ecoDMS Version 18.09 (apu)

ecoDMS API

Imprint

ecoDMS Version:	18.09 (apu)
Manual Date:	15th June 2021
Type:	API REST Service
Language:	EN

Author / Originator:	ecoDMS GmbH
©2021	Dresdener Straße 1
	52068 Aachen
	Germany
Website:	www.ecodms.de
Email:	info@ecodms.de
Phone:	0049 241 47572 01
Company's registered office:	Aachen
Registry court:	Amtsgericht Aachen 19201
Management:	Dipl.-Betrw. Michael Schmitz
	Helge Lühmann

Important Information

Contents

1 Introduction

- To work with the ecoDMS API, you need so-called API connects. You can purchase them as an option with your existing ecoDMS license number.
- The costs for the ecoDMS API depend on the number of required monthly API connects (scaled prices)
- 1 API connect = 1 monthly upload or download via the ecoDMS API
- For more information about the license model, refer to www.ecodms.de/index.php/de/ecodms-api/lizenzmodell

The ecoDMS API is intended for developers. It allows you to connect all third-party systems such as CRM software, inventory management systems and much more. Use the interface to connect with the base functions of ecoDMS Server, such as archiving, classification, or downloading. The individual functions are called via the REST web services. Each function thus has a unique address expressed in form of a URL and which can be used, among others, in web browsers.

If, for example, you want to download a document with the ID=5 from ecoDMS, use the command

```
'/document/5'
```

, where $'/document'$ is the REST call of the function and$'5'$ is the input parameter that is interpreted as Document ID=5.

Please note: ecoDMS API REST Service only supports UTF-8 coding. This means that before you call any REST function, you must ensure that the HTTP request is coded in UTF-8.

1. Base URL:

   ```
   http://hostname:[port]/api/[REST-Command]
   ```

2 Activate API Service

To use the ecoDMS API, you must enable the service in the Settings of ecoDMS Client.

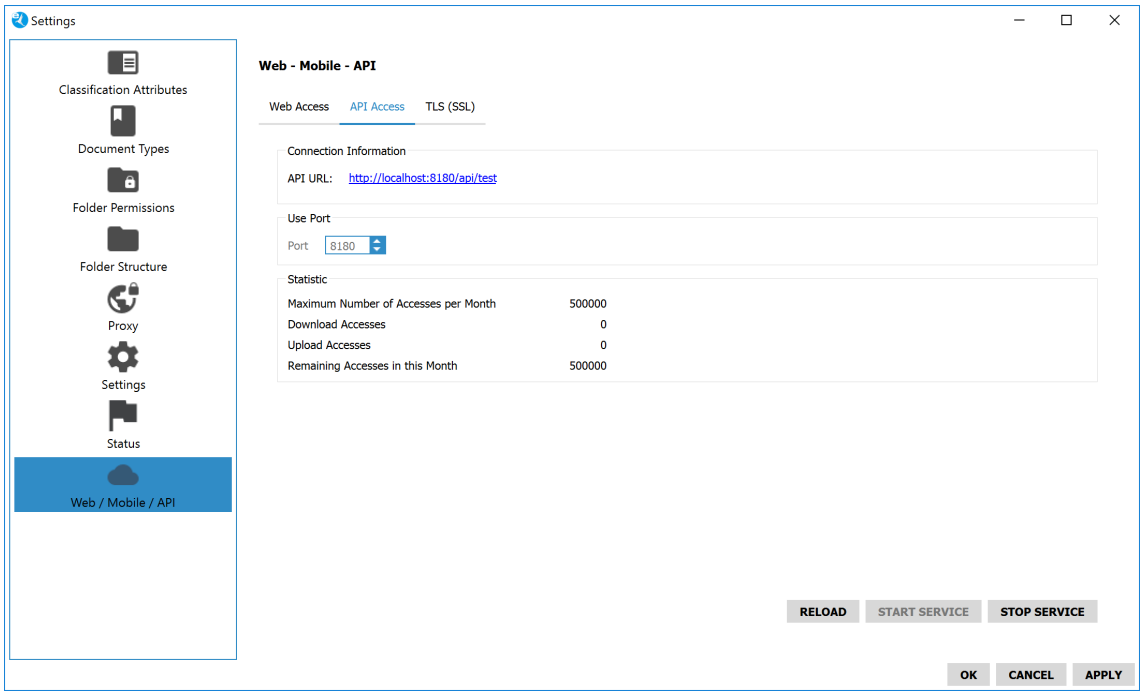

Figure 2.1: Settings - Web - Mobile - API - API Access

2.1 Start Service

To start the API service, complete the following steps:

1. Open the Settings in ecoDMS Client as administrator
2. Click "Web / Mobile / API -> API Access"
3. **Use Port:** The default setting for the port is "8180". Modify the port if necessary.
4. **Start Service:** Click “Start Service” to initiate API access with your settings
   a) The API access is enabled. This process may take a while.
5. **API URL:** This is where ecoDMS displays the link to call the ecoDMS API in your network (internal)
6. Click “Apply” to save or “Cancel” to abort the process

2.2 Stop Service

To stop the API service, complete the following steps:

1. Open the Settings in ecoDMS Client as administrator
2. Web / Mobile / API -> API access -> Stop service

a) The API access is stopped. This process may take a while.

