opentext™

OpenText® Documentum® D2FS REST Services

opentext™

Abstract

This document is a guide to using OpenText Documentum D2FS REST Services to utilize the D2 Configurations on Documentum Repositories.

opentext™

Table of Contents

opentext™

What's changed in 23.2?

Please refer to the below sections and their content for what's changed in 23.2.

1. Added new section: "View Permission".
2. Updated the section "Common Definition – Query Parameters".
3. Updated the subsection "Request Body" under section "Object Creation".
4. Updated subsection 5 "Creating Digital Signature Request" under section "Digital Signature".
5. Updated subsection 3. "Validation API" under section "Digital Signature".

Overview

This document is a guide to using OpenText D2FS REST services.

Documentum D2 provides two clients: D2 Client and D2-Config. D2 Client is a web-based application that gives users the ability to interact with content in one or more repositories. D2-Config is the administration client of Documentum D2.

D2FS REST services allows a REST client to utilize configurations that are defined in D2-Config. This document is intended for developers and architects who are building clients for D2FS REST Services.

What are D2FS REST Services?

OpenText D2FS REST Services are a set of RESTful web service interfaces that interact with D2 Configurations and honor them as required. Being developed in a purely RESTful style, D2FS REST Services provides high efficiency and simplicity when programming and makes all services easy to consume. These advantages make D2FS REST Services the best choice for next generation applications and mobile applications to interact with the D2 Configurations.

D2FS REST Services are written as part of the extensibility feature of the OpenText Documentum Platform REST services (also called Documentum REST services). Hence, the REST services provided by Documentum REST services and D2FS co-exist in the D2 server space. Currently, D2FS REST services, which have corresponding link relations and URLs in Documentum REST, are identified by the suffix "-d2" in their link relations and URL over the existing resources provided by Documentum REST services. For any new service which is specific to D2FS, the link relation may not carry the suffix.

As with Documentum REST services, D2FS REST Services models objects in Documentum repositories as resources and identifies resources by Uniform Resource Identifiers (URIs). It

defines specific media types to represent resources and drives application state transfers by using link relations. It uses a limited number of HTTP standard methods (GET, PUT, POST, and DELETE) to manipulate resources over the HTTP protocol. D2FS REST Services supports the JSON and XML format for resource representation. D2FS REST Services does not introduce any new media types and only uses the existing types that are available as part of the Documentum REST services.

Consuming D2FS REST Services

D2 REST Services delivers a deployable Java web archive (WAR) that runs in a web container of a Java EE application server (refer to release notes for system requirements). D2 REST Services exposes the interface as network-accessible resources identified by URIs. D2 REST Services is programming language independent, therefore, you can consume the services by using any language that has a HTTP client library, such as Java, .NET, Python, or Ruby. Because of these features, D2 REST Services doesn't ship any kind of client or SDK (Software Development Overview Kit) to the users. You can freely develop the REST client to consume the REST service if you follow hypertext-driven principles.

Relationship with other Documentum Platform APIs

D2 REST Services rely on the D2FS library to perform operations based on the D2 Configurations available in the Documentum Content Server. The D2FS library uses DFC and communication between the REST server and Content Server is conducted over Netwise RPC. D2 REST Services is a lightweight alternative to the existing D2FS SOAP Services. However, it is not intended to provide the equivalent functionalities in the initial release. You can leverage the simplicity of RESTful services to achieve highly productive programming.

General REST Definitions

Common Definition - HTTP Headers

D2 REST Services supports the following common HTTP headers.

Http Header Name	Description	In Request or Response	Value Range
Authorization	Authorization header for authentication	Request	HTTP basic authentication header with the credential part encoded, for example: Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ== Or, Kerberos authentication header with the credential part encoded, for example: Authorization: Negotiate YIIZG1hZG1pbjpwYXNzd29yZ...
Accept	Acceptable media type for the response	Request	See the topic, "Supported MIME Types."
Content-Type	MIME type of the request body or response body	Request /Response	See Supported MIME Types. The REST server ignores the charset parameter in the Content-Type header.
Location	URI of the newly-created resource	Response	URI
Content-Length	Size of the entity-	Request	Non-negative number

	body, in decimal number of OCTETs, sent to the recipient	/Response	

Common Definition – Query Parameters

D2 supports the following common query parameter:

- Inline

The rest of the query params remain unsupported as the extensibility framework does not have options to consume an existing collection.

D2-REST only supports date time input in ISO format (yyyy-MM-dd'T'HH:mm:ss.SSSZZ or yyyy-MM-dd'T'HH:mm:ss.SSSXXX). For example: 2023-02-09T10:15:00.000+0530. Append the time zone with date and time as shown in the example.

Please refer to the "Common Definition - Query Parameters" section of the Documentum Platform REST services – Developer Guide for detailed information on query parameters. Their purpose and meaning are described in the Developer Guide.

HTTP Status Codes

Please refer to the "HTTP Status Codes" section of the Documentum Platform REST services – Developer Guide for detailed information on list of status codes that will be returned by the services. Their purpose and meaning are described in the Developer Guide.

Supported MIME Types

A media type (also called a content type or MIME type) is a short string identifying the format of a document. Once you know a document's media type, you can parse it. As described in the D2FS REST services section, the services do not introduce any new MIME types--they use the ones that are already available with the Documentum Platform REST services. Please refer to the "Supported MIME Types" section of the Documentum Platform REST services – Developer Guide for detailed information on the list of supported MIME types. Their purpose and meaning are described in the Developer Guide.

HTTP Methods

Documentum D2FS REST Services supports the following HTTP methods:

- GET

Use this method to retrieve a representation of a resource.

- POST

Use this method to create new resources or update existing resources.

- DELETE

Use this method to delete a resource.

Link Relations

The D2FS REST service is a hypermedia driven API. Link relation forms the basis of hypermedia driven APIs. The key purpose of a link relation type is to identify the semantics associated with the link. The client chooses one of the available link relations for a state transition—either an application or a resource.

D2FS introduces new link relations to access the D2 specific resources and to consume the existing D2 Configurations.

Clients are always required to use the link relations that are provided by the server for state transition. The link relations are immutable and the links("href") are constructed at runtime, depending on the client's application state.

Link relations are not URIs. The URI that they refer to provide information on how to use these link relations. Clients need to refer to the actual links "href" associated with the link relations to achieve state transition of resources.

Public Link Relation Registry

The Internet Assigned Numbers Authority (IANA) maintains the public link relations registry here: http://www.iana.org/assignments/link-relations. The following table describes the link relations that are used within D2FS REST Services and includes links to the detailed specifications.

Link Relation	Description	Specification
Edit	Points to a resource that can be used to edit the link's context.	http://tools.ietf.org/html/rfc5023
self	Conveys an identifier for the link's context.	http://www.ietf.org/rfc/rfc4287

opentext™

D2FS Link Relations

The following table includes link relations defined within OpenText for use by D2FS REST Services.

Link Relation	Description	Application State
http://identifiers.opentext.com/linkrel/ creation-profiles	Provides a href to the Creation Profiles resource. This resource allows you access the list of creation profiles configured in D2-Config.	After the user logins into a repository space.
http://identifiers.opentext.com/linkrel/ creation-profile	Provides a href to the Creation Profile resource. This resource allows you access to a specific profile configured in D2-Config.	After the user gets into the list of creation profiles.
http://identifiers.opentext.com/linkrel/ type-configurations	Provides a href to the list of types defined within the selected creation profile in D2 Config.	After the user gets into a specific creation profile.
http://identifiers.opentext.com/linkrel/ type-configuration	Provides a href to a specific type within the selected creation profile in D2 Config.	After the user gets into the list of types for a specific profile configuration.
http://identifiers.opentext.com/linkrel/ object-creation	Provides a href to create an object honoring the D2 Configuration provided.	After the user logins into a repository space.
http://identifiers.opentext.com/linkrel/ document-templates	Provides a href to the Templates resource. This resource allows you access the list of template documents configured in D2-Config.	After the user gets into a specific document/object.
http://identifiers.opentext.com/linkrel/ document-template	Provides a href to the Template resource. This resource allows you access the specific template configured in D2-Config.	After the user gets into a list of template documents.

http://identifiers.opentext.com/linkrel/comments	Provides a href to the Comments resource. This resource allows you access the list of comments on the document.	After the user gets into a specific document/object.
http://identifiers.opentext.com/linkrel/comment	Provides a href to the Comment resource. This resource allows you to access a specific comment.	After the user gets into a list of comments.

The link relations appear in the relevant application state within the context of the client.

The below link relations will exist until the Core REST extensibility is in place. Once, extensibility is used in D2FS REST services, then the below link relations will cease to exist.

