

# Tutorial: Managing Object Permissions in Documentum REST Services

Version 30

created by lili6561 on Nov 1, 2016 4:35 AM, last modified by lili6561 on Dec 12, 2016 8:35 PM

rest-version 7.3+

This document is to introduce how to manage access control with Documentum REST Services 7.3. In the following article, we will talk about how to manage Access Control List (ACL) and object permission set in REST APIs, and explain the relationship between ACL and permission set.

## Background

### ACL

An Access Control List (ACL) is applied to a repository object to define object-level security, or to a folder for use in folder security. Each SysObject has an associated ACL to control access for the object.

### Object Permission Set

The object permission set is a set of accessor (user or group) permission rules to control the access to this SysObject. The permission set definition derives from the SysObject's associated ACL definition, but it has the intuitive representation from human being understanding.

### Object Permissions

The object permissions is to get a specific user or group's **calculated effective permissions** on a SysObject. The object permissions include BASIC permission and EXTEND permissions.

### User Permission Set

The user definition permission set is the **default** permission set which applies to SysObjects newly created by this user. Each user has his own user permission set. More precisely speaking, the user permission set applies when an ACL is not explicitly assigned to the new SysObject and the *default\_acl* of Content Server config is set to 3.

## Permission Management Resources

This table lists all REST resources about the permission management, that are newly introduced in Documentum REST Services 7.3.

Resource Name	Resource Description
ACLs Collection Resource	The ACLs collection resource models the collection of all available ACLs in the repository, it supports POST method to create a new ACL.
ACL Resource	The ACL resource models the object instance for an ACL, it supports POST and DELETE methods to edit or delete this object.
ACL Associations Collection Resource	The ACL associations resource models the collection of all the available Sysobjects that associated with a specific ACL. An ACL can be applied on multiple SysObjects, this resource can get all the available SysObjects that the specific ACL is applied on.
Permission Set Resource	The permission set resource models the permission management on a SysObject, it's associated with an specific SysObject.
Permissions Resource	The permissions resource models the calculated effective basic and extend permissions on a specific SysObject for a specified user. By default, it returns the effective permissions for the current login user.
User Permission Set Resource	The user permission set resource models a specific user's(or group's) permission set. The user's permission set will be set as the default permission set of the SysObject that created by this user when the <i>default_acl</i> is set 3.

- *ACLs and ACL resources model dm\_acl* objects as regular persistent resources with CRUD operations. It is usually an admin role user who uses this API.

The following diagram shows the state transitions among these resources (and other resources).

 ECD Architecture Team > Tutorial: Access Control Management in REST Services > Capture.PNG

## Samples

In the following samples, we will show how to manage (CRUD) ACLs and object permission set in REST Services. We will use two users with different privileges to run this sample.

- user '**dave**' is a normal user (privilege=0)
- user '**dmadmin**' is a super user (privilege=16)

Samples in this tutorial are organized as below table.

Samples Index	Description
<b>Permission Set Management Samples</b>	Samples about getting, editing a SysObject's permission set object or getting computed permissions on the SysObject for specified user with REST API
<b>ACL(s) Management Samples</b>	Samples about getting ACLs collection feed, managing(CRUD) or getting associations of an single ACL object with REST API
<b>ACL and Permission Set Comparison</b>	Compare the REST representations of ACL and permission set object

## Permission Set Management Samples

Samples in this section are show as below user stories.

Story Name	User	Intention
Get a SysObject's permission set	dave	Gets the permission set object of a specified SysObject
Edit a SysObject's permission set	dave	Updates the permission set object of a specified SysObject
View permissions for specified user or group	dave	Gets the computed basic and extend permissions on a specific SysObject for a specified user or group

### Permission set notes

The permission set representation could have four sections:

- *permitted*
- *restricted*
- *required-group*
- *required-group-set*

When certain sections do not appear in the permission set representation, it indicates that those sections are not set by this permission set. Here are more detail explanations of these sections.

