** (default value): Navigates to a new main resource with the content within the Shell app. • **dapi_inner_supply**: Opens an Inner Supply (display area below the app bar) with the content. • **dapi_outer_supply**: Opens an outer supply (display area above the app bar) with the content. • **_blank**: Opens a new tab in the browser with the content.

You can use the following values when defining context actions for the detail view in the following areas:

- Defining activation conditions for the **propertyId** property
- Defining placeholders in the **uriTemplate** property

Topic	Value	Description
Repository	**repository.id**	ID of the repository as specified in the d.ecs repo app. You can find the repository ID in the detail section of the **d.3 repositories** feature (**https://<Base address>/repo/repositories/**). You can obtain this ID either by using the URL or clicking **All Programs > d.velop > d.3one > Repository Configuration** in the Start menu.
User	**user.d3.group_id**	Activation condition: ID of a d.3 user group (maximum of eight characters) of which the user that is currently logged in is a member. Placeholder **uriTemplate**: List of IDs of the d.3 user groups (each a maximum of eight characters) of which the user currently logged in is a member as an array in JSON format.
	user.idp.group_id	Activation condition: GUID of an identity provider app user group of which the user that is currently logged in is a member. Placeholder **uriTemplate**: List of GUIDs of the identity provider app user groups of which the user currently logged in is a member as an array in JSON format.
Property for the item	**dmsobject.property_editor**	Editor of the item.
	dmsobject.property_owner	Owner of the item.
	dmsobject.property_filename	File name for the item.
	dmsobject.property_filetype	File type for the item.

Topic	Value	Description
	dmsobject.property_document_number	Document number of the item.
	dmsobject.property_creation_date	Creation date of the item.
	dmsobject.property_size	File size of the item.
	dmsobject.property_state	Document status of the item. Possible values: • **Archived**: The element has the **Archive** status. • **VerificationInProgress**: The element has the **Verification** status. • **Processing**: The element has the **Processing** status. • **Released**: The element has the **Released** status.
	dmsobject.property_variant_number	Variant number of the item.
	dmsobject.property_access_date	Access date for the item.
	dmsobject.property_remark	Remarks about the item.
	dmsobject.property_last_alteration_date	Alteration date of the item.
	dmsobject.property_caption	Title for the item.
	dmsobject.property_category	Item category ID.
	dmsobject.property_category_uuid	UUID of the item category.
	dmsobject.property_colorcode	Color marking for the item.
	dmsobject.property_document_class	Activation condition: A document class abbreviation for the item. Placeholder **uriTemplate**: List of document class IDs for the item as an array in JSON format.
	dmsobject.property_document_id	Document ID of the item.
	dmsobject.property_display_version_id	Unique ID for the display version for the DMSApp.
	dmsobject.type	The item type. Possible values: • **Document**: The item is an item of the "Document" type. • **Dossier**: The item is an item of the "Dossier" type.
	dmsobject.<NUMBER|UUID>	For the **<NUMBER|UUID>** placeholder, enter the advanced property ID or UUID as it is defined in the d.3 repository. If the d.3 property is a multi-value property, the placeholder will be replaced by the first or the first filled value of the property (depending on the d.3 repository configuration). If there are multiple values for the multi-value property, three dots (...) are added to the value returned.
	dmsobject.fieldposition.<NUMBER>	For the **<NUMBER>** placeholder, enter the database position of the advanced property. **Note** We strongly recommend using the advanced property ID (**dmsobject.<NUMBER>**). Use the database position only in exceptional cases when the advanced property ID is not available to you or can only be determined in a very time-consuming process. If the d.3 property is a multi-value property, the placeholder will be replaced by the first or the first filled value of the property (depending on the d.3 repository configuration). If there are multiple values for the multi-value property, three dots (...) are added to the value returned.

