centraQuest

centraQuest DMS Webservice API Description

November 2020

Document History

Filename:	centraQuest_DMS_API_Description.docx
Author:	ComLog GmbH
Created:	12.05.2014
Changes:	30.11.2020

Inhaltsverzeichnis

Introduction

The functionality of centraQuest DMS is accessible programmatically by a JSON REST Webservice API. There is also another server based API that is used internally within the server process. The server side API is used to implement for example server side jobs and triggers and HTML templates, etc. The server side API is similar in some ways to stored procedures found in relational databases. The server side API is not scope of this document.

API Description

DMS REST Webservice API Calls

All requests, except for login request, must be passed a valid session ticket for authentication. The session ticket can either be passed as a parameter in the URL or as a cookie HTTP-Header.

The API is designed so that it can be used directly from a browser/HTML page without the need for client libraries.

Function/URL is the relative URL, e.g. 'dms/login' for example would be called with the following URL https://<hostname>[:port]/<Function/URL>?....

Many API calls have an id parameter. The ID parameter is usually (depending on the backend) a GUID id that uniquely identifies a document/object in the DMS.

Example: https://<hostname>/dms/login?user=<user>&password=<password>

dms/login

Function/URL	dms/login
Description	A valid session ticket is needed to access the API the login function is used to obtain a session ticket, by passing a user name and password as parameter. If the login is successful a session ticket is returned and also the name for the cookie (Either GET or POST are supported)
Params	- user - password (encrypted)
Result	{"session-cookie-name":"centraquest","ticket":"4fcede6817d984db9d686d7997f8a023"}
Error	{"message":"Login failed","id":"login-failed"}
Example	dms/login?user=<user>&password=<password>

dms/logout

Function	dms/logout
Description	Invalidates an existing session (session ticket). Sessions timeout automatically if not used for a longer (configurable) time, but it is best practice to call logout when the session is not needed anymore.
Params	No parameters except for session ticket
Result	{"status":"ok"}
Error	Status Code: 500, 400 depending on error condition
Example	dms/logout dms/logout?ticket=xxxxyyyydfff

dms/ssologin

Function	dms/ssologin
Description	Challenge Response based single sign on login usually used together with other SSO mechanisms like NTLM, Kerberos, etc. , two-factor authentication Not publicly documented as it is not intended to be used directly, but through other SSO mechanisms
Params	
Result	
Error	Status Code: 500, 400 depending on error condition
Example	dms/ssologin

dms/children

Function	dms/children
Description	Returns child objects of an object, e.g. for folder objects this corresponds to the content of a folder. But any object (including documents) in centraQuest DMS can have parent child relations, e.g. a main contract document can be linked to a subcontract with a parent-child relation. Returns metadata of each child object as a json object, i.e. the result is a list of JSON objects.
Params	Id – object id of parent
Result	[{"obj_name":"98 Daten","obj_profile":"cq_folder","_icon":"resource:icons\/cq_icon_folder.jpg","obj_created":"20160323212355","obj_child_count":1,"obj_references":["2a7883cff13511e5a7f69d05d3d2de9c"],"obj_vstamp":3,"obj_mask":"cq_folder","obj_permission":60,"obj_sort_order":"Aobj_name","obj_id":"2a7d8cdef13511e5b8779d05d3d2de9c","obj_owner_name":"admin","_children":"\/dms\/children?id=2a7d8cdef13511e5b8779d05d3d2de9c&sort_order=Aobj_name", "obj_modified":"20160323212355","_parent":"2a7883cff13511e5a7f69d05d3d2de9c"}]
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/children?id=root&ticket=<ticket>

dms/content

Function	dms/content
Description	Returns content of an object/document
Params	- id - ticket - lock
Result	Returns content of document/object in HTTP response body and HTTP headers like content_type, content_length, content_disposition are set accordingly, i.e. works directly when called in a browser
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/content?id=<id>&ticket=<ticket>