1. ***permitted***: the accessor-permissions defined in the *permitted* type means the user or group is granted with corresponding permissions to this object, it's to define the user or group can do which level operations on this object.
2. ***restricted***: the accessor-permissions defined in the *restricted* type is to restrict the user's(or group's) access to this object with corresponding permissions. As the basic permission is hierarchical, the basic permission defined in the *restricted* type is the user's or group's upper level permission, and the defined extend permissions in the *restricted* type are excluded from the granted.
3. ***required-group***: it's a group member constraint to the accessor, only the user or group who is a member of all the groups that defined in the *required-group* can access the object.
4. ***required-group-set***: it's also a group member constraint, the accessor has to be a member of at least one of the groups that defined in the *required-group-set*.

## Get a SysObject's permission set object

Before managing the permission set, user 'dave' firstly tries to create a document in his own cabinet.

### Request

```

01. POST /dctm-rest/repositories/REPO/folders/0c00000580001909/documents HTTP/1.1
02. Host: localhost:8080
03. Content-Type: application/vnd.emc.documentum+json
04. Authorization: Basic ZGF2ZTpwYXNzd29yZA==
05.
06. {
07.     "name": "document",
08.     "type": "dm_document",
09.     "properties": {
10.         "object_name": "dave_created_doc_for_samples",
11.         "r_object_type": "dm_document",
12.         "owner_name": "dave"
13.     }
14. }
```

### Response

```

01. {
02.     "name": "document",
03.     "type": "dm_document",
04.     "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_document",
05.     "properties": {
06.         "object_name": "dave_created_doc_for_samples",
07.         "r_object_type": "dm_document",
08.         "owner_name": "dave",
09.         "acl_domain": "dave",
10.         "acl_name": "dm_450000058000d07",
11.         "r_object_id": "0900000580004ba5",
12.         .....
13.     },
14.     "links": [
15.         .....
16.         {
17.             "rel": "http://identifiers.emc.com/linkrel/permission-set",
18.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004ba5/permis
19.         },
20.         {
21.             "rel": "http://identifiers.emc.com/linkrel/permissions",
22.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004ba5/permis
23.         }
```

```

24.     }
25.   ]
26. }

```

From the created document response, there are two permission related link relations

- <http://identifiers.emc.com/linkrel/permission-set>
- <http://identifiers.emc.com/linkrel/permissions>

The **permission-set** link points to this document's permission set (from ACL definition). The **permissions** link points to the calculated basic and extend permissions on this document for specified accessor.

Next, the client invokes GET method on the **permission-set** link relation to view the permission set definition for this document.

### Request

```

01. GET /dctm-rest/repositories/REPO/objects/0900000580004ba5/permission-set HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==

```

The response body contains a set of accessor permissions.

### Response

```

01. {
02.   "permitted": [
03.     {
04.       "accessor": "dm_world",
05.       "basic-permission": "Read",
06.       "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
07.     },
08.     {
09.       "accessor": "dm_owner",
10.       "basic-permission": "Delete",
11.       "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
12.     },
13.     {
14.       "accessor": "dm_group",
15.       "basic-permission": "Version"
16.     }
17.   ],
18.   "links": [
19.     {
20.       "rel": "self",
21.       "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004ba5/permis
22.     },
23.     {
24.       "rel": "edit",
25.       "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004ba5/permis
26.     },
27.     {
28.       "rel": "http://identifiers.emc.com/linkrel/acl",
29.       "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/450000058000d07"
30.     }
31.   ]
32. }

```

In this sample, no explicit ACL is assigned to the new document during the document creation. Per the demo Content Server configuration, *dave's* user permission set is assigned to the newly created document. By the way, you can find *dave's* user permission set following the link relation "<http://identifiers.emc.com/linkrel/permission-set>" on *dave's* user resource.

## Edit a SysObject's permission set

Next, we will update this document's permission set for these intentions.

- Grant group **docu** with the **write** permission
- Revoke user **Administrator** with the **version** permission and **CHANGE\_LOCATION** permission

Following the "edit" link relation on permission set representation, the client can invoke PUT method to update the permission set. The client must provide a complete permission set object in the request body, including changing parts and unchanged parts. When any property is null or empty, the property's value will be reset to null.