Link Relation	**Description**	**Application State**
http://identifiers.opentext.com/linkrel/ object-creation	Provides a href to create an object honoring the D2 Configuration provided. This will be overriding core REST link relations in Document Resource, Folder Child Object Resource where creation of object is applicable. Refer below for more details.	After the user logins into a repository space.
http://identifiers.opentext.com/linkrel/checkout/	Provides a href to checkout/lock this object. This will be overriding core REST link relation which is applicable for checkout/lock.	After the user gets into a specific document/object.
http://identifiers.opentext.com/linkrel/cancel-checkout	Provides a href to cancel checkout/unlock this object. This will be overriding core REST link relation which is applicable for cancel checkout/unlock.	After the user gets into a specific document/object.
http://identifiers.opentext.com/linkrel/checkin-next-major	Provides a href to check-in next major version of this object. This will be overriding core REST link relation which is applicable for check-in next	After the user gets into a specific document/object.

	major version.	
http://identifiers.opentext.com/linkrel/checkin-next-minor	Provides a href to checkin next minor version of this object. This will be overriding core REST link relation which is applicable for check-in next minor version.	After the user gets into a specific document/object.
http://identifiers.opentext.com/linkrel/checkin-branch	Provides a href to checkin branch version of this object. This will be overriding core REST link relation which is applicable for check-in branch version.	After the user gets into a specific document/object.
http://identifiers.opentext.com/linkrel/checkin-same	Provides a href to checkin same version of this object. This will be overriding core REST link relation which is applicable for check-in same version.	After the user gets into a specific document/object.
http://identifiers.com/com/linkrel/preview-urls	Provides a href to the preview urls resource. This resource allows you access the list of preview urls on the document. This will be overriding core REST link relation which is applicable for Content/Contents Resource.	After the user gets into a specific document/object.

The link relation (http://identifiers.opentext.com/linkrel/object-creation) which is used for object creation will override link relations such as: http://identifiers.opentext.com/linkrel/objects, http://identifiers.opentext.com/linkrel/documents, and other relations where object creation is applicable.

opentext™

Explore the D2FS REST services

This section provides a sample which guides you through how to use the D2FS REST services.

This sample uses JSON representation and collection pattern to represent feeds. This assumes that the D2FS REST service is deployed in localhost:8080 and the client is a web-browser capable of rendering JSON results.

Common Tasks

The common tasks that are currently available in the D2FS REST services is to:

A) Understand the list of creation profiles that are available for a user.

B) Understand the list of types that are available within a specific profile.

C) Understand the configurations present on a specific type.

D) Create an object of a specific type that honors the D2-Configurations.

E) Understand the list of templates available for an object.

E) View/add new/delete existing comments to an existing object in the repository.

Resources

Organization of Resource Reference Documentation

These guidelines apply to each resource reference documentation entry:

- All method parameters are optional unless otherwise noted.
- DELETE and GET methods do not have a request body.

Profiles

The profiles resource represents a collection of all the profiles that are available in the context of the user.

URI Template

/repositories/ {repositoryName}/ profile-configuration

repositoryName	Name of the repository

Feed

Feed Id	Feed Title	Updated	Entry	Supports POST or not
URI of the Creation Profiles resource without the file extension	List of D2 Creation Profiles	Server's current time	Profile	NO

Entry Id	Entry Title
URI of the Creation Profile	Profile Name

Link Relation

Link Relation	Description
self	URI of the Creation Profile collection feed.

Operations

Method	Description
GET	Lists all the creation profiles in the repository for the current user's context.

Get Profiles-D2

Lists all the creation profiles in the repository for the current user's context.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

- Accept
- Authorization
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

None

Response Headers

- Content-Length
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing

Response Status

200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
403 - Permission denied
500 - Other unexpected server error

Response Body

This is a XML or JSON representation of the D2 Creation Profiles collection.

- The body contains a list of D2 Creation Profiles.
- Each object may contain all properties of the creation profile, depending on the setting of the query parameter (inline = true).
- The returned child objects collection only contains those that the user has access to.
- Each profile entry must contain link that point to a specific profile.

opentext™

Profile

The profile resource represents a creation profile in the repository.

URI Template

/repositories/ { repositoryName }/ profile-configuration /{profileId}

repositoryName	Name of the repository
profileId	Object ID of the Creation Profile

Link Relation

Link Relation	Description
self	Link to this creation profile
type-configuration [1]	Collection of types that are defined as part of this profile

[1] This link relation is defined by Documentum. The fully qualified Documentum link relation path is prefixed with the following string: http://identifiers.opentext.com/linkrel/

Operations

Method	Description
GET	Retrieves properties, and other information of the Creation profile resource.

Get a Profile

Gets properties and other information of this Creation Profile. Properties are returned as embedded elements in the response message body. Other information, such as types, is referenced from the link relations of the response message body.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

- Accept
- Authorization
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

None.

Response Headers

- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing)

Response Status

200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
404 – D2 Creation Profile not found.
500 - Other unexpected server error

Response Body

This is a XML or JSON representation of the D2 Creation Profile.

Search Configurations

The search configurations resource represents a collection of all the search configurations that are available in the context of the user.

URI Template

/repositories/ {repositoryName}/ search-configuration

repositoryName	Name of the repository

Feed

Feed Id	Feed Title	Updated	Entry	Supports POST or not
URI of the Search Configurations resource	List of D2 Search Configurations	Server's current time	Search Configuration	NO

Entry Id	Entry Title
URI of the Search Configuration	Search Configuration name

Link Relation

Link Relation	Description
self	URI of the Search Configuration collection feed

Operations

Method	Description
GET	Lists all the search configurations in the repository for the current user's context.

Get Search Configurations

Lists all the search configurations in the repository for the current user's context.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

- Accept
- Authorization
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

None.

Response Headers

- Content-Length
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing

Response Status

200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
403 - Permission denied
500 - Other unexpected server error

Response Body

This is a XML or JSON representation of the D2 Search Configurations collection.

- The body contains a list of D2 Search Configurations.
- Each object may contain all properties of the search configuration, depending on the setting of the query parameter (inline = true).
- The returned child objects collection only contains those that the user has access to.
- Each search configuration entry must contain link that point to a specific configuration.

Search Configuration

The search configuration resource represents a search configuration in the repository.

URI Template

/repositories/ { repositoryName }/ search-configuration /{configId}

repositoryName	Name of the repository
configId	Object ID of the Search Configuration

Link Relation

Link Relation	Description
self	Link to this search configuration

Operations

Method	Description
GET	Retrieves the types names, attribute names, facets and other data associated with a D2 Search Configuration.

Get a Search Config

Gets types, attributes, and other information of the Search Configuration. The data is returned as embedded elements in the response message body.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

- Accept
- Authorization
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

None.

Response Headers

- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing)

Response Status

200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
404 – D2 Creation Profile not found.
500 - Other unexpected server error

Response Body

This is a XML or JSON representation of the D2 Creation Profile.

Types

The types resource represents a collection of all the types that are defined in a specific D2 Creation profile.

URI Template

/repositories/ {repositoryName}/ type-configuration?profile={profileId}

repositoryName	Name of the repository
profileId	Object ID of the Creation Profile (mandatory)

Feed

Feed Id	Feed Title	Updated	Entry	Supports POST or not
URI of the types resource without the file extension	List of types	Server's current time	Type	NO

Entry Id	Entry Title
URI of the type	Type Name

Link Relation

Link Relation	Description
self	URI of the D2 Types collection feed for this profile.

Operations

Method	Description
GET	Lists all the D2 Types in the repository for this creation profile.

Get Types-D2

Lists all the types configured for the specific creation profile.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

- Accept
- Authorization
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

None.

Response Headers

- Content-Length
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types

- application/atom+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing

Response Status

200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
403 - Permission denied
500 - Other unexpected server error

Response Body

This is a XML or JSON representation of the D2 Types collection.

- The body contains a list of types configured for this creation profile.
- Each object may contain all configured properties of this type, depending on the setting of the query parameter (inline = true).
- The returned child objects collection only contains those that the user has access to.
- Each type entry must contain link that point to a specific type.

opentext™

Type

The type resource represents the configuration of a type in a specific creation profile.

URI Template

/repositories/{repositoryName }/type-configuration/{typeId}?profile={profileId}

repositoryName	Name of the repository
profileId	Object ID of the Creation Profile to which this type belongs (mandatory)
typeId	Object ID of the type

Link Relation

Link Relation	Description
self	Link to this type
type-configuration	Collection of types that are defined as part of this specific profile

Operations

Method	Description
GET	Retrieves properties, and other information of the type resource.