dms/upload

Function	dms/upload
Description	Transfers/uploads a file/document to the DMS. The call must be done using POST method and the request body must be multipart form-data encoded. It is designed to work directly with a simple form upload in HTML. The file can be uploaded including index values, the parameter names must be the internal names for the attributes, e.g. obj_name,obj_description, etc. The file parameter must be named 'FILE'. <form action="dms/upload" method="post" enctype="multipart/form-data"> Select image to upload: <input type="file" name="FILE" id="FILE"> <input type="text" id="obj_name" value="<name of the document>"> <input type="submit" value="Upload Image" name="submit"> </form>
Params	In addition the following URL parameters can be passed - id (GUID of an existing document in the DMS. If present the uploaded file will be added as a new version to the existing document. Without id parameter the content will be uploaded as a new object/document) to the DMS - ticket - parent (id of parent folder where new document should be put in) - original_filename - {params} additional index values can also be passed in url. Names must start with _ and then internal name of the index field. - version (version number) - version_comment (description of this version)
Result	Returns metadata of new object (or existing object if a version was added) as json object
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/upload?id=<id>&version=2.0&version_comment=Hinweis&ticket=<ticket>

dms/checkout

Function	dms/checkout
Description	Puts an exclusive lock on the object. Only the owner of the lock is allowed to modify the document/object as long as the lock is present. This is usually used to prevent concurrent updates by multiple users.
Params	- id (id of existing content in archiv) - ticket
Result	{'status': 'ok', 'operations' : [{'operation' : 'refresh', 'id' : id, 'properties' : 'dms/properties?id=%s' % id}]}

Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/checkout?id=<id>&ticket=<ticket>

dms/cancel-checkout

Function	dms/cancel-checkout
Description	Removes the exclusive lock from an object/document created with the checkout call
Params	- id (id of existing content in archiv) - ticket
Result	{'status': 'ok', 'operations' : [{'operation' : 'refresh', 'id' : id, 'properties' : 'dms/properties?id=%s' % id}]}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/cancel-checkout?id=<id>&ticket=<ticket>

dms/delete

Function	dms/delete
Description	Deletes an object/document in the DMS
Params	- id (id of existing content in archiv) - ticket
Result	{"operations":[{"operation":"remove","type":"cq_list","id":"77f5cd30f1cf11e5a9efa359a95b1588"}],"status":"ok"}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/delete?id=<id>&ticket=<ticket>

dms/create-relation

Function	dms/create-relation
Description	Creates a relation between to existing documents/objects in the DMS. Relations can be named (i.e. typed). For example for parent child relations the name is 'parent'
Params	- id (id of child object) - parent (id of parent object) - ticket
Result	{"operations":[{"parent":"root","operation":"add","properties":"dms\/properties?id=eade0280f13f11e5bf847d43ecd1db7b","id":"eade0280f13f11e5bf847d43ecd1db7b"}],"status":"ok"}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/create-relation?id=<id>&parent=<parent_id>&name=parent&ticket=<ticket>

dms/delete-relation

Function	dms/delete-relation
Description	Deletes an existing relation or relations between to objects/documents in the DMS.
Params	- id (id of child object) - parent (id of parent object) - name (name/type of relation) - ticket Only one of parent an id parameter must be present. name parameter is optional.
Result	{"operations":[{"parent":"root","operation":"add","properties":"dms\/properties?id=eade0280f13f11e5bf847d43ecd1db7b","id":"eade0280f13f11e5bf847d43ecd1db7b"}],"status":"ok"}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/delete-relation?id=<id>&parent=<parent_id>&name=parent&ticket=<ticket>