### Request

```

01. PUT /dctm-rest/repositories/REPO/objects/0900000580004ba5/permission-set HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==
04. Content-Type: application/vnd.emc.documentum+json
05.
06. {
07.     "permitted": [
08.         {
09.             "accessor": "dm_world",
10.             "basic-permission": "Read",
11.             "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
12.         },
13.         {
14.             "accessor": "docu",
15.             "basic-permission": "Write"
16.         }
17.     ],
18.     "restricted": [
19.         {
20.             "accessor": "Administrator",
21.             "basic-permission": "Version",
22.             "extend-permissions": "CHANGE_LOCATION"
23.         }
24.     ]
25. }

```

### Response

```

01. {
02.     "permitted": [
03.         {
04.             "accessor": "dm_world",
05.             "basic-permission": "Read",
06.             "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
07.         },
08.         {
09.             "accessor": "dm_owner",
10.             "basic-permission": "Delete",
11.             "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
12.         },
13.         {
14.             "accessor": "docu",
15.             "basic-permission": "Write",
16.             "extend-permissions": "EXECUTE_PROC,CHANGE_LOCATION"
17.         }
18.     ],
19.     "restricted": [
20.         {
21.             "accessor": "Administrator",
22.             "basic-permission": "Version",
23.             "extend-permissions": "CHANGE_LOCATION"
24.         }
25.     ]

```

```

26.     "links": [
27.         {
28.             "rel": "self",
29.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004ba5/permis
30.         },
31.         {
32.             "rel": "edit",
33.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004ba5/permis
34.         },
35.         {
36.             "rel": "http://identifiers.emc.com/linkrel/acl",
37.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580002195"
38.         }
39.     ]
40. }

```

There are two built-in accessors in *permitted* section: *dm\_world* (all users) and *dm\_owner* (the owner of the object). Even if the REST client does not specify any access level for *dm\_world* and *dm\_owner*, Content Server grants them with the default permissions automatically. Thus, we see there is a built-in accessor *dm\_owner* in response body, though it is not provided in the request body.

The result of the update leads to a new ACL object created and applied on the document. In the sample, the updated permission set derives from the ACL object with id "4500000580002195".

## View permissions for specified user or group

Next, we will check the calculated permissions that specific user or group has on this document.

Following the link relation "<http://identifiers.emc.com/linkrel/permissions>" on the document, the client invokes a GET method to get the calculated permissions for the document. The client can get the calculated basic and extend permissions on this document for a specific accessor. When no accessor is specified, the default accessor is the current login user.

In this sample, we get the document permission for user *Administrator*. Please note user *Administrator* is within group *docu*.

### Request

```

01. GET /dctm-rest/repositories/REPO/objects/0900000580004ba5/permissions?accessor=Administrator HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==

```

### Response

```

01. {
02.     "accessor": "Administrator",
03.     "basic-permission": "Relate",
04.     "extend-permissions": "EXECUTE_PROC",
05.     "links": [
06.         {
07.             "rel": "self",
08.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/0900000580004ba5/permis
09.         }
10.     ]
11. }

```

Please recall how we updated the document's permission set.

- the group *docu* is granted with *Write* basic permission and *EXECUTE\_PROC, CHANGE\_LOCATION* extend permissions in *permitted* section,
- the user *Administrator* is restricted to access to this document with *Version* basic permission and *CHANGE\_LOCATION* extend permission in restricted section

As the basic permit is hierarchical and the upper level basic permit of accessor *Administrator* is restricted to *Version*, now user *Administrator's* basic permission should be **Relate**. Also user *Administrator* should only have **EXECUTE\_PROC** extend permission on this document, as the extend permission **CHANGE\_LOCATION** is restricted.

## ACL(s) Management Samples

The table describes the user stories for ACL(s) management we are going to explain in the next part.

Story Name	User	Intention
Get ACLs collection	dave	Gets all available ACLs in the repository
Navigate an ACL	dadmin	Gets a specified ACL object in the repository
Create an ACL	dave	Creates a new ACL in the repository
Edit an ACL	dave	Updates the specific ACL
Get an ACL associations	dave	Cets associated SysObjects of the specified ACL
Force delete an ACL	dadmin	Force destroy the specific ACL

### Get ACLs collection

Assumed that you have got a repository resource by user 'dave'.