Get a Type

Gets properties and other information of this Type. Properties are returned as embedded elements in the response message body.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

• Accept
• Authorization
• Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

None.

Response Headers

• Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types

• application/vnd.emc.documentum+xml
• application/vnd.emc.documentum+json
• application/xml (For compatible viewing)
• application/json (For compatible viewing)

Response Status

200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
500 - Other unexpected server error

Response Body

This is a XML or JSON representation of the type configured in a specific D2 Creation Profile.

Templates

The templates resource represents the list of available templates for this object.

URI Template

/repositories/{ repositoryName }/objects/{objectId}/ document-templates

repositoryName	Name of the repository
objectId	Object ID

Link Relation

Link Relation	Description
self	Link to the templates for this object.

Operations

Method	Description
GET	Retrieves the list of content templates applicable for this object.

Get Templates

Gets the list of available templates for this object. The template properties are returned as embedded elements in the response message body. Other information, such as the link to the actual template, is referenced from the link relations of the response message body.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

- Accept
- Authorization
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

None.

Response Headers

- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json
- application/xml (For compatible viewing)
- application/json (For compatible viewing)

Response Status

200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
500 - Other unexpected server error

Response Body

This is a XML or JSON representation of the list of templates available for this object.

Create Object Content using Templates

Creates an object's content based on an existing template. This operation is performed on an already created object. When the object has no content attached to it, the template is bound to the object. When the object already has content and is of a compatible format to the template, then the template is attached. If the format is incompatible, an appropriate error message is provided.

URI Template

/repositories/{ repositoryName }/objects-d2/{objectId}

repositoryName	Name of the repository
objectId	Object ID

This URI template will be replaced with /repositories/{repositoryName}/objects/{objectId}. This will be done when the URIs from Core REST can be overridden.

Operations

Method	Description
POST	Specify the template name to be attached to the object.

HTTP Method

POST

Server Accepted Request Media Types
• application/vnd.emc.documentum+xml
• application/vnd.emc.documentum+json

Query Parameters
None

Request Headers
• Accept
• Authorization
• Content-Type
For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body
XML or JSON representation of the properties of template to be set on the object.

Example:
{ “properties”: {“template_name”: “AOP Document”, “folder_id”:” 0c01e2408004ee4e”}}

Response Headers
• Content-Type
For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types
• application/vnd.emc.documentum+xml
• application/vnd.emc.documentum+json
• application/xml (For compatible viewing)
• application/json (For compatible viewing)

Response Status
200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
500 - Other unexpected server error

Response Body
This is an XML or JSON representation of the object to which template was bound.

opentext™

Preview URLS

The templates resource represents the list of available preview URLs for this object. BOCS/ACS and O2/C2 plugins are supported.

URI Template

/repositories/{repositoryName}/objects/{objectId}/preview-urls-d2

repositoryName	Name of the repository
objectId	Object ID

Link Relation

Link Relation	Description
self	Link to the parent object.
parent	Link to the parent object

Operations

Method	Description
GET	Retrieves the list of preview URLs applicable for this object.

Get Preview Urls

Gets the list of available preview URLs for this object. URLs are returned as embedded elements in the response message body.

HTTP Method

GET

Server Accepted Request Media Types

None

Query Parameters

Inline – Refer to Common Definition – Query Parameters for more info.

Request Headers

• Accept

• Authorization
• Content-Type
For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body
None.

Response Headers
• Content-Type
For more details about HTTP headers, see Common Definition - HTTP Headers.

Supported Response Media Types
• application/vnd.emc.documentum+xml
• application/vnd.emc.documentum+json
• application/xml (For compatible viewing)
• application/json (For compatible viewing)

Response Status
200 - Information retrieved successfully
400 - Bad request; invalid property name or value
401 - Authentication failed
500 - Other unexpected server error

Response Body
This is an XML or JSON representation of the list of URLs available for this object.

Configuration

Edit “settings.properties” file in “dctm-rest/WEB-INF/classes” path.
Define the D2FS server url.

```
connection.remote.url=http[s]://[proxy|server][:[port]]/[D2_Ctx]
```

C2 and O2 plugins installation

O2/C2 plugins are supported. They need to be manually installed.

C2 : Copy “C2-API.jar” and “C2-Plugin.jar” files in “dctm-rest/WEB-INF/lib” path.

O2 : Copy “O2-API.jar” and “O2-Plugin.jar” files in “dctm-rest/WEB-INF/lib” path.

Examples

Follow these steps after deploying Documentum D2FS REST Services:

1. Type the following URL in your web browser to navigate to the service.
Define <docbase> and <id> by a valid value.

http://localhost:8080/dctm-rest/repositories/<docbase>/objects/<id>/preview-urls

JSON result :


```
{
   "entries": [
      {
         "content": {
            "links": [
               {
                  "href":"http://<server>/d2fs-
rest/repositories/<docbase>/objects/<id>/preview-urls-d2/",
                  "rel": "self"
               },
               {
                  "href":"http://<server>/d2fs-rest/repositories/<docbase>/objects/",
                  "rel": "parent"
               }
            ],
            "url":
"http://<server>/D2/servlet/Download?uid=context_rest_XXXXXXXX&_docbase=<docbase>&_username=XXXX&_password=XXXXXX&_locale=en&id=<id>&format=pdf&event_name=d2_view_inline&c2_config_name=C2+QA+View"
         },
         "id": "http://<server>/d2fs-rest/repositories/<docbase>/objects/<id>/preview-urls-
d2/",
         "links": [
            {
               "href":"http://<server>/d2fs-
rest/repositories/<docbase>/objects/<id>/preview-urls-d2/",
               "rel": "self"
            }
         ],
```

```
        "title":
"http://<server>/D2/servlet/Download?uid=context_rest_XXXXXXXX&_docbase=<docbase>&_userna
me=XXXX&_password=XXXXXX&_locale=en&id=<id>&format=pdf&event_name=d2_view_inline&c2_confi
g_name=C2+QA+View",
        "updated": "2014-08-25T15:36:22.882+02:00"
      }
   ],
   "id": "http://<server>/d2fs-rest/repositories/<docbase>/objects/<id>/preview-urls-d2",
   "links": [
      {
        "href":"http://<server>/d2fs-rest/repositories/<docbase>/objects/<id>/preview-
urls-d2/",
        "rel": "self"
      }
   ],
   "title": "Preview urls on object: <id>",
   "updated": "2014-08-25T15:36:22.882+02:00"
}
```


If C2 is installed, check if “c2_config_name” parameter is present in the URL.
If O2 is installed, check if “o2_config_name” parameter is present in the URL.

Object Creation

Unlike Core REST APIs, where the creation of objects happens within a folder, in the D2 REST Server Space, the object creation can happen at the “repository” level. Created objects are automatically placed into the user’s home directory or auto-linked to a specific folder, depending on the D2 configuration. Hence, the object creation URI template resides at the repository level.

As in Core REST, objects can be created in one or two steps. In a single-step creation process, the objects are created by passing both content and properties together. In a two-step creation, an object without content is created and then content is attached to the created object.

While attaching content, templates can be bound as detailed in the Templates section. The other way to attach content is by using the link relation “contents” on the object. For more details on how to do this, see https://community.opentext.com/docs/DOC-33250 .

URI Template

/repositories/ { repositoryName }/object-creation

repositoryName	Name of the repository

Link Relation

All the link relations that is applicable for a newly created object. See the “Link Relation” section, in the SysObject resource of the Core REST Developer Guide, which provides information on what kind of links are available for a certain type of object.

Operations

Method	Description
POST	Creates a new object

HTTP Method

POST

Server Accepted Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

Query Parameters

None.

Request Headers

- Accept
- Authorization
- Content-Type

For more details about HTTP headers, see Common Definition - HTTP Headers.

Request Body

XML or JSON representation of the object to create.

- The object type is specified in the r_object_type property of the request body.
- Any other properties of the object can also be specified in the request body. The value of input date attributes should be in ISO format with time zone suffixed. The format should be yyyy-MM-dd'T'HH:mm:ss.SSSXXX (e.g.: 2023-02-09T10:15:00.000+0530). Other date time formats are not supported.
- The D2 Configurations that are applicable for this object creation is passed in the request body along with other properties such as object_name.
- D2 Configuration takes precedence over supplied properties, such as object_name or folder_id, if there are auto-naming rules and auto-linking rules specified.

• During creation, it is not possible to pass the template information. Template information can only be set after the object is created because the lists of available templates that can be bound to the object depend on the created object. So, the template is not a valid d2 configuration during creation.

• In case of mulitpart, it is necessary to set the Content-type to **multipart/form-data; boundary=boundary_string.** The boundary string acts as a separation between the form-data and the content. For more information, see the "Multi-part Request Representation" section of the Core REST Developer Guide.