3. Connection information -> Web client URL: The link to the ecoDMS API is not available
4. Click “Apply” to save or “Cancel” to abort the process

3 Statistics

You can view the number of available API connects and those consumed during the current month in the ecoDMS Settings.

3.1 Settings in ecoDMS

1. Open the Settings in ecoDMS Client as administrator
2. Web / Mobile / API -> API Access
3. The statistics display the following values for ecoDMS API:
   a) **Maximum number of accesses per month:** Available number of monthly API connects
   b) **Download accesses:** Number of already downloaded or retrieved documents via the ecoDMS API in the current month
   c) **Upload accesses:** Number of already archived or uploaded documents via the ecoDMS API in the current month
   d) **Remaining accesses in this month:** Number of remaining API connects in the current month

4 Swagger Online Documentation

You can use the Swagger UI for access testing and for API documentation.

4.1 Start

1. Start the Swagger UI with the following command:

   ```
   /api/startDebug
   ```

2. To start Swagger, the API service is restarted in the background.
   a) This may take several seconds.
3. You are automatically redirected to the swagger-ui homepage.
4. After a successful start, you can reach the page through the following link:

   ```
   <servername>/swagger-ui.html#/eco-dms-rest-controller
   ```

For more details on how to use Swagger, go to the project website:

```
https://swagger.io/
```

Figure 4.1: Example for API Documentation

4.2 Stop

When you are using the API port productively or if it is available to everyone, it makes sense to stop Swagger.

1. Stop the Swagger UI with the following command:

   ```
   <server-name>/api/stopDebug
   ```

2. To stop Swagger, the API service is restarted in the background.
   a) This may take several seconds.

5 Login & Authentication in ecoDMS API REST Service

The ecoDMS API REST Service uses HTTP basic authentication according to RFC 2617.

1. The first step when logging in is to connect to ecoDMS Server. To do so, call the REST function *'connect'*. You must select an existing ecoDMS-Archive. As input parameter of this function, enter the ID number of the archive.
   a) Example:
   ```
   http://localhost:8180/api/connect/1
   ```
2. As soon as the connection with an ecoDMS-Archive has been established, check the validity of the ecoDMS user credentials (user name and password). You can do this by sending the Base64-coded authentication pattern in the header of the HTTP request to the ecoDMS API REST Service. To do so, call the REST function *'/test'* (see description of the '/test' function).
   a) Example:
   ```
   http://localhost:8180/api/test
   ```
3. The HTTP response also contains cookie information, which is saved locally by the client and is transmitted for any further transactions in the HTTP request header.

5.1 Available Functions: REST Commands

5.1.1 Test Connection

```
HttpMethod.GET /test
```

Tests the connection with the ecoDMS API REST Service and does not require login. If the connection is successful, the message "applord Test: OK", or similar, displays.

5.1.2 Login and Authentication

```
HttpMethod.GET /connect/{archiveId}
```

Used to login and authenticate an already existing ecoDMS user with ecoDMS API REST Service. You must enter the ID of the respective ecoDMS-Archive. Transactions can only be made if the API REST Service user has been authenticated with the valid ecoDMS user credentials (refer to the example Login and Authentication).

1. Resource URL:
   ```
   http://host:[port]/api/connect/1
   ```

5.2 Parameters

You can use various parameters.

5.2.1 archived ID (required)

ID number of the archive intended for the subsequent transactions.

1. Example value:

```
1
```

5.2.2 ecoDMSUser (required)

This is the ecoDMS user name. The user name must be transmitted to the server with the user password as Base64-coded authentication pattern in the HTTP header.

1. Example value:

```
1
```

5.2.3 ecoDMSPassword (required)

This is the ecoDMS user password. The user password must be transmitted to the server with the user password as Base64-coded authentication pattern in the HTTP header.

1. Example value:

```
sampleuser
```

5.2.4 Example: Login and Authentication

```
exampleuser:mypassword
```

Output Base64-coded (HTTP basic authentication)

```
Basic bXVzdGVydXNlcjpteXBhc3N3b3Jk
```

5.2.5 Terminate Connection

```
HttpMethod.GET /disconnect
```

This command terminates the connection with the ecoDMS API REST Service. A message confirms that the current ecoDMS user is logged off.