#### Request

```
01. GET /dctm-rest/repositories/REPO/ HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==
```

From the repository resource representation, you can find the ACL feed link relation “<http://identifiers.emc.com/linkrel/acls>”, which points to all available ACL objects in the repository. This feed allows you to create a new ACL object in the repository.

#### Response

```
01. {
02.   "id": 5,
03.   "name": "REPO",
04.   "description": "",
05.   "servers": [{.....}],
06.   "links": [
07.     {
08.       "rel": "self",
09.       "href": "http://localhost:8080/dctm-rest/repositories/REPO/"
10.     },
11.     .....
12.     {
13.       "rel": "http://identifiers.emc.com/linkrel/acls",
14.       "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls"
15.     },
16.     .....
17.   ]
18. }
```

By invoking a GET method on the URI of the “<http://identifiers.emc.com/linkrel/acls>” link relation, you can get all available ACLs in the repository.

#### Request

```
01. GET /dctm-rest/repositories/REPO/acls HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==
```

The ACLs resource lists all ACLs in a flatten feed.

#### Response

```

01. {
02.   "id": "http://localhost:8080/dctm-rest/repositories/REPO/acls",
03.   "title": "ACLs",
04.   "author": [
05.     {
06.       "name": "EMC Documentum"
07.     }
08.   ],
09.   "updated": "2016-01-21T06:02:28.478+00:00",
10.   "page": 1,
11.   "items-per-page": 100,
12.   "links": [
13.     {
14.       "rel": "self",
15.       "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls"
16.     },
17.     .....
18.   ],
19.   "entries": [
20.     {
21.       "id": "4500000580000100",
22.       "title": "dm_4500000580000100",
23.       "updated": "2016-01-21T06:02:28.525+00:00",
24.       "published": "2016-01-21T06:02:28.525+00:00",
25.       "links": [
26.         {
27.           "rel": "edit",
28.           "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000100"
29.         }
30.       ],
31.       "content": {
32.         "type": "application/vnd.emc.documentum+json",
33.         "src": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000100"
34.       }
35.     },
36.     .....
37.   ]
38. }

```

To get the full representation of individual ACL object in a feed, you could append a URI query parameter "inline=true" on the "<http://identifiers.emc.com/linkrel/acls>" resource URI.

## Navigate an ACL

From the ACLs feed response, there is an "edit" link relation for every entry in the feed, following the "edit" link, you can navigate to the specific ACL object. In order to get all link relations of the ACL object, we authenticate with super user 'dmadmin'.

### Request

```

01. GET /dctm-rest/repositories/REPO/acls/4500000580000100 HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZG1hZG1pbjpwYXNzd29yZA==

```

The response body contains the ACL properties and link relations, there are two *privilege aware* link relations: "edit" and "<http://identifiers.emc.com/linkrel/delete>". These two links only appear to the users who have sufficient privileges to update or delete this ACL object. In other words, these two links only appear to the ACL object owner or super users. In this sample, as we assumed the user "dmadmin" is a super user, we can see these two links.

The "<http://identifiers.emc.com/linkrel/associations>" link relation points to the SysObjects that this ACL is applied on.

### Response

```

01. {
02.   "name": "acl",

```



```

03.     "type": "dm_acl",
04.     "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl",
05.     "properties": {
06.         "object_name": "dm_4500000580000100",
07.         "description": "dm_4500000580000100",
08.         "owner_name": "REPO_ADMIN",
09.         "r_object_id": "4500000580000100",
10.         "r_accessor_name": ["dm_world", "dm_owner", "docu"],
11.         "r_accessor_permit": [3,7,5],
12.         "r_permit_type": [0,0,0],
13.         "r_accessor_xpermit": [0,0,3],
14.         "r_application_permit": ["", "", ""],
15.         "r_is_group": [false, false, true],
16.         "globally_managed": false,
17.         "acl_class": 0,
18.         .....
19.     },
20.     "links": [
21.         {
22.             "rel": "self",
23.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000100"
24.         },
25.         {
26.             "rel": "http://identifiers.emc.com/linkrel/associations",
27.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000100/associati
28.         },
29.         {
30.             "rel": "edit",
31.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000100"
32.         },
33.         {
34.             "rel": "http://identifiers.emc.com/linkrel/delete",
35.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580000100"
36.         }
37.     ]
38. }