Examples

• Creation of object without content:

{"properties":{"r_object_type": "bookpreview", "d2_configuration": ["start_version=2.2", "lifecycle=New lifecycle", "folder_id=0c01e24080000105"], "object_name":"test.txt"}}

• Creation of object with content:

--91FJE9lOv7JqplS6Y_AYIoCtY2ePfQxPaiK
Content-Disposition: form-data; name="object"
Content-Type: application/vnd.emc.documentum+json; charset=UTF-8

{"properties":{"r_object_type": "dm_document", "d2_configuration": {"inheritance_config ":"New inheritance", "inherited_id":"0900304280007df9","inherit_properties":"true", "lifecycle": "Document Life Cycle"}, "object_name":"test33.txt"}}
--91FJE9lOv7JqplS6Y_AYIoCtY2ePfQxPaiK
Content-Disposition: form-data; name="content"; filename="plain.txt"
Content-Type: plain/text; charset=UTF-8

This is a primary content sample with plain text!
--91FJE9lOv7JqplS6Y_AYIoCtY2ePfQxPaiK

Object Versioning

Objects can be versioned by the user who has sufficient permissions on the object. In addition to the permissions, the D2-Configuration offers configuration to place additional options and restrictions when checking in and checking out the object.

As in Core REST, objects can be versioned by doing a checkout followed by a check-in. The check-in can be a minor/major/branch/same version. The checkout places a lock on the object which can be removed by doing a cancel-checkout. The check-in request can be a POST request which posts:

a) text/plain: for content only.

b) multi-part: for content and properties.

c) JSON/XML: for properties only.

While doing a checkout/check-in the corresponding D2 configurations, check-in and checkout config objects are read. The configurations are honored by the D2REST services.

URI Template

/repositories/ { repositoryName }/ objects/{chronicleId}/versions?object-id={objectId}&version-policy=next-major

/repositories/ { repositoryName }/ objects/{chronicleId}/versions?object-id={objectId}&version-policy=next-minor

/repositories/ { repositoryName }/ objects/{chronicleId}/versions?object-id={objectId}&version-policy=branch-version

/repositories/ { repositoryName }/ objects/{chronicleId}/versions?object-id={objectId}&version-policy=same-version

repositoryName	Name of the repository
chronicleId	Chronicle ID of the object.
objectId	Object ID

Link Relation

All the link relations that is applicable for a newly created object. See “Link Relation”, section in SysObject resource of the Core REST Developer Guide, which provides information on what kind of links are available for a certain type of object.

Other Details

Please refer to the Core REST Developer Guide for more info since these link relations are extensions from Core REST and they honor D2-Config. Sample steps of how object can be locked and versioned are also present in the Guide.

Sample Steps

These steps describe how to perform common tasks in D2.

Viewing profiles, types

Follow these steps after deploying Documentum D2FS REST Services:

1. Type the following URL in your web browser to navigate to the service node (Home Document).

http://localhost:8080/dctm-rest/services.json

The service node contains a list of the available services.


```
{
    "resources": {
        "http://identifiers.emc.com/linkrel/repositories": {
            "href": "http://localhost:8080/d2fs/repositories.json",
            "hints": {
                "allow": [
                    "GET"
                ],
                "representations": [
                    "application/xml",
                    "application/json",
                    "application/atom+xml",
                    "application/vnd.emc.documentum+json"
                ]
            }
        },
        "about": {
            "href": "http://localhost:8080/d2fs/product-info.json",
            "hints": {
                "allow": [
                    "GET"
                ],
                "representations": [
                    "application/xml",
                    "application/json",
                    "application/vnd.emc.documentum+xml",
                    "application/vnd.emc.documentum+json"
                ]
            }
        }
    }
}
```


In the link relation http://identifiers.opentext.com/linkrel/repositories, note the URI to the repositories resource.

2. Click the repositories link you got from step 1 to navigate to the list of all available repositories. Explore the output and note the information in the entries element.

3. Click the href link of the edit link relation of a repository in the entries element to retrieve the details of the repository. Enter your credentials if you are prompted for authentication.

By accessing the “href” values for the various link relations in the links element, you can drill down various resources in the repository.

4. By clicking the “href” for link relation (**http://identifiers.opentext.com/linkrel/profiles-d2**) in the links section, you can navigate to the list of available D2 Creation Profiles. This list is based on the user’s context which is configured in D2.

```
{
    "id": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2",
    "title": "List of D2 Creation Profiles",
    "updated": "2014-07-04T16:18:55.268+05:30",
    "links": [
        {
            "rel": "self",
            "href": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2"
        }
    ],
    "entries": [
        {
            "id": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/0000304280001521",
            "title": "Common_Matrix",
            "content": {
                "src": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/0000304280001521"
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/0000304280001521"
                }
            ]
        },
        {
            "id": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/0000304280001526",
            "title": "Dm_Document",
            "content": {
                "src": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/0000304280001526"
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/0000304280001526"
                }
            ]
        },
        {
            "id": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/000030428000167b",
            "title": "SKV_Document",
            "content": {
                "src": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/000030428000167b"
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/d2fs/repositories/winsql2/profiles-d2/000030428000167b"
                }
            ]
        }
    ]
}
```


Optionally, “inline=true” query param can be used for the complete entry to be visible.

```
{
    "id": "http://localhost:8080/d2fs/repositories/xCP21C7repo/profiles-d2",
    "title": "List of D2 Creation Profiles",
    "updated": "2014-07-04T16:21:59.552+05:30",
    "links": [
        {
            "rel": "self",
            "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/profiles-d2"
        }
    ],
    "entries": [
        {
            "id": "http://localhost:8080/d2fs/repositories/xCP21C7repo/profiles-d2/0001e24080006b20",
            "title": "aaaaa",
            "content": {
                "object_name": "aaaaa",
                "r_object_id": "0001e24080006b20",
                "title": "",
                "parent_configuration_profile": "BookPreviewCreation",
                "user_group": "",
                "linked_document_profile": false,
                "creation_profile": true,
                "import_profile": true,
                "skip_edit_content": true,
                "inherit_properties": true,
                "inherit_content": true,
                "inherit_vd_structure": false,
                "dictionary_names": [
                    "BookLanguage"
                ],
                "attribute_names": [
                    ""
                ],
                "links": [
                    {
                        "rel": "self",
                        "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/profiles-d2/0001e24080006b20"
                    },
                    {
                        "rel": "http://identifiers.emc.com/linkrel/types-d2",
                        "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2?profile=0001e24080006b20"
                    },
                    {
                        "rel": "http://identifiers.emc.com/linkrel/profiles-d2",
                        "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/profiles-d2"
                    }
                ]
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/profiles-d2/0001e24080006b20"
                }
            ]
        },
```


5. Alternatively, to only view a specific profile, navigate on the link rel: “self” of an entry.

```
{
    "object_name": "aaaaa",
    "r_object_id": "0001e24080006b20",
    "title": "",
    "parent_configuration_profile": "BookPreviewCreation",
    "user_group": "",
    "linked_document_profile": false,
    "creation_profile": true,
    "import_profile": true,
    "skip_edit_content": true,
    "inherit_properties": true,
    "inherit_content": true,
    "inherit_vd_structure": false,
    "dictionary_names": [
        "BookLanguage"
    ],
    "attribute_names": [
        ""
    ],
    "links": [
        {
            "rel": "self",
            "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/profiles-d2/0001e24080006b20"
        },
        {
            "rel": "http://identifiers.emc.com/linkrel/types-d2",
            "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2?profile=0001e24080006b20"
        }
    ]
}
```


This response has link relations to point to the:

a) types present in this profile.

b) a self-link pointing to itself (this profile).

6. To know the list of types configured in a profile, navigate to the “href” of the link relation identified by: **http://identifiers.opentext.com/linkrel/types-d2**. Use inline=true to get the type configuration along with the types. Otherwise, navigate on the “self” link relation of each type entry.

```
{
    "id": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2",
    "title": "Types for Profile:  0001e24080006b20",
    "updated": "2014-07-04T16:27:12.445+05:30",
    "links": [
        {
            "rel": "self",
            "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2?profile=0001e24080006b20"
        }
    ],
    "entries": [
        {
            "id": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2/0001e24080006def?profile=0001e24080006b20",
            "title": "bookpreview",
            "content": {
                "r_object_id": "0001e24080006def",
                "type_name": "bookpreview",
                "first_value": "English",
                "dictionary_values": [
                    "English"
                ],
                "property_config": "PreviewDocProperty",
                "start_version": "0.1",
                "attribute_inheritance": "PreviewInheritance",
                "lifecycle": "New lifecycle",
                "workflow": "New workflow",
                "default_value": "previewValueTempalte",
                "o2_config": "O2Transfer",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2/0001e24080006def?profile=0001
                    },
                    {
                        "rel": "http://identifiers.emc.com/linkrel/types-d2",
                        "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2?profile=0001e24080006b20"
                    }
                ]
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/d2fs/repositories/xCP21C7repo/types-d2/0001e24080006def?profile=0001e240
                }
            ]
        }
    ]
}
```


As seen earlier for profile, each type has similar link relations to navigate to:

a) the types which are present in this profile (types-d2).

b) self link (which is this type).