This function requires HTTP basic authentication.

1. Resource URL:

```
http://host:[port]/api/disconnect
```

2. Example response:

```
Logoff for user musteruser successful
```

6 Search Filters

```
HttpMethod.POST /searchDocuments
```

Provides a list with ecoDMS document information objects (ecoDocInfo) from the selected ecoDMS-Archive, which is defined with a user-defined search filter. Each ecoDocInfo object contains the following attributes:

- Document ID
- Classification ID
- ID of ecoDMS that contains the specified document
- ecoDMS classification attributes including the descriptions and values (e.g. main folder, folders, comment, status, revision, document type, date, etc.)

The search filter is similar to the extended search in ecoDMS Client and can contain several search criteria. Each search criterion consists of the following three parts:

1. Classification attribute
2. Operator
3. Value

Example: This example displays a list of documents with the status "Resubmission":

```
'status=2'
```

JSON schema:

```
[{"classifyAttribut":"status", "searchOperator":"=", "searchValue":"2"}]
```

To define the search filter, use the following REST functions:

```
/types
/status
/folders
```

These functions supply the IDs required for the search (status IDs, folder IDs, document type IDs,...). For a search request, the values of the respective classification attributes must be set according to their IDs and not according to their names. If, for example, you want to determine the content of an ecoDMS folder, you must specify the exact folder ID. When searching for the content of a folder with the ID = 1.4.5.2 - without including the subfolders - you must set the full folder ID and not only the first part.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/searchDocuments
   ```

6.1 Parameters

You can use various parameters for the search.

6.1.1 searchFilterList (required)

This is a list with individual search filters. This list must be transferred in the HTTP body as JSON message to the API REST Service.

6.1.2 Example Values

```
docid > 0; docid <= 10; docart=2; folder=1.3
```

```
[{"classifyAttribut":"docid", "searchOperator":">", "searchValue":"0"},
{"classifyAttribut":"docid", "searchOperator":"<=", "searchValue":"10"},
{"classifyAttribut":"docart", "searchOperator":"=", "searchValue":"2"},
{"classifyAttribut":"folder", "searchOperator":"=", "searchValue":"1.3"}
{"classifyAttribut":"folder", "searchOperator":"ilike", "searchValue":"TEXT"}]
```

- **Information zum Beispiel:** Zur Suche nach einer Textstelle / einem Wort wird der searchOperator "ilike" benötigt.

6.2 Returns

6.2.1 docInfoList

This is a list containing the ecoDMS document information objects as JSON list.

6.2.2 Example JSON List

```
[{
{"DocID":15,"clDocId":15,"archiveName":"1","classifyAttributes":
{"changeid":"ecoDMS","status":"3","mainfolder":"1","ctimestamp":"2013-05-10
09:58:39","dyn_000011366794281":"","folder":"1.3","revision":"1.3",
"docid":"15#15","docart":"1","comment":"ExampleDocument","rechte":"W",
"defdate":"","cdate":"2013-05-10"},"editRoles":null,"readRoles":null}
}
,{
{"DocID":23,"clDocId":23,"archiveName":"1","classifyAttributes":
{"changeid":"ecoDMS","status":"3","mainfolder":"6","ctimestamp":"2013-06-25
12:49:34","dyn_000011366794281":"","folder":"6.6","revision":"1.5",
"docid":"23#23","docart":"21","comment":"ExampleDocument","rechte":"W",
"defdate":"","cdate":"2013-05-10"},"editRoles":null,"readRoles":null}
}]
```

6.3 Advanced Search Functions

```
HttpMethod.POST /searchDocumentsExt
```

The advanced search function works the same way as the normal search, but it also supports four additional URL parameters.
Example Call:

```
/api/searchDocumentsExt?maxDocumentCount=100&personalDocumentsOnly=false&readRoles=false&
    trashedDocuments=true
```

maxDocumentCount Number of returned documents (max 1,000)

personalDocumentsOnly=true View personal documents only (default=false)

readRoles=false Read document permissions (performance tuning) (default = true)

trashedDocuments=true Read documents from the trash (default = false)

The structure of the search filters and the response data correspond to the process described in the "Parameters" section of this chapter.

7 Call ecoDMS Attributes

You can call each ecoDMS classification attribute.

7.1 Status

```
HttpMethod.GET /status
```

This command provides a list with the statuses defined in ecoDMS. Each status has a unique ID and a unique name. The status "Resubmission", for example, contains the following attributes:

- ID=2
- Name=Resubmission

You need the ID, for example, to search for the value of the "Status" attribute in ecoDMS Archive with a search filter. Moreover, you can modify the value within a classification for a specific document using the status ID.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/status
   ```

7.1.1 Returns

7.1.1.1 statuslist

This list contains the statuses defined in ecoDMS in form of a JSON list.