```

- The ACL object contains six repeating attributes: *r\_accessor\_name*, *r\_accessor\_permit*, *r\_accessor\_xpermit*, *r\_permit\_type*, *r\_application\_permit* and *r\_is\_group*, these six attributes' values match to each other according to the corresponding index position, the values at the same index of all repeating attributes define the permit type and permissions( basic, extend and application permissions) for a specified user or group.
- In this sample, the first index values define following accessor permissions:

Attributes Name	Attributes Value at the First Index
r_accessor_name	dm_world
r_is_group	false
r_accessor_permit	3
r_accessor_xpermit	0
r_application_permit	NULL
r_permit_type	0

Please refer to Content Server System Object Reference guide for the attributes detail explanation.

## Create an ACL

Following the "<http://identifiers.emc.com/linkrel/acls>" link relation, you can create a new ACL in the repository. The request body is the ACL object that you want to be created, and the repeating attributes should have equivalent numbers of values. You can also post an empty request body to create a system default ACL.

### Request

```

01. POST /dctm-rest/repositories/REPO/acls HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==
04. Content-Type: application/vnd.emc.documentum+json
05.
06.
07. {
08.   "name": "acl",
09.   "type": "dm_acl",
10.   "properties": {
11.     "object_name": "TEST_ACL_0",
12.     "description": "test_acl",
13.     "owner_name": "dave",
14.     "r_accessor_name": ["Administrator","dmdadmin"],
15.     "r_accessor_permit": [5,6],
16.     "r_accessor_xpermit": [0,0],
17.     "r_permit_type": [0,0 ],
18.     "globally_managed": false,
19.     "acl_class": 0
20.   }
21. }
```

### Create an ACL Notes

The ACL object includes five editable repeating

attributes: *r\_accessor\_name*, *r\_accessor\_permit*, *r\_accessor\_xpermit*, *r\_permit\_type*, *r\_application\_permit*, and there are some constraints between these five repeating attributes' value when we create or edit an ACL object.

1. the length of all provided repeating attributes' value should be equal, as the five attributes' values match to each other according to the corresponding index position
2. if the *r\_permit\_type* value is set to either 6 or 7, the value in *r\_accessor\_name* at the corresponding index position must be a group name.

Every ACL has two built-in accessors: *dm\_world* and *dm\_owner*, even if the REST client does not specify any access level for *dm\_world* and *dm\_owner*, Content Server grants them with the default permissions automatically.

We see there are only two values for each repeating attributes in the request body, but in the response body there are four values, as the two build-in accessors generated automatically by Content Server.

### Response

```

01. {
02.   "name": "acl",
03.   "type": "dm_acl",
04.   "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl",
05.   "properties": {
06.     "object_name": "TEST_ACL_0",
07.     "description": "test_acl",
08.     "owner_name": "dave",
09.     "r_object_id": "4500000580001d02"
10.     "r_accessor_name": ["dm_world","dm_owner","Administrator","dmdadmin"],
11.     "r_accessor_permit": [3,7,5,6],
12.     "r_accessor_xpermit": [0,0,0,0],
13.     "r_is_group": [false,false,false,false],
14.     "r_permit_type": [0,0,0,0],
15.     "r_application_permit": ["","","",""],
16.     .....
17.   },
18.   "links": [
19.     {
20.       "rel": "self",
```

```

21.         "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580001d02"
22.     },
23.     .....
24. ]
25. }

```

## Edit an ACL

When you are the ACL owner or a supper user, you will see the “**edit**” link relation in the ACL representation. It means you get the privilege to update this ACL. Using POST method to update the attributes on the URI from the “**edit**” link relation.

Now user 'dave' tries to update the above created ACL object.