dms/dql

Function	dms/dql
Description	centraQuest DMS supports an SQL inspired query language for accessing data in the DMS. The query language is called DQL (stands for Document Query Language). See chapter about DQL for more details.
Params	- ticket - query (must be URL encoded or form url encoded if post method is used) - result_limit (maximum number of results that should be returned) - include_info (optional – returns additional info, like colum names and lables that can be used by client application to display results)
Result	Returns result as a list of json objects [{"obj_permission":60,"columns":[{"length":20,"type":"string","id":"obj_modifier"},{"length":20,"type":"string","id":"obj_name"},{"length":20,"type":"string","id":"obj_profile"},{"length":20,"type":"string","id":"cq_objnr"},{"length":20,"type":"string","id":"obj_content_type"},{"length":20,"type":"string","id":"obj_content"},{"length":20,"type":"string","id":"cq_objname"},{"length":20,"type":"string","id":"obj_vstamp"},{"length":20,"type":"string","id":"obj_mask"},"obj_name":"E - Kreditorenrechnung - Loreda GmbH & Co. KG - ","obj_profile":"cq_kreditor","_icon":"\/dmslite\/static\/icons\/format\/f_text_plain_16.gif","cq_objnr":"","obj_content_type":"text\/plain","obj_content":"2016\/68a\/68a2dbc0da5f11e5b89f7be476875c9c.txt","cq_objname":"","obj_vstamp":5,"obj_mask":"cq_document","cq_adrnr":"10013","cq_description":"","obj_size":16,"obj_id":"68a329e1da5f11e587f07be476875c9c","obj_created":"20160223195821","cq_cust_date1":"20160223","obj_references":["39d1b061dd6f11e58ac6c187f8147e2d","39fdc970dd6f11e583e8c187f8147e2d"]}]
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/dql?query=select+%2A+from+cq_document+where+obj_profile%3D%27cq_kreditor%27+order+by+dobj_created&include_info=true&result_limit=100&ticket=<ticket>

dms/dql-query

Function	dms/dql-query
Description	Like the dms/dql except that the DQL is not passed as a parameter, but the query is defined server side and only the id of the server side query is passed as a parameter.
Params	- query (id of server side query) - result_limit (maximum number of results that should be returned) - include_info (optional – returns additional info, like colum names and lables that can be used by client application to display results) - {params} (optional params to be passed to server side query)
Result	Returns result as a list of json objects [{"label":"Andere","obj_permission":60,"id":"Andere"},{"label":"Erdarbeiten","obj_permission":60,"id":"Erdarbeiten"},{"label":"F & B","obj_permission":60,"id":"F & B"},{"label":"GL","obj_permission":60,"id":"GL"},{"label":"IT","obj_permission":60,"id":"IT"},{"label":"Personal\/Finanzen","obj_permission":60,"id":"Personal\/Finanzen"},{"label":"Sekretariat\/Dienste","obj_permission":60,"id":"Sekretariat\/Dienste"}]
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/dql-query?query=qryGetKeywords&parent=all&field=cq_department&profile=all&ticket=<ticket>

dms/properties

Function	dms/properties
Description	Returns the metadata (index values) of a document/object as a json object if GET method is

	used. If POST method is used with application/json as content_type for the request body then the properties are updated
Params	- ticket - id
Result	{"_icon":"\/dmslite\/static\/icons\/type\/t_dm_folder_16.gif","obj_created":"20140326213505","obj_child_count":5,"obj_sprt_order":"","obj_vstamp":1661,"cq_client":"100 - Firma","obj_permission":60,"obj_sort_order":"aobj_name","obj_id":"root","_children":"\/dms\/children?id=root&sort_order=aobj_name","obj_modified":"20160323212355","_parent":"root"}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/properties?id=<id>&ticket=<ticket>

dms/copy

Function	dms/copy
Description	Creates a copy of an existing object/document in the DMS
Params	- ticket - id - parms (optional paramters to specifiy index values that should be used for the copied object instead of the index values from the original object. Also additional folder links can be passed for the copy.
Result	{"operations":[{"parent":"b7a4f61ee78b11e58bea4bf7306065e8","operation":"add","properties":"dms\/properties?id=cedce20001a611e681971bb33c20c17a","parent_children":"dms\/children?id=b7a4f61ee78b11e58bea4bf7306065e8","id":"cedce20001a611e681971bb33c20c17a"}],"status":"ok"}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/copy?id=<id>&[{obj_name=Test123}]&ticket=<ticket>