Content creation

There are 2 ways to create content:

A) Create Object without Content; attach content to the object (two steps).

B) Create object with content (single step).

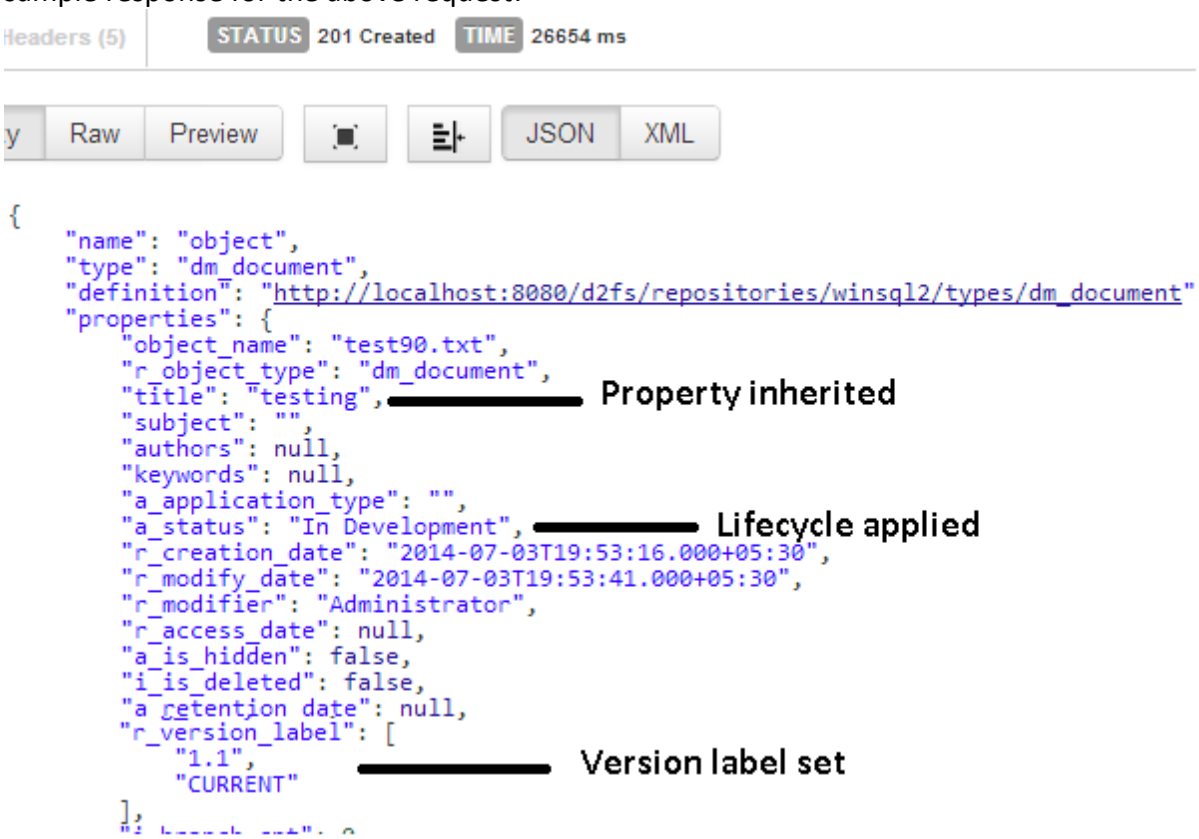
Create Object without content

A) To create Object without content, navigate to the href pointed by linkrelation: **http://identifiers.opentext.com/linkrel/objects -d2.**

Now, POST data such as below to create an object without content:

```
{"properties":{"r_object_type": "dm_document",
               "d2_configuration":
               {
                 "start_version":"1.1",
                 "inheritance_config":"New inheritance",
                 "inherited_id":"0900304280007df9",
                 "inherit_properties":"true",
                 "lifecycle": "Document Life Cycle"
               },
               "object_name":"test33.txt"
              }
}
```

The "d2_configuration" section provides the list of configurations that are to be applied to the created object. Note that these configurations need to exist in the Docbase. Here is a sample response for the above request:



Now, to set the content, follow the Core REST link relations or use templates from D2.

```
{
    "rel": "contents",
    "href": "http://localhost:8080/d2fs/repositories/winsql2/objects/090030428000801b/contents"
},
{
    "rel": "http://identifiers.emc.com/linkrel/primary-content",
    "href": "http://localhost:8080/d2fs/repositories/winsql2/objects/090030428000801b/contents/content"
},
{
    "rel": "http://identifiers.emc.com/linkrel/templates-d2",
    "href": "http://localhost:8080/d2fs/repositories/winsql2/objects/090030428000801b/templates-d2"
},
```


To get the list of available templates for this object, follow the href for the link relation: **"http://identifiers.opentext.com/linkrel/document-templates"**
The response contains information about the template and the URL to the actual template document object.


```
{
    "name": "templates-d2",
    "templates": [
        {
            "r_object_id": "080030428000693a",
            "object_name": "Bank Word Template.doc",
            "preview_mime": "msw8",
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/d2fs/repositories/winsql2/objects/080030428000693a"
                }
            ]
        },
        {
            "r_object_id": "080030428000698e",
            "object_name": "O2 Demo Template AR 2.doc",
            "preview_mime": "msw8",
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/d2fs/repositories/winsql2/objects/080030428000698e"
                }
            ]
        }
    ],
    "links": [
        {
            "rel": "self",
            "href": "http://localhost:8080/d2fs/repositories/winsql2/objects/0900304280008014/templates-d2"
        },
        {
            "rel": "parent",
            "href": "http://localhost:8080/d2fs/repositories/winsql2/objects/0900304280008014"
        }
    ]
}
```


To set content to the object without using templates, use the other link relations "content".

Create Object with content

To create Object with content, navigate to the href pointed by linkrelation: **http://identifiers.opentext.com/linkrel/object-creation.**

Now, POST data such as below to create an object with content.

This content can be stored as a file in the local machine and the content-type needs to be set as: **multipart/form-data; boundary=boundary_string**.

```
--91FJE9lOv7JqplS6Y_AYIoCtY2ePfQxPaiK
Content-Disposition: form-data; name="object"
Content-Type: application/vnd.emc.documentum+json; charset=UTF-8

{"properties":{"r_object_type": "dm_document", "d2_configuration": {"inheritance_config":"New inheritance",
"inherited_id":"0900304280007df9","inherit_properties":"true", "lifecycle": "Document Life Cycle"},
"object_name":"test33.txt"}}
--91FJE9lOv7JqplS6Y_AYIoCtY2ePfQxPaiK
Content-Disposition: form-data; name="content"; filename="plain.txt"
Content-Type: plain/text; charset=UTF-8

This is a primary content sample with plain text!
--91FJE9lOv7JqplS6Y_AYIoCtY2ePfQxPaiK¡ª
```

This is like the earlier request; however, the “d2_configuration” is passed along with the content.

The response obtained is like the earlier response.

C2-View & C2-Print

To support C2-View and C2-Print, install the C2-plugin to the D2 REST Web application. Manually copy C2-API.jar and C2-plugin.jar into the D2 Rest Web War file.

URI Templates:

1) X_C2_VIEW_D2_URI_TEMPLATE : {repositoryUri}/objects/{objectId}/views/c2-view
2) X_C2_PRINT_D2_URI_TEMPLATE: '{repositoryUri}/objects/{objectId}/views/c2-print'

C2-View :

C2-View lists all the download URLs for the given object only if the object has at least one PDF rendition. It gets download urls for all the C2_View_configs applicable for the given document.

Link Relation:

This new http://identifiers.opentext.com/linkrel/views/c2-view link relation is available to the object if the object's content itself is PDF or it has at least one PDF rendition available.

Eg :

```
<link rel="http://identifiers.opentext.com/linkrel/views/c2-view"
href="http://10.31.168.68:8090/d2_rest/repositories/CSAUTO/objects/090030398005c5b2/views/c2-view"/>
```

HTTP Method

GET

Server Accepted Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

Query Parameters

None.