Topic	Value	Description
	dmsobject.self_url_relative	Relative URL of the item.
Original file for the item	**dmsobject.mainblob.content_type**	MIME type of the original file (e.g. **text\plain** or **image\jpeg**).
	dmsobject.mainblob.content_url	(**Obsolete**) Absolute URL of the original file. If the user does not have the authorization to export the document, then this parameter is empty.
	dmsobject.mainblob.content_url_relative	Relative URL of the original file. If the user does not have the authorization to export the document, then this parameter is empty.
	dmsobject.mainblob.id	ID of the original file if the user has the right to export the original file.
Dependent files for the item	**dmsobject.dependentblobs**	List of the IDs of dependent files if the user has the right to export the dependent file.
	dmsobject.dependentblob.ID.content_url	(**Obsolete**) Absolute URL of the dependent file with the ID from **dmsobject.dependentblobs**.
	dmsobject.dependentblob.ID.content_url_relative	Relative URL of the dependent file with the ID from **dmsobject.dependentblobs**.

Replacing the content of the "View" perspective in the detail view

Released: Extension point

You can replace the content of the **View** perspective for the detail view for an item (e.g. a document or dossier) so that you can display your own preview under certain conditions.

The app that you want to provide such an extension must return an HTTP response in JSON format under the **dmsobjectextensions** link relation. This response must provide the following information for each extension point of the **View** perspective:

- Extension point context: **DmsObjectDetailsPreview**
- Activation condition for displaying your preview.
- The URL for the preview that you want to display when the user opens the **View** perspective in the detail view and meets your predefined activation conditions. Specify a relative link that is called from DMSApp using **HTTP GET**.

Any context action can also be created via **HTTP POST request**for the URL **/dms/extensions** in the DMSApp. The request must be made by a user with administration rights only. If the request response is successful, the extension is stored in the DMSApp and you get a **location URL**in the response header. Via the **location URL**, a user with administration rights can also delete the extension in the DMSApp. To do this, an **HTTP DELETE request**must be sent to the **Location URL**value.

Example

The example shows how you arrange the HTTP request to add an extension for the **DmsObjectDetailsPreview** extension point. You define different properties for each extension.

```
       {
       "id": "myapp.viewer",
       "activationConditions": [{
           "propertyId": "dmsobject.mainblob.content_type",
           "operator": "or",
           "values": [
                               "text/plain",
               "application/pdf"
                       ]
       }],
       "context": "DmsObjectDetailsPreview",
       "uriTemplate": "/myapp/preview?
```

```
layer0={dmsobject.mainblob.content_url}"
        }
```

Property	Properties of a contained object	Description
id	-	Specifies the unique technical name used to differentiate the extension from other extensions.
activationConditions		For each extension, the application notifies you of which activation conditions are used to display the extension of the **View** perspective. These activation conditions are reported by the application in advance. If the activation conditions were not reported in advance, DMSApp would have to query other apps with a network request at a later time when the user is viewing an item. The waiting time for the user would then increase significantly if an app only responds to this request with a delay. An extension is displayed if all the sub-conditions are met. If the list of activation conditions does not contain an entry, the extension is generally active. You specify the activation conditions as an array.
	propertyId	Specifies the ID of the property that is tested for the activation condition. The available values are described in more detail below.
	operator	The operator specifies how a sub-condition is evaluated. The following operator is available: **or**: An **or** condition is fulfilled if the current value of the property corresponds to one of the values. Capitalization is not taken into account. **notOr**: A **notOr** condition is fulfilled if the current value of the property does not correspond to any of the values. Capitalization is not taken into account.
	values	Specifies the values in the form of an array that is compared with the value of the **propertyId** property.
context	-	Specifies the extension point to which you want to add the extension. Enter the following value for extensions of the **View** perspective in the detail view: **DmsObjectDetailsPreview**
uriTemplate	-	In the **uriTemplate** property, you define the URL that you want to be called when the user opens the **View** perspective in the detail view. You can define placeholders to receive more information about the current context. The placeholders are replaced by the actual properties when the extension is called. The properties are transferred to the URI with URL encoding. You can find the available placeholders below. **Note** Using a placeholder in the host part of the URL is not permitted for security reasons and leads to an error.

While defining extensions that replace the content of the **View** perspective, you can use the following values in the following areas:

- Defining activation conditions for the **propertyId** property
- Defining placeholders in the **uriTemplate** property

Topic	Value	Description
Repository	**repository.id**	ID of the repository as specified in the d.ecs repo app. You can find the repository ID in the detail section of the **d.3 repositories** feature (**https://<Base address>/repo/repositories/**). You can obtain this ID either by using the URL or clicking **All Programs > d.velop > d.3one > Repository Configuration** in the Start menu. In the chapter Determining a repository, you can learn how to determine the ID for a repository with the program.