### Request

```

01. POST /dctm-rest/repositories/REPO/acls/4500000580001d02 HTTP/1.1
02. Host: localhost:8080
03. Content-Type: application/vnd.emc.documentum+json
04. Authorization: Basic ZGF2ZTpwYXNzd29yZA==
05.
06. {
07.     "properties": {
08.         "description": "update_test_acl_by_owner",
09.         "r_accessor_name": [
10.             "dm_world",
11.             "dm_owner",
12.             "Administrator",
13.             "dmadmin",
14.             "dave"
15.         ],
16.         "r_accessor_permit": [3,7,5,6,7],
17.         "r_accessor_xpermit": [0,0,3,0,0],
18.         "r_permit_type": [0,0,1,0,0],
19.         "r_application_permit":["","","","",""]
20.     }
21. }

```

### Edit an ACL Notes

1. Only modifiable attributes(*description*, *acl\_class*, *r\_access\_name*, *r\_access\_permit*, *r\_accessor\_xpermit*, *r\_application\_permit*, *r\_permit\_type*) SHOULD be put in the message body
2. An ACL template instance(*acl\_class*=2) can not be updated directly, neither change an regular(*acl\_class*=0), public(*acl\_class*=3) or template(*acl\_class*=1) ACL object to an template instance.
3. The value length of all provided repeating attributes should be equal, if only parts of the five editable repeating attributes provided, then their length should also be equal to the ACL original accessor account.

Then we can get the updated ACL object from the response body.

### Response

```

01. {
02.     "name": "acl",
03.     "type": "dm_acl",
04.     "definition": "http://localhost:8080/dctm-rest/repositories/REPO/types/dm_acl",
05.     "properties": {
06.         "object_name": "TEST_ACL_0",
07.         "description": "update_test_acl_by_owner",
08.         "owner_name": "dave",
09.         "r_object_id": "4500000580001d02"
10.         "r_accessor_name": [
11.             "dm_world",
12.             "dm_owner",
13.             "Administrator",

```

```

14.         "dmadmin",
15.         "dave"
16.     ],
17.     "r_accessor_permit": [3,7,0,6,7],
18.     "r_accessor_xpermit": [0,0,0,0,0],
19.     "r_is_group": [false,false,false,false,false],
20.     "r_permit_type": [0,0,0,0,0],
21.     "r_application_permit": ["", "", "", "", ""],
22.     .....
23. },
24. "links": [
25.     {
26.         "rel": "self",
27.         "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580001d02"
28.     },
29.     {
30.         "rel": "http://identifiers.emc.com/linkrel/associations",
31.         "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580001d02/associati
32.     },
33.     .....
34. ]
35. }

```

## Get an ACL associations

An ACL is used to impose object-level permissions on SysObjects in Content Server. Each SysObject has an ACL and the ACL assigned to a SysObject is used to control access to that object. From ACL aspect, the ACL associations resource allows REST client to get all available SysObjects that an specified ACL is assigned on.

Following the link relation "<http://identifiers.emc.com/linkrel/associations>" in ACL representation, the client can get all available SysObjects that associated with this specified ACL.

### Request

```

01. GET /dctm-rest/repositories/REPO/acls/4500000580001d02/associations HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==

```

No entry in the response feed is expected, as the above newly created ACL is not assigned on any SysObjects in repository.

### Response

```

01. {
02.     "id": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580001d02/associations",
03.     "title": "Objects",
04.     "author": [
05.         {
06.             "name": "EMC Documentum"
07.         }
08.     ],
09.     "updated": "2016-01-22T07:27:00.603+00:00",
10.     "page": 1,
11.     "items-per-page": 100,
12.     "links": [
13.         {
14.             "rel": "self",
15.             "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580001d02/associati
16.         }
17.     ]
18. }

```

Now let's create a document under user *dave*'s home cabinet with the above created ACL object assigned to the new document.

### Request

```

01. POST /dctm-rest/repositories/REPO/folders/0c00000580001909/documents HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZGF2ZTpwYXNzd29yZA==
04. Content-Type: application/vnd.emc.documentum+json
05. {
06.   "properties":{
07.     "object_name":"REST-API-DOC-WITH-SPECIFIED-ACL",
08.     "r_object_type":"dm_document",
09.     "acl_name":"TEST_ACL_0",
10.     "acl_domain":"dave"
11.   }
12. }

```

- There are two properties `acl_name` and `acl_domain` to identify an ACL object in a SysObject, and the `acl_name` and `acl_domain` correspond to properties `object_name` and `owner_name` in ACL object.

After the document is successfully created, we can get the ACL associated SysObjects again, and it is expected to see one entry in the result feed, which is the exact document we just created.