7.1.1.2 Example JSON List

```
[{"id":3,"name":"Done"},{"id":2,"name":"Resubmission"},{"id":1,"name":"ToDo"}]
```

7.2 Roles / Permissions

7.2.1 List of All Roles

```
HttpMethod.GET /roles
```

This command provides a list with the roles (users, groups) in ecoDMS. You can, for example, use these values to transfer a permission during classification.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/roles
   ```

7.2.2 List of User Roles

```
Http.Methot GET /userRoles
```

This command lists the ecoDMS roles of the current user currently connected with the system.

7.2.3 Returns

7.2.3.1 Roles Overview

This list contains the statuses defined in ecoDMS in form of a JSON list.

7.2.3.2 Example Overview (All Roles)

```
["Accounting","ecoICELogon","ecoSIMSADMIN","ecoSIMSALLDOCS","ecoSIMSCLASSIFY",
"ecoSIMSCREATEFOLDER","ecoSIMSHISTORY","ecoSIMSUSER","ecoSIMSVERSIONING","ecoSIMSWEBCLIENT", "
    ecoSIMSTEMPLATES", "ecoSIMSDELETE","Management","r_ecodms","scanner","Sales"]
```

7.2.3.3 Example Overview (User Roles)

```
["ecoSIMSADMIN", "ecoSIMSUSER", "ecoSIMSCLASSIFY", "ecoSIMSHISTORY", "ecoSIMSALLDOCS", "
    ecoSIMSWEBCLIENT", "ecoSIMSTEMPLATES", "ecoSIMSDELETE", "r_j.doe"]
```

7.3 Document Types

```
HttpMethod.GET /types
```

Supplies a list with the "Document Types" defined in ecoDMS. Each document type has a unique ID, a unique name and a retention period.

Example for the "Invoice" document type.

- Id:'5'
- Name: 'Invoice'
- Retention period:{"years":0,"months":0,"days":0}}.

You need the ID, for example, to search for the value of the "Document Type" attribute in ecoDMS Archive with a search filter. Moreover, you can modify the value within a classification for a specific document using the document type ID.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/types
   ```

7.3.1 Returns

7.3.1.1 docTypeList

This list contains the document types defined in ecoDMS as JSON list.

7.3.1.2 Example Overview


```
[
{"id":1,"name":"Request","frist":{"jahre":0,"monate":0,"tage":0}},
{"id":2,"name":"Offer","frist":{"jahre":0,"monate":0,"tage":0}},
{"id":3,"name":"Letter","frist":{"jahre":0,"monate":0,"tage":0}},
{"id":4,"name":"Notification","frist":{"jahre":0,"monate":0,"tage":0}},
{"id":5,"name":"Invoice","frist":{"jahre":0,"monate":0,"tage":0}}
]
```


7.4 Folders

```
HttpMethod.GET /folders
```

Provides a list of all folders in the ecoDMS folder structure. Each Folder object contains the following information:

- Unique folder ID
- Folder name
- Keys
- Key words
- mainfolder variable

  This variable is assigned to folders that are declared as main folders in ecoDMS.

The unique folder ID of each folder is used, for example, to define a search filter.

```
folder=1.3
```

If you enter the folder ID in the search, ecoDMS returns the content of the specified folder and the content of all subfolders. If the values of the classification attributes "main folder" or "folder" are affected during classification, the folder IDs of the folders are also required (not the folder names).

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/folders
   ```

7.4.1 Returns

7.4.1.1 folderList

This list contains the folders defined in ecoDMS as JSON list.

7.4.1.2 Example Overview


```
[{"mainFolder":false,"oId":"1","foldername":"Customers","externalKey":"10000","buzzwords":""},"
    mainFolder":false,"oId":"1.1","foldername":"ExampleCustomer01","externalKey":"","buzzwords":
    ""},"mainFolder":false,"oId":"1.3","foldername":"ExampleCustomer03","externalKey":"","
    buzzwords":""},{"mainFolder":false,"oId":"2.1","foldername":"ExampleSupplier01","externalKey
    ":"20100","buzzwords":""}]
```

8 Documents and Versions

```
HttpMethod.GET /document/{docID}
```

This function loads the requested document with the transmitted "DocID" from ecoDMS.