HTTP Sample Request:

```
http://10.31.168.68:8090/d2_rest/repositories/CSAUTO/objects/090030398005c5b2/views/c2-view
```

Sample Response for the above request is shown below:

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:dm="http://identifiers.emc.com/vocab/documentum" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 ▼<id>
    http://10.31.168.68:8090/d2fs-rest-web-4.6.0/repositories/CSAUTO/objects/090030398005c5b2/views/c2-view
  </id>
  <title>List of C2 View URLs on object 090030398005c5b2</title>
 ▼<author>
    <name>openText Author  </name>
  </author>
  <updated>2015-12-28T15:29:34.781+00:00</updated>
  <dm:page>1</dm:page>
  <dm:items-per-page>1000</dm:items-per-page>
  <dm:total>1</dm:total>
  <link rel="self" href="http://10.31.168.68:8090/d2fs-rest-web-4.6.0/repositories/CSAUTO/objects/090030398005c5b2/views/c2-view"/>
 ▼<entry>
   ▼<id>
      http://10.31.168.68:8090/d2fs-rest-web-4.6.0/repositories/CSAUTO/objects/090030398005c5b2/views/c2-view
    </id>
    <link rel="edit" href="http://10.31.168.68:8090/d2fs-rest-web-4.6.0/repositories/CSAUTO/objects/090030398005c5b2/views/c2-view"/>
   ▼<content>
     ▼<dm:view-url view-type="c2_view">
       ▼<dm:url>
          http://10.31.71.225:8080/D2/servlet/Download?
          uid=context_rest_2015122820592836916133718_docbase=CSAUTO&_username=Administrator&_password=DM_TICKET%3DT0JKIE5VTEwgMAoxMwp2ZXJzaW9uIElOVCBTIDAI
        </dm:url>
       ▼<dm:links>
          <link rel="self" href="http://10.31.168.68:8090/d2fs-rest-web-4.6.0/repositories/CSAUTO/objects/090030398005c5b2/views/c2-view"/>
          <link rel="parent" href="http://10.31.168.68:8090/d2fs-rest-web-4.6.0/repositories/CSAUTO/folders/0c00303980000107"/>
        </dm:links>
      </dm:view-url>
    </content>
  </entry>
</feed>
```


<dm:url>: The download URL also lists BOCS/ACS URL based on the following properties configured in d2fs.properties.

D2-BOCS=true

includeAcsServer=true

Note: OpenText does not support Linked objects and Multi repository solutions as it's a limitation in CORE REST.

C2-Print :

C2-Print lists all the print URLS for the given object only if the object has at least one PDF rendition. It gets print-urls for all the c2_print_configs applicable for the given document.

Link Relation:

This new http://identifiers.opentext.com/linkrel/views/c2-print link relation is available to the object if the object's content itself is PDF or it has at least one PDF rendition available.

Eg :

```
<link rel="http://identifiers.opentext.com/linkrel/views/c2-print"
href="http://10.31.168.68:8090/d2_rest/repositories/CSAUTO/objects/090030398005c5b2/views/c2-print"/>
```

HTTP Method

GET

Server Accepted Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

Query Parameters

controlled_field_1
controlled_field_2
controlled_field_3.

HTTP Sample Request:

http://10.31.168.68:8090/d2_rest/repositories/CSAUTO/objects/090030398005c5b2/views/c2-print

Sample Response for the above request is shown below:


```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:dm="http://identifiers.emc.com/vocab/documentum" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
▼<id>
   http://localhost:8080/d2fs-rest-web/repositories/CSAUTO/objects/090030398005c5b2/views/c2-print
  </id>
  <title>List of C2 Print URLs on object 090030398005c5b2</title>
▼<author>
    <name> openText Author  </name>
  </author>
  <updated>2015-12-29T08:51:43.930+00:00</updated>
  <dm:page>1</dm:page>
  <dm:items-per-page>1000</dm:items-per-page>
  <dm:total>1</dm:total>
  <link rel="self" href="http://localhost:8080/d2fs-rest-web/repositories/CSAUTO/objects/090030398005c5b2/views/c2-print"/>
▼<entry>
  ▼<id>
     http://localhost:8080/d2fs-rest-web/repositories/CSAUTO/objects/090030398005c5b2/views/c2-print
    </id>
    <link rel="edit" href="http://localhost:8080/d2fs-rest-web/repositories/CSAUTO/objects/090030398005c5b2/views/c2-print"/>
  ▼<content>
    ▼<dm:view-url view-type="c2_print_view">
      ▼<dm:url>
         http://10.31.71.225:8080/D2/servlet/Download?
         uid=context_rest_201512291420032046785335&_docbase=CSAUTO&_username=Administrator&_password=DM_TICKET%3DT0JKIE5VTEwgMAoxMwp2ZXJzaW9uIElOVCBTIDA
       </dm:url>
      ▼<dm:links>
         <link rel="self" href="http://localhost:8080/d2fs-rest-web/repositories/CSAUTO/objects/090030398005c5b2/views/c2-print"/>
         <link rel="parent" href="http://localhost:8080/d2fs-rest-web/repositories/CSAUTO/folders/0c00303980000107"/>
       </dm:links>
     </dm:view-url>
   </content>
  </entry>
</feed>
```


Workflow and Task related REST Services

List of REST End points related to workflow

Task List End Point:

http://{host}:{port}/d2fs-rest/repositories/{repositoryName}/tasklist?inline=true

Http Method: GET

Description: Task List End point lists all the tasks of a logged in user, if inline=true, the task response will include elaborated task description.

Note : Task response is not capturing Workflow participants information or Workflow Delegation user list.

Task Status End Point:

http://{host}:{port}/d2fs-rest/repositories/{repositoryName}/
/processes/{processName}/{processId}/{taskName}/{taskId}/status

Http Method: GET

Parameters:
processName: Name of the dm_workflow process
processId: r_object_id of the dm_workflow process
taskName: Name of the workflow activity
taskId: r_object_id of the dmi_queue_item

Description: Task Status End point provides the state of the workflow task.

Http Method: Post

Parameters:
processName: Name of the dm_workflow process
processId: r_object_id of the dm_workflow process
taskName: Name of the workflow activity
taskId: r_object_id of the dmi_queue_item

Description: Post method of Task Status End point allows us to perform various operations on Task.

1) JSON request body for Accepting a task

```
{
 "properties": {
                  "action" : "acquire"
                }
}
```

2) JSON request body for rejecting a task

```
{
"properties": {
"action" : "reject",
"comment": "test comment",
"next_task_id": "task id"
    }
 }
```

3) JSON request body for forwarding a task with sign-off intentions

```
        {
 "properties": {
  “action" : "forward",
 "comment": "test comment",
 "next_task_id": "task id"
  "signoff_login": "user name",
 "signoff_password": "user password"
   }
}
```

4) JSON request body for delegating a task

```
 {
   "properties": {
   "action" : "delegate",
   "user":"user name"
   }
```

TaskNotes EndPoint

http://{host}:{port}/d2fs-rest/repositories/{repositoryName}/processes/{processName}/{processId}/{taskName}/{taskId}/notes

HTTP Method

PUT

Server Accepted Request Media Types

- application/vnd.emc.documentum+xml
- application/vnd.emc.documentum+json

Sample request