### Response

```

01. {
02.   "id": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580001d02/associations",
03.   "title": "Objects",
04.   "author": [
05.     {
06.       "name": "EMC Documentum"
07.     }
08.   ],
09.   "updated": "2016-01-22T07:44:51.822+00:00",
10.   "page": 1,
11.   "items-per-page": 100,
12.   "links": [
13.     {
14.       "rel": "self",
15.       "href": "http://localhost:8080/dctm-rest/repositories/REPO/acls/4500000580001d02/associati
16.     }
17.   ],
18.   "entries": [
19.     {
20.       "id": "http://localhost:8080/dctm-rest/repositories/REPO/objects/090000058000451a",
21.       "title": "REST-API-DOC-WITH-SPECIFIED-ACL",
22.       "author": [
23.         {
24.           "name": "dave",
25.           "uri": "http://localhost:8080/dctm-rest/repositories/REPO/users/dave"
26.         }
27.       ],
28.       "summary": "dm_document 090000058000451a",
29.       "updated": "2016-01-22T07:14:20.000+00:00",
30.       "published": "2016-01-22T07:14:20.000+00:00",
31.       "links": [
32.         {
33.           "rel": "edit",
34.           "href": "http://localhost:8080/dctm-rest/repositories/REPO/objects/090000058000451a
35.         }
36.       ],
37.       "content": {
38.         "type": "application/vnd.emc.documentum+json",
39.         "src": "http://localhost:8080/dctm-rest/repositories/REPO/objects/090000058000451a"
40.       }

```

```

41.     }
42.   ]
43. }

```

## Force delete an ACL

The REST client could invoke DELETE method to the link relation "<http://identifiers.emc.com/linkrel/delete>" in an ACL representation to delete the specified ACL object when the accessor is the ACL owner or a super user.

If the ACL has been assigned to any SysObjects in the repository, the client will fail to delete this ACL and a status 409 will be returned by the server. The super users can forcibly destroy the applied ACL by appending query parameter "force=true", even if the ACL has associated documents.

- **Delete ACL notes**

If the client succeed to force destroy an ACL, then the ACL associated SysObjects will have no referenced ACL, these SysObjects will become un-modifiable, and only the objects owner or super users can read them, and can only be deleted by super users.

In this sample, we try to use super user 'dmadmin' to force destroy the above assigned ACL.

### Request

```

01. DELETE /dctm-rest/repositories/REPO/acls/4500000580001d02?force=true HTTP/1.1
02. Host: localhost:8080
03. Authorization: Basic ZG1hZG1pbjpwYXNzd29yZA==

```

## ACL and Permission Set Comparison

The ACL object resource models the "dm\_acl" object as a regular persistent object. The permission set resource derives from "dm\_acl" object and contains the ACL entries part only. The permission set resource representation is comprehensive comparing to the raw ACL object resource and is easier for permission update. In other words, we can say that the permission set object is the editorial representation of an dm\_acl object.

The below table summaries the raw ACL property definition in Content Server.

Property Name	Description
r_accessor_name	List of users that have access to the object attached to the ACL. Valid entries in the list are individual user names, group names, and aliases. This must be or resolve to a group name if the r_permit_type is RequiredGroup or RequiredGroupSet.
r_accessor_permit	Specifies the access level granted to the user or group at the corresponding index level. Valid values are: • 0: Null • 1: None • 2: Browse • 3: Read • 4: Relate • 5: Version • 6: Write • 7: Delete
r_accessor_xpermit	Specifies the extended permission level granted to the user or group at the corresponding index level. Extended permissions are: • execute_proc • change_location • change_state • change_permit • change_owner • delete_object • change_folder_links
r_application_permit	Specifies a permission recognized by an application. An application permit is not recognized or enforced by Content Server. The application is responsible for enforcing the permission. The permission defined at a particular index position is applicable to the user or group identified in the corresponding position in r_accessor_name.
r_permit_type	Specifies the permit type for the entry. Valid values are: • 0: AccessPermit • 1: ExtendedPermit • 2: ApplicationPermit • 3: AccessRestriction • 4: ExtendedRestriction • 5: ApplicationRestriction • 6: RequiredGroup • 7: RequiredGroupSet To set this value to either 6 or 7, the value in r_accessor_name at the corresponding index position must be a group name.