```
HttpMethod.GET /document/{docID}/version/{versionNo}
```

If you want to download a specific document version, you can also use this function. However, you must also specify the version together with the DocID. To do so, add $'/version/versionNo'$ in the URL.

As a result, the function returns an ecoDMS File object, which represents the document archived in ecoDMS. Each ecoDMS File object has the following structure:

docid	38
File size	98890
Document name	38OrderSampleCompany.pdf
File type	.txt, .pdf, .doc, etc. -> here: .pdf
File stream	byte

The file information is in the HTTP body of the HTTP response.

This function requires HTTP basic authentication.

1. Resource URL:

```
http://host:[port]/api/document/38
http://host:[port]/api/document/38/version/2
```

8.1 Parameters

This function has various parameters.

8.1.1 docID (required)

DocID stands for "Document Identification Number". After the archiving process, the DocID is assigned automatically and sequentially to each archived document. The ID number is specified in the URL. Then you can download the document associated with the DocID.

Example value:

```
38
```

8.1.2 versionNr (optional)

The version management function allows you to archive many document versions in ecoDMS. Each new version is archived as a new document with an incremented version number. Use this command to retrieve a specific version.

Example value:

```
/version/2
```

8.1.3 Example REST-Command

This example shows how to download the second version of the document archived in ecoDMS with DocID 38.

```
/document/38/version/2
```

8.2 Returns

8.2.1 documentObject

The filestream is transmitted in the HTTP body of the HTTP response message. All file information is in the HTTP header of the HTTP response message.

9 Archiving / Versioning New Documents

To archive a new document in ecoDMS, you can use the following REST functions:

```
/uploadFile
/documentInfo
/classifyDocument
```

1. `/uploadFile`

   First, load the file to ecoDMS Server and into ecoDMS you specify. To do so, use the API REST Service function $'/uploadFile'$. As a result, the function returns the document ID of the newly archived document.

2. `/documentInfo`

   As soon as the document has been successfully loaded to ecoDMS Server and a DocID has been created, you have the option of classifying the file. Some classification attributes such as "Date", "Last Change", etc. are automatically assigned by ecoDMS Server when archiving a new document. You must get this information from ecoDMS before classifying. Use the REST command $'/documentInfo'$ to retrieve the document information. ecoDMS returns the ecoDMS document information object (ecoDocInfo). The $'ecoDocInfo'$- object has the following structure:

   - Document ID
   - Classification ID
   - ID number of ecoDMS which contains the document ecoDMS classification attributes with identification and values (main folder, folder, comment, status, revision, document type, date, etc.)

   The information is provided as a JSON message by the API REST Service (see REST function $'/documentInfo'$). If you want to change a value of a specific classification attribute, you must also make the change at the respective entry in the $'ecoDocInfo'$ object (JSON). Please note that you can only use the IDs as value for a classification attribute and not the name of the attribute.

3. `/classifyDocument`

   If the $'ecoDocInfo'$ object (JSON) contains all necessary information, you can call the API REST Service function $'/classifyDocument'$. To do this, you must specify at least the filename. This entry is interpreted as value for the classification attribute "Comment" in the new ecoDMS document. Refer to the chapter "Classification and Document Information" for more details on classification.

9.1 Transfer File to ecoDMS Archive

```
HttpMethod.POST /uploadFile/{versionControlled}
```

This function transfers a new file into ecoDMS Archive and returns the document ID of the archived document as the result.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/uploadFile/false
   ```

9.1.1 Parameters

This function has various parameters.

9.1.1.1 file (required)

This is the file you want to save in ecoDMS Archive. The filestream is transmitted in the HTTP body of the request message to the API REST Service.

9.1.1.2 versionControlled (required)

This command shows whether this is a versioned document.
Example value:

- $false$: For a non-versioned document.
- $true$: For a versioned document.

9.1.2 Returns

9.1.2.1 docID

After the archiving process, the DocID is assigned automatically and consecutively to each archived document. This is the ID of the newly archived document.

9.2 Transfer File and PDF to ecoDMS Archive

```
HttpMethod.POST /uploadFileWithPDF/{versionControlled}
```

This function transfers a new file and the associated PDF to ecoDMS Archive. You can save a PDF file, for example, with the ecoDMS PDF/A printer or with an ecoDMS Office plugin. The document ID of the archived document is returned as the result.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/uploadFileWithPdf/true
   ```

9.2.1 Parameters

This function has various parameters.

9.2.1.1 file (required)

This is the file you want to save in ecoDMS Archive. The filestream is transmitted in the HTTP body of the request message to the API REST Service.

9.2.1.2 pdfFile (required)

This is the file which is displayed in ecoDMS as default document (e.g. PDF).

9.2.1.3 versionControlled (required)

This command shows whether this is a versioned document.
Example value:

- $false$: For a non-versioned document.
- $true$: For a versioned document.

9.2.2 Returns

9.2.2.1 docID

After the archiving process, the DocID is assigned automatically and consecutively to each archived document. This is the ID of the newly archived document.

9.3 Add New Version

```
HttpMethod.POST /addVersionToDocument/{docID}/{fixed}
```

Use this call to add a new version to an already archived document in ecoDMS. The new file is archived with the specified document as a further version in ecoDMS. You need to specify whether you want to fix the current version of the document. If you set "Fix document", you cannot add any further versions to this document after archiving.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/addVersionToDocument/15/false
   ```

9.3.1 Parameters

This function has various parameters.