dms/link

Function	dms/link
Description	Links a document/object to a folder
Params	- ticket - id (id of object that should be linked to folder) - parent (id of folder)
Result	{'status': 'ok', 'operations' : [{'operation' : 'add', 'id' : id, 'parent' : parent, 'properties' : 'dms/properties?id=%s' % id}]}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/link?id=<id>&parent=root&ticket=<ticket>

dms/unlink

Function	dms/unlink
Description	Unlinks a document/object from a folder
Params	- ticket - id (id of object that should be unlinked from folder) - parent (id of folder) - delete (optional, if present then object document is deleted if object/document is not contained in any folder anymore)
Result	{'status': 'ok', 'operations' : [{'operation' : 'remove', 'id' : id, 'parent' : parent}]}
Error	Status Code: 500, 400 depending on error condition
Example	https://<hostname>/dms/unlink?id=<idparent=root&ticket=<ticket>

DQL – Document Query Language

One of the biggest differences to SQL is that in the FROM clause no table names are used, but actually object/document classes. The data model is also not really relational, but more object relational including multi value support. The object class 'object' is the basic types and means query over all object/document classes. There is also the 'class' relation which used used to query relations between objects and documents.

Examples:

```
select * from object
```

Select all objects/documents (usually not a good idea)

```
select obj_name as name,obj_id from object
```

Select explicitly certain fields in select clause

```
select * from contract
```

Select only contract documents for example

```
select distinct obj_name, count(*) as cnt from object
select distinct obj_name, substring(foo,3,5) as cnt from object
select distinct obj_name from cqp_list  order by Aobj_name limit 100
```

Count, distinct and various functions like substring, order by and limit are supported as well

```
select * from object where obj_created >= DATE('2010.2.13','%%Y.%%m.%%d')
order by Dobj_created
```

```
select count(*) from object where obj_profile = 'cqp_list' and (obj_name like
'%a%')
```

```
select * from object where obj_created >= DATE('2010-2-13') order by
Dobj_created
```

A where clause can be specified to constrain the result just like in SQL.

```
select * from object where obj_id in (select child_id from relation where
name='related')
```

Subqueries are also supported. In this example all object/documents that have a relation to another object/document of type 'related' are returned.

Subqueries are supported as well.

```
update cm_contract set obj_name = 'foo'
```

```
update object set obj_name = 'foo' link '/Administration/archived' where
status=5
```

Document objects can also be updated using DQL including linking and unlink an object from and to a folder.

```
delete contract objects where status=1
```

Deleting is also supported. A where clause must be specified to select the documents/objects to be deleted. This example deletes all contract objects where status = 1

Appendix – Standard Fields/Attributes

Any number of custom metadata fields/attributes can be defined in the DMS, but there are a couple of built-in fields that are always present and ready to be used. The built-in fields begin either with prefix 'obj_' or 'cq_'.

Here is a list of the most important built-in fields:

obj_id	GUID of object/document
obj_name	Name / Label of document
obj_created	Creation timestamp
obj_modified	Last modification timestamp
obj_owner_name	Owner of the object/document
obj_profile	Name of profile. Profile is sort of an object class and defines behavior and presentation in UI
obj_lock_name	Name of user who locked the object
obj_hist_count	Number of version
obj_child_count	Number of child objects linked to this object (usually a folder)
obj_vstamp	Internal update counter used for optimistic locking
obj_mask	Base type of objects. Defines which fields are associated with this object
obj_version	Version number of object, eg. 1.0, 1.1.1, 2.0, etc.
obj_version_labels	Symbolic version labels, e.g. draft, effective, final (multivalue)
obj_version_comment	Additional free text describing particular version
obj_content	Reference to content "file"
obj_content_type	Mime Type of content (only used for documents)