The below table describes the property mapping between ACL entry and permission set object.

ACL property	ACL property value type	Permission Set property	Permission Set property value type
r_accessor_name	string	accessor	string

ACL property	ACL property value type	Permission Set property	Permission Set property value type
r_accessor_permit	integer (0,1,2,3,4,5,6,7)	basic-permission	string (Null,None,Browse,Read,Relate,Version,Write,Delete)
r_accessor_xpermit	integer	extend-permissions	string
r_application_permit	string	application-permission	string
r_permit_type	integer	<b>permitted:</b> when r_permit_type value is 0,1 or 2 <b>restricted:</b> when r_permit_type value is 3,4 or 5 <b>required-group:</b> when r_permit-type value is 6 <b>required-group-set:</b> when r_permit_type is 7	string

From above table, we know the *r\_accessor\_name*, *r\_application\_permit* in an ACL entry match the *accessor*, *application-permission* in an accessor permission, and the *r\_accessor\_permit*, *r\_accessor\_xpermit* integer type values in ACL entry match the *basic-permission*, *extend-permissions* with corresponding string value in an accessor permission. And the value of *r\_permit\_type* in an ACL entry decides in which section this accessor permission should be contained, such as if the *r\_permit\_type* is 0,1 or 2, this accessor permission should be included in *permitted* section of the permission set object.

In the following table, we use an example to compare the representation between ACL and permission set resource.

ACL Representation	Permission Set Representation
--------------------	-------------------------------

ACL Representation	Permission Set Representation
<pre> 01. { 02.   "properties": { 03.     "r_accessor_name": [ 04.       "dm_world", 05.       "dm_owner", 06.       "docu", 07.       "Administrator" 08.     ], 09.     "r_accessor_permit": [ 10.       3, 11.       7, 12.       6, 13.       5 14.     ], 15.     "r_accessor_xpermit": [ 16.       0, 17.       0, 18.       0, 19.       2 20.     ], 21.     "r_application_permit": [ 22.       "", 23.       "", 24.       "", 25.       "" 26.     ], 27.     "r_permit_type": [ 28.       0, 29.       0, 30.       0, 31.       3 32.     ], 33.     ..... 34.   }, 35.   "links": [ 36.     ..... 37.   ] 38. } </pre>	<pre> 01. { 02.   "permitted": [ 03.     { 04.       "accessor": "dm_world", 05.       "basic-permission": "Read", 06.       "extend-permissions": "EXECUTE_PROC,CHANGE_LC 07.     }, 08.     { 09.       "accessor": "dm_owner", 10.       "basic-permission": "Delete", 11.       "extend-permissions": "EXECUTE_PROC,CHANGE_LC 12.     }, 13.     { 14.       "accessor": "docu", 15.       "basic-permission": "Write", 16.       "extend-permissions": "EXECUTE_PROC,CHANGE_LC 17.     } 18.   ], 19.   "restricted": [ 20.     { 21.       "accessor": "Administrator", 22.       "basic-permission": "Version", 23.       "extend-permissions": "CHANGE_LOCATION" 24.     } 25.   ], 26.   "links": [ 27.     ..... 28.   ] 29. } </pre>

In an ACL object, the combination of all repeating attributes at the same index position makes up an ACL entry rule:

- r\_accessor\_name (dm\_world),
- r\_accessor\_permit (3),
- r\_accessor\_xpermit (0),
- r\_application\_permit (""),
- r\_permit\_type (0)

According to the mapping between the ACL and the permission set, we can get the following accessor permission in permission set object:

- accessor (dm\_world)
- basic-permission (Read)
- extend-permissions (EXECUTE\_PROC,CHANGE\_LOCATION)

## Summary

This tutorial is an introduction for ACL and permission set in REST Services. After learning from this tutorial, you get to know:

- ACL is used to define object-level security for SysObjects in repository, ACLs and ACL resources model *dm\_acl* objects as a regular persistent object with CRUD operations. It is usually an admin role user to uses this



API.

- The permission set resource derives from the "*dm\_acl*" object. It provides the intuitive representation for permission management on SysObjects.

[Learn more about Documentum REST Services >>](#)