9.3.1.1 file (required)

This is the file you want to save in ecoDMS Archive. The filestream is transmitted in the HTTP body of the request message to the API REST Service.

9.3.1.2 docID (required)

Contains the docID of the document to which the file was added as new version.
Example value:

```
15
```

9.3.1.3 fixed / finalised (required)

If you enable "Finalise document", you cannot add any further versions to this document after archiving. Example values: false: Do not fix version.
Example value:

- $false$: Do not finalise version
- $true$: Finalise version

9.3.2 Returns

9.3.2.1 docID

After the archiving process, the DocID is assigned automatically and consecutively to each archived document. This is the ID of the newly archived document.

9.4 Add New Version with PDF

```
HttpMethod.POST /addVersionWithPdfToDocument/{docID}/{fixed}
```

Use this call to add a new version including PDF file to an already archived document in ecoDMS. The new file is archived with the specified document as a further version in ecoDMS. You need to specify whether you want to fix the current version of the document. If you enable "Fix document", you cannot add any further versions to this document after archiving.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/addVersionWithPdfToDocument/15/false
   ```

9.4.1 Parameters

This function has various parameters.

9.4.1.1 file (required)

This is the file you want to save in ecoDMS Archive. The filestream is transmitted in the HTTP body of the request message to the API REST Service.

9.4.1.2 docID (required)

This is the file which is displayed as default document in ecoDMS.

- For example: PDF

9.4.1.3 fixed (required)

If you enable "Fix document", you cannot add any further versions to this document after archiving.
Example value:

- $false$:Do not fix version
- $true$: Fix version

9.4.2 Returns

9.4.2.1 docID

After the archiving process, the DocID is assigned automatically and consecutively to each archived document. This is the ID of the newly archived document.

10 Classification and Document Information

Classification is the assignment and allocation of document information and document permissions. You can assign the folder, document type, status, permissions and further details to each archived file. You can assign, display and edit the classification attributes and document information from ecoDMS.

10.1 Display Document Information

```
HttpMethod.GET /documentInfo
```

Provides an ecoDMS document information object (ecoDocInfo) for the document archived in ecoDMS if you enter the matching document ID. This object contains the following attributes:

- Document ID
- Classification ID
- ID number of ecoDMS which contains the document
- ecoDMS classification attributes with identification and values (main folder, folder, comment, status, revision, document type, date, etc.)

This document information is required, for example, for classification. You can assign new classifications and edit existing classifications.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/documentInfo/38
   ```

10.1.1 Parameters

The following parameters are available for this function.

10.1.1.1 docID (required)

This is the ID number of the archived document that requires the classification attributes and document information. The ID is specified in the URL.
Example value:

```
38
```

10.1.2 Returns

10.1.2.1 docInfoObject

You can read the ecoDMS document information object in the HTTP header of the HTTP response message as JSON message.
Example:
JSON structure of an ecoDMS document information object (ecoDocInfo)

```
{"DocID":10,"clDocId":10,"archiveName":"1","classifyAttributes":
{"changeid":"ecoDMS","status":"3","mainfolder":"1","ctimestamp":"2013-05-10 09:59:09","
    dyn_000011366794281":"","folder":"1.3","revision":"1.3","docid":"10#10","docart":"19","
    comment":"ExampleDocument","rechte":"W","defdate":"","cdate":"2013-05-10"},"editRoles":null,
    "readRoles":null}
```


10.2 Classify

```
HttpMethod.POST /classifyDocument
```

Use this command to classify the document archived in ecoDMS. This command sets the current values of all required classification attributes. Use the REST function $'/documentInfo'$ to call the ecoDMS document information object (ecoDocInfo) for each document.

The document information object must be adjusted with the current classification values. Then the object can be transmitted in the body of the HTTP request to the API REST Service.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/classifyDocument
   ```

10.2.1 Parameters

The following parameters are available for this function.

10.2.1.1 docID (required)

This is the ID number of the archived document you want to classify. The ID is specified in the URL.
Example value:

```
38
```

10.2.1.2 docInfoObject (required)

This is the ecoDMS document information object with the current classification values. Example: JSON structure of an ecoDMS document information object (ecoDocInfo).