```
{
        "properties" : {"task_note" :"POST NOTE THROUGH REST SERVICE"}
}
```

Digital Signature

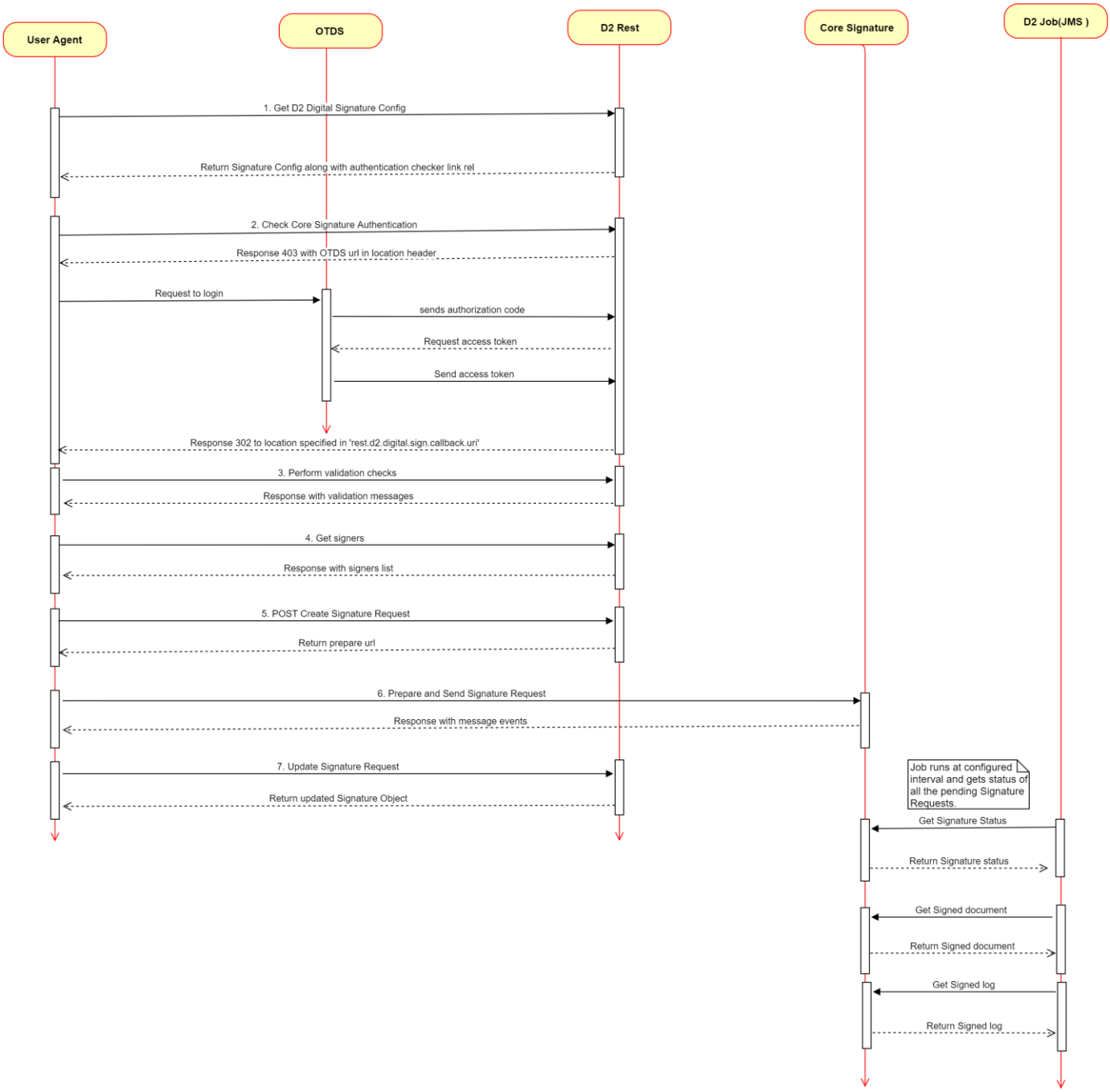
D2 SmartView provides the capability to send a document to digital signature providers for digital signatures. Digital signature solutions allow D2 to integrate with enterprise-grade signature applications that allow employees and third parties to sign quotes, contracts, and other documents in a fast, compliant and hassle-free way. This feature is achieved via D2 Smart View Workflows.

D2 Smart View supports OpenText Core Signature as a digital signature provider. OpenText™ Core Signature is a professional and enterprise-grade electronic signature application.

Refer to the D2 Administration Guide and the D2 Installation Guide for details on the different configurations required for this.

There are multiple Rest APIs to support this feature as listed below. For detailed information on Rest API spec, RADL needs to be referred.

The following diagram illustrates different entities involved and flow of Rest APIs calls.



1. **Digital Signature Configuration:** The rest end-point 'd2-task-digital-signature-config' returns the Digital Signature configurations of a D2 workflow task in D2-Config.
   The link rel (http://identifiers.emc.com/linkrel/d2-task-digital-signature-config) for this API can be accessed in the response of task details rest end-point when the task has been configured to use 'Digital Signature'. The configuration contains information such as Digital Signature provider name, whether signer grouping is allowed, whether external signer is allowed, and other configuration details.

2. **Authentication with Core Signature**: In order to access Core Signature APIs, one needs a valid OTDS token. D2 Rest uses authorization code flow to get valid access token.

   'd2-digital-signature-auth-checker' REST end-point has been provided to check if the user is authenticated with Core Signature. The link rel (http://identifiers.emc.com/linkrel/d2-digital-signature-auth-checker) to access this is provided in the response of 'd2-task-digital-signature-config' REST end-point.

   If the user is already authenticated with Core Signature, then HTTP status 200 would be returned. Otherwise, HTTP status 403 would be returned with the Location header with Core Signature authorization code URL. The Location header would also contain redirect URI.

   Once the user provides OTDS credentials then the authorization code would be sent to D2 Rest using the redirect_uri specified in the Location header URL. D2 Rest server exchanges the authorization code for an access token.
   This access token is stored in the D2CORESIGN-TOKEN cookie for future access to Core Signature APIs using D2 Rest.

   Clients need to specify a call back handler in rest-api-runtime.properties using configuration "rest.d2.digital.sign.callback.uri". Once the access token is received by Rest API, the client would be navigated to this call back handler.

3. **Validation API:** If a document is sent for signature, then the user shouldn't be allowed to send it again until the signature is complete/cancelled. Clients can call the 'd2-digital-signature-documents-validator' rest end point to check this. The link rel http://identifiers.emc.com/linkrel/ d2-digital-signature-documents-validator for this API is returned in the 'd2-task-digital-signature-config' rest end point. This also checks other validation conditions like if document is checked out, if the document size is more than the allowed size, or if the document type is allowed.

4. **Get signers list**: The rest end point 'd2-digital-signature-signees' returns the list of signers as configured in D2-Config. The link rel 'http://identifiers.emc.com/linkrel/d2-digital-signature-signees' is returned in ' d2-task-digital-signature-config' rest end point.

5. **Creating Digital Signature Request:** Create a Digital Signature Request by sending a POST request to the 'd2-digital-signatures' end point. The link rel (http://identifiers.emc.com/linkrel/d2-digital-

signatures) for this is returned in 'd2-task-digital-signature-config' REST end-point. This API would acquire a lock on the document on behalf of the superuser.

This API takes signer details, working documents, and supporting documents (attachments) in the request body. In response to this API, the signature provider prepare URL is returned.

6. **Send Digital Signature Request:** Clients need to open the prepare URL in another window to send the document for digital signature. Once the document is sent from the signature provider UI, there are two options to get call back. One is to append 'redirect_url' to the prepare URL and open it. The signature provider will redirect the user to this URL. The other options is to capture the windows message events from the opened window. Core signature notifies the caller with the message events once the document is successfully sent for signature.

7. **Updating Digital Signature Request:** The client needs to call 'd2-digital-signature' REST end-point once the signature request has been sent in the Signature provider UI in order to update the signature status in D2 Digital Signature object. The link rel (http://identifiers.emc.com/linkrel/d2-digital-signature) is returned from the task details end point. This API doesn't take any request body. Instead, it gets the latest status and signers' details from the provider and updates D2 Digital Signature object.

8. **Get Digital Signature Request**: Clients can call 'd2-digital-signature' end-point to know the latest status of a digital signature tied to a given task. The link rel (http://identifiers.emc.com/linkrel/d2-digital-signature) is returned from task details end point. This API is useful in scenarios when a signature request is sent but a task couldn't be marked as completed because of some network issue. So, when a user revisits the incomplete task, clients can check if the signature is already created and sent by calling this API. If the signature is sent, then the user might be allowed to complete the task. If the signature is created but not sent, then the client needs to send a new digital signature request to D2 Rest. An old signature request related to the task would be deleted from D2 as well as from the signature provider.

9. **Background D2 job**: As described in the D2 Administration Guide, for each signature provider, there is a job running on Documentum server. D2 method linked to the job polls the signature provider to get the latest status on signature request. Every time there is a change in signer status or overall status, there would be entry in the audit table depending on D2 audit config.
D2 installer deploys D2JobCoreSignStatusUpdate and D2CoreSignStatusUpdateMethod for Core Signature Provider. Once a signature has been completed, D2 method would do the following:

   a) It would get the signed document from signature provider and check-in the document to the DCTM repository. It would honor any check-in config specified in D2 Config under Digital Signature config. Once the document is checked in, the lock on the document is released.

   b) It would get the signing log from the provider. The log file is added as relation 'D2_DIGITAL_SIGNATURE_LOG' to each of the signed documents. One needs to specify location and ACL for the signing log in Digital Signature config.

c) The workflow “Digital Signature Tracker Task” would be marked as completed, and the workflow would move forward.

d) If ‘Digital Signature Tracker Task’ has been designed with a ‘Reject’ flow and the signature request is ‘declined/cancelled’, then the task would be completed with a Reject path.

10. Workflow reporting:

a) ‘d2-workflow-tasks’ REST end-point returns information on ‘Digital Signature Tracker Task’. The task would have configuration ‘is_digital_signature_awaiting_task’ set to true.

b) ‘d2-workflow-tasks-audits’ REST end-point returns information on all the digital signature events auch as document sent for signing or signer signed/declined. This end-point takes a request param ‘is_digital_signature_awaiting_task’.

Zip and Download

Using D2-rest end-points user can create a compressed (zip) file from a set of files and folders. To do this following rest end-points required to be called in the given order

1. Validate
2. Initialize zip and download
3. Status check
4. Download content
5. Cancel. This API can be called after the Initialize zip and download or before the Download content API to have its meaningful impact.

Addition to the above APIs there is a job which runs on method sever to clean up the compressed files. Details of the above rest end-points and job are documented in the following sections.

These rest end-points depend on ‘Smartview Content Import/Export’ configurations. So please refer admin guide for more info on ‘Smartview Content Import/Export’ configurations.

1. Validate

The rest end-point ‘ d2-zip-and-download-validation‘ is used to validate the allowed file size, file type and total count. link rel (http://identifiers.emc.com/linkrel/d2-zip-and-download-validation) for this rest end-point can be accessed in the response of the repository rest end-point. This rest end-point needs a preconfigured zip and download configuration to mention about the allowed total file size, file type, total files count etc. For more info on this please refer “RADL” documentation.