Topic	Value	Description
User	**user.d3.group_id**	Activation condition: ID of a d.3 user group (maximum of eight characters) of which the user that is currently logged in is a member. Placeholder **uriTemplate**: List of IDs of the d.3 user groups (each a maximum of eight characters) of which the user currently logged in is a member as an array in JSON format.
	user.idp.group_id	Activation condition: GUID of an identity provider app user group of which the user that is currently logged in is a member. Placeholder **uriTemplate**: List of GUIDs of the identity provider app user groups of which the user currently logged in is a member as an array in JSON format.
Property for the item	**dmsobject.property_editor**	Editor of the item.
	dmsobject.property_owner	Owner of the item.
	dmsobject.property_filename	File name for the item.
	dmsobject.property_filetype	File type for the item.
	dmsobject.property_document_number	Document number of the item.
	dmsobject.property_creation_date	Creation date of the item.
	dmsobject.property_size	File size of the item.
	dmsobject.property_state	Document status of the item. Possible values: • **Archived**: The element has the **Archive** status. • **VerificationInProgress**: The element has the **Verification** status. • **Processing**: The element has the **Processing** status. • **Released**: The element has the **Released** status.
	dmsobject.property_variant_number	Variant number of the item.
	dmsobject.property_access_date	Access date for the item.
	dmsobject.property_remark	Remarks about the item.
	dmsobject.property_last_alteration_date	Alteration date of the item.
	dmsobject.property_caption	Title for the item.
	dmsobject.property_category	Item category ID.
	dmsobject.property_category_uuid	UUID of the item category.
	dmsobject.property_colorcode	Color marking for the item.
	dmsobject.property_document_class	Activation condition: A document class ID for the item. Placeholder **uriTemplate**: List of document class IDs for the item as an array in JSON format.
	dmsobject.property_document_id	Document ID of the item.
	dmsobject.property_display_version_id	Unique ID for the display version for the DMSApp.
	dmsobject.property_version_id	Unique ID for the current version for the DMSApp.

Topic	Value	Description
	dmsobject.type	The item type. Possible values: • **Document**: The item is an item of the "Document" type. • **Dossier**: The item is an item of the "Dossier" type.
	dmsobject.<NUMBER|UUID>	For the **<NUMBER|UUID>** placeholder, enter the advanced property ID or UUID as it is defined in the d.3 repository. If the d.3 property is a multi-value property, the placeholder will be replaced by the first or the first filled value of the property (depending on the d.3 repository configuration). If there are multiple values for the multi-value property, three dots (...) are added to the value returned.
	dmsobject.fieldposition.<NUMBER>	For the **<NUMBER>** placeholder, enter the database position of the advanced property. **Note** We strongly recommend using the advanced property ID (**dmsobject.<NUMBER>**). Use the database position only in exceptional cases when the advanced property ID is not available to you or can only be determined in a very time-consuming process. If the d.3 property is a multi-value property, the placeholder will be replaced by the first or the first filled value of the property (depending on the d.3 repository configuration). If there are multiple values for the multi-value property, three dots (...) are added to the value returned.
	dmsobject.self_url_relative	Relative URL of the item.
Original file for the item	**dmsobject.mainblob.content_type**	MIME type of the original file (e.g. **text\plain** or **image\jpeg**).
	dmsobject.mainblob.content_url	(**Obsolete**) Absolute URL of the original file. If the user does not have the authorization to export the document, then this parameter is empty.
	dmsobject.mainblob.content_url_relative	Relative URL of the original file. If the user does not have the authorization to export the document, then this parameter is empty.
	dmsobject.mainblob.id	ID of the original file if the user has the right to export the original file.
Dependent files for the item	**dmsobject.dependentblobs**	List of the IDs of dependent files if the user has the right to export the dependent file.
	dmsobject.dependentblob.ID.content_url	(**Obsolete**) Absolute URL of the dependent file with the ID from **dmsobject.dependentblobs**.
	dmsobject.dependentblob.ID.content_url_relative	Relative URL of the dependent file with the ID from **dmsobject.dependentblobs**.