```
{"DocID":10,"clDocId":10,"archiveName":"1","classifyAttributes":
{"changeid":"ecoDMS","status":"3","mainfolder":"1","ctimestamp":"2013-05-10 09:59:09",
"dyn_000011366794281":"","folder":"1.3","revision":"1.3","docid":"10#10","docart":"19","comment"
    :"ExampleDocument","rechte":"W","defdate":"","cdate":"2013-05-10"},"editRoles":null,"
    readRoles":null}
```


10.2.2 Returns

10.2.2.1 docID

After the archiving process, the DocID is assigned automatically and consecutively to each archived document. This is the ID of the classified document.

10.3 Display List of Defined Classification Attributes

```
HttpMethod.GET /classifyAttributes
```

Use this call to display a list of all classification attributes defined in ecoDMS. Each list entry contains the following structure:

- "Column name in the database" : "Name in ecoDMS"

You can use this function, for example, if you need the column names of the dynamic attributes.

1. Resource URL:

   ```
   http://host:[port]/api/classifyAttributes
   ```

10.3.1 Returns

10.3.1.1 classificAttrList

This list contains the classification attributes defined in ecoDMS as JSON list.

10.3.1.2 Example Overview

```
{
"changeid":"Editedby","status":"Status",
"mainfolder":"MainFolder","ctimestamp":"LastChange",
"dyn_000011366794281":"OrderNumber","folder":"Ordner",
"docid":"DocId","revision":"Revision","dyn_000011380876313":"Paul02",
"docart":"DocumentType","comment":"comment",
"rechte":"Permission","dyn_000011380867530":"Test2",
"defdate":"Resubmission","cdate":"Datum"
}
```

10.4 List of Default Classification Attributes

Various attributes can be assigned during assignment / classification. There are default attributes and dynamic attributes. The identification of dynamic classification attributes (attributes that are manually defined in ecoDMS by the administrator), can be called with the REST function $'documentInfo'$.

Example: Identifier of a dynamic classification attribute:

- dyn_000011366794281

List of default classification attributes:

```
ECO_CLASSIFY_MAINFOLDER = "mainfolder";
ECO_CLASSIFY_FOLDER = "folder";
ECO_CLASSIFY_STATUS = "status";
ECO_CLASSIFY_DOCTYPE = "docart";
ECO_CLASSIFY_comment = "comment";
ECO_CLASSIFY_DOCID = "docid";
ECO_CLASSIFY_REVISION = "revision";
ECO_CLASSIFY_FULLTEXT = "fulltext";
ECO_CLASSIFY_RESUBMISSION = "defdate";
ECO_CLASSIFY_DATE = "cdate";
ECO_CLASSIFY_MODIFIED_BY = "changeid";
ECO_CLASSIFY_MODIFIED= "ctimestamp";
```

10.5 Create Multiple Classification

```
HttpRequest.POST /createNewClassify
```

Creating additional classifications (multiple classifications) for a document.
This function requires HTTP basic authentication.

1. Example URL:

   ```
   http://host:[port]/api/createNewClassify
   ```

10.5.1 Parameters

The following parameters are available for this function.

10.5.1.1 docInfoObject (required)

This is the ecoDMS document information object with the current classification values. Example: JSON structure of an ecoDMS document information object (ecoDocInfo).


```
{"DocID":10,"clDocId":10,"archiveName":"1","classifyAttributes":
{"changeid":"ecoDMS","status":"3","mainfolder":"1","ctimestamp":"2013-05-10 09:59:09",
"dyn_000011366794281":"","folder":"1.3","revision":"1.3","docid":"10#10","docart":"19","comment"
    :"ExampleDocument","rechte":"W","defdate":"","cdate":"2013-05-10"},"editRoles":null,"
    readRoles":null}
```


10.5.2 Returns

10.5.2.1 cldocID

ecoDMS provides the unique classification ID for the multiple classification you created.

11 Create New Folder

```
HttpMethod.POST /createFolder/parent/{parentoid}
```

This function creates a new folder in the ecoDMS folder structure.

- If you want to create a new folder at the top folder level, you do not require the values for *'parent'* and *'parentoid'* .
- If you want to create a subfolder in an already existing ecoDMS folder, you must specify *'parent'*, followed by the respective folder ID of the superordinate folder in the URL.

To define a new folder, use the ecoDMS folder object. Each ecoDMS folder object contains the following information:

- Unique folder ID
- Folder name
- ecoDMS key
- Key words
- mainfolder variable

  This variable is assigned to folders that are declared as main folders in ecoDMS.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:[port]/api/createFolder/parent/2.1
   ```

11.1 Parameters

The following parameters apply:

11.1.1 dmsOrdnerObject (required)

The ecoDMS folder object (dmsOrdnerObject) contains all required parameters for creating a folder.

11.1.2 parentoid (optional)

This is the "folder ID" of the superordinate folder (parent folder).
Example value, if you require a subfolder from ecoDMS with the ID=2.1:

```
2.1
```

11.2 Returns

11.2.1 ordnerID

The system automatically assigns the folder ID. This is the ID of the newly created folder.