2. Initialize zip and download

The rest end-point 'd2-zip-and-downloads' returns a unique ID representing dm_sysobject which is used to track the status of files archiving process(zip and download process status). The link rel (http://identifiers.emc.com/linkrel/d2-zip-and-downloads) can be accessed in the response of the validation rest end-point , 'd2-zip-and-download-validation'. This rest end-point creates a tracking dm_sysobject and starts a separate background thread which gets files from Content Server, compresses and creates a downloadable archieved file (zip file). This background thread also creates a manifest file which contains details about the files included in the archieve and also the warning messages for the files which are not included as part of the archieved file. For more info on this please refer "RADL" documentation.

3. Status Check

This rest end-point 'd2-zip-and-download-status 'returns current status of the files archiving process. Based on the status user/client takes a call whether to call download content API or not. The link rel (http://identifiers.emc.com/linkrel/d2-zip-and-download-status) can be accessed in the response of the 'd2-zip-and-downloads' API. The possible status values are STARTED, INPROGRESS, COMPLETED, CANCELLED. For more info on this please refer "RADL" documentation.

4. Download content

This rest end-point 'd2-zip-and-download-content' downloads the archived file to the clients machine by the name download.zip. If the file-name is passed in the request then downloaded file will have the name passed. The link rel (http://identifiers.emc.com/linkrel/d2-zip-and-download-content) can be accessed in the response of the rest end-point 'd2-zip-and-download-status '. This link rel will be available only after the status has the value COMPLETED in the response of the rest end-point 'd2-zip-and-download-status '. For more info on this please refer "RADL" documentation.

5. Cancel Zip and download

This rest end-point 'd2-zip-and-download-cancel' stops the file archive thread/process which is responsible for downloading and creating a archived file. After this rest end-point is called user will not be allowed to download half-baked archived file. The link rel (http://identifiers.emc.com/linkrel/d2-zip-and-download-cancel) can be accessed in the response of the rest end-point d2-zip-and-download-status. For more info on this please refer "RADL" documentation.

6. Zip and download clean up job

This job is used to clean up the archived files stored in the shared location. This job can be run periodically to clean up. Following are the details of the job.

Job Name: D2JobZipAndDownloadCleanup

Method Name: D2ZipAndDownloadCleanupMethod

Method Command: com.emc.d2.api.methods.D2Method -class_name com.emc.d2.api.methods.D2ZipAndDownloadCleanupMethod
Custom Argument: -age
Description: Zip and Download init process will save the base directory location in the tracker object in field **directory_location.**
This job will collect the directories information based on the following conditions,
Default age is one day or 1440 minutes
If status is INPROGRESS or STARTED, then the method will check the **DATEDIFF** with **zip_process_init_time**
If status is COMPLETED, then the method will check the **DATEDIFF** with **archive_end_date**
If the status is CANCELLED, then method will always collect the rows from DB
Once the Object Ids and Directories are collected then Job will delete the all the corresponding rows from the DB and delete the directories.
In this process this JOB will assume that all the DATEs are saved in the DB in UTC format a common format for APP server and Method Server, so that the date comparison can be uniform.

Advanced Search

Overview:

The advanced search feature from D2-REST provides search related capabilities to the REST clients such as creating and saving search criteria, fetching objects based on search criteria, fetching saved search objects, creating folders to save search criteria. Using advanced search, a user can search for documents based on a lot of different criteria such as the content of the document, the document properties or the type of document. The column and facet configuration defined by the user in advanced search displays the search result as per the user requirement. It is also possible to limit the search to user defined folders through this advanced search APIs.

This feature has been broken down into several different APIs based on their usage and for ease of access to this feature. Please refer to the glossary below to better understand the advanced search terminology in REST APIs before proceeding further.

Search terminology:

Search group: A user may choose to create a saved search object, the folder where this object is stored is referred to as a Search group. A search group is of 2 types: PRIVATE and PUBLIC. If a search object is meant to be shared with others, it should be stored under a PUBLIC search group else store it in a PRIVATE search group. Search group is also referred to as Search category.

Search sub-group: For better classification of search objects and for ease of management a user may choose to create folders in Search group. The saved search objects can then directly be stored under these folders. These folders are referred to as search sub-group or search sub-criteria.

opentext™

Advanced search APIs:

Advanced search configuration:
The Advanced search config REST API (link-rel.: http://identifiers.emc.com/linkrel/search-configuration) is used to fetch all the applicable advanced search configurations applicable to the current logged in user's context. These configurations include all the information a REST client might need to get started with advanced search. Information like facet enabled flag, full-text search enabled, case sensitive option enabled etc. are part of this API response. Refer RADL for a full sample response. This API also indicates whether the logged in user is allowed to create public saved search objects.

Advanced search attribute value list API:
This REST API (link-rel.: http://identifiers.emc.com/linkrel/d2-search-attribute-value-list) is used to fetch value of attributes that are configured with DQL/Dictionary. This REST API will return all the values configured for an attribute for the list of types sent as request param. These types must be configured in D2-Config search config.

Advanced search groups/sub-groups API:
This REST API (link-rel.: http://identifiers.emc.com/linkrel/d2-saved-search-groups) is used for 2 purposes:

1. To GET the list of all the search groups and sub-groups (PRIVATE or PUBLIC)
2. To create a new PRIVATE or PUBLIC search sub-group. When creating a new PUBLIC search group, the end users can choose to define READER groups.
   If a PUBLIC search sub-group is created without READER groups, all users will have READ access to saved search objects stored within this PUBLIC search sub-group.
   If a PUBLIC search sub-group is created with READER groups, only users belonging to the READER group will have READ access to saved search objects stored within this PUBLIC search sub-group. To get the possible list of users-groups that can be set as READER groups use the REST API (link-rel.: http://identifiers.emc.com/linkrel/groups).

Advanced search group/sub-group API:
This REST API (link-rel.: http://identifiers.emc.com/linkrel/d2-saved-search-group) is used for 3 purposes:

1. To get the details of a particular search sub-group
2. To rename a search sub-group
3. To delete a search group
   When deleting a search sub-group, it is possible to delete the search sub-group and all the saved search objects inside it by using the "force_delete" flag.

D2 Saved Searches API:
This REST API (link-rel.: http://identifiers.emc.com/linkrel/d2-saved-searches) is used for 2 purposes:

1. To get the list of all saved search objects and search sub-groups. This REST API can fetch all saved search object (PRIVATE, PUBLIC OR PRIVATE+PUBLIC) and search sub-groups in a single go, however, the request params exposed in this API help narrow down the response.
2. To create an advanced search object.

D2 Saved Search API:
This REST API (ink-rel.: http://identifiers.emc.com/linkrel/d2-saved-search) is used for 2 purposes:

1. To get the detailed information on advanced search object.
2. To edit advanced search object.

*Refer RADL for more information on the above APIs.

View Permission

Permissions API
This CORE-REST API (link-rel.: http://identifiers.emc.com/linkrel/permissions) is used to get the basic and extended permissions for a logged in user, any particular user or any group for the selected object

D2 Users Groups API
This D2-REST API (link-rel.: http://identifiers.emc.com/linkrel/d2-users-and-groups) is used for the following purposes:

1. To get the list of users and groups in the system.
2. Use the filter request parameter, to get a list of only users, only groups, or to find a user or a group by a property (such as name or email).

D2 Permissions set API

This D2-REST API (link-rel: http://identifiers.emc.com/linkrel/d2