11.3 Create Folder at Top Level

```
HttpRequest.POST /createFolder
```

Creating a new folder without parent folder.
This function requires HTTP basic authentication.

1. Example URL:

   ```
   http://host:[port]/api/createFolder
   ```

2. Transfer parameters

   ```
   {"active": true, "buzzwords": "keywords ", "externalKey": "key", "foldername": "Folder Name
       ", "mainFolder": true}
   ```

12 Document Preview

```
HttpMethod.GET /thumbnail/{DocID}/page/{pageNr}/height/{height}
```

Displays the preview of a specified document with the transmitted DocID and the specified page number $'pageNr'$. You must always define the preview quality with the $'height'$ parameter.

This function requires HTTP basic authentication.

1. Resource URL:

   ```
   http://host:8080/api/thumbnail/15/page/1/height/1080
   ```

12.1 Parameters

Three parameters are required for this function.

12.1.1 docID (required)

DocID stands for "Document Identification Number". After the archiving process, the DocID is assigned automatically and consecutively to each archived document. The ID number is specified in the URL.
Example value:

```
15
```

12.1.2 pageNr (required)

The number of a specific page in a document you want to preview. The number of the first page starts with the value $'1'$. If the specified page does not exist, the function returns "HTTP ERROR 404: Not found".

12.1.3 height (required)

Height of the preview. The specified height influences the quality of the resulting preview. The greater the value, the higher the quality of the preview.

12.2 Returns

12.2.1 File Stream

The preview of a specified page from a selected document.

12.2.2 Preview Size

The maximum display size (height) for a thumbnail is 1080 pixels.

13 Linking

When linking documents, any archived files, including their classifications, can be added to a document and combined to form one process. The links are displayed in ecoDMS Client table as expandable sub-entries of the main document. The main entry of each linked document remains as it is. This function is similar to a virtual document clip.

13.1 Link

```
HttpRequest.POST /document/{clDocId}/linkToDocuments
```

clDocId Corresponds to the classification ID. You can call it with the following API function:

```
 /documentInfo/{docId}
```

Example URL:

```
http://localhost:8180/api/document/7/linkToDocuments
```

Transfers the list of clDocIds of the documents you want to link
Example parameter for Post:

```
[1, 2, 3, 4, 5, 6]
```

13.2 Display

```
HttpRequest.GET /document/{clDocId}/readLinkedDocuments
```

This command provides a list of the linked documents.
Example URL:

```
http://localhost:8180/api/document/7/readLinkedDocuments
```

Example Response:

```
[1, 2, 3, 4, 5, 6]
```

13.3 Delete

```
HttpRequest.POST /document/{clDocId}/removeDocumentLink
```

This command removes the links between documents.
Example URL:

```
http://localhost:8180/api/document/7/removeDocumentLink
```

Example data for Post:

```
[1, 2, 3]
```

After deleting the links, you can check the result with the following command:

```
readLinkedDocuments
```

14 Delete

The API REST Service has a restricted trash function. Documents can be moved to the trash. You can remove documents / classifications that are no longer needed from the standard view (table in ecoDMS Client). In this process, the files are not deleted but stored in a virtual recycle bin.

- The search and filter functions are also available in the trash.
- Classification is not possible in the trash.
  - To edit the classification, the document would need to be restored from the trash with the restore function.

14.1 Move to Trash

```
HttpRequest.GET document/{clDocId}/moveToTrash
```

This command moves the selected document / classification to the trash.
Example URL:

```
http://localhost:8180/api/document/7/moveToTrash
```

Example Response:

```
true
```

14.2 Restore

```
HttpRequest.GET /document/{clDocId}/removeFromTrash
```

This command restores a document from the trash.
You can use the following API function to search for deleted documents:

```
searchDocumentsExt
```

Example URL:

```
http://localhost:8180/api/document/7/removeFromTrash
```

Example Response:

```
true
```

15 Upload Size

The default upload size for ecoDMS API REST Service is limited to 10 MB per file. As of ecoDMS version 18.09 (apu), you can adjust the upload size on ecoDMS Server if required.

15.1 Windows

To adjust the upload size on a Windows installation of ecoDMS Server, complete the following steps:

1. Open the Windows Services Manager.
2. Stop the “ecoDMS Server 18.09” service.
3. Open the file “ecodms.properties” in the following folder:

   ```
   C:\Program Files (x86)\ecoDMS GmbH\ecoDMS Server
   ```

4. Now adjust the entry "rest.api.maxUploadSizePerFile". Please add the two lines to the file:

   ```
   rest.api.maxUploadSize=-1
   rest.api.maxUploadSizePerFile=104857600
   ```

   Please do NOT change the other values in the file!

5. Save this file to the same path.
6. Start the “ecoDMS Server 18.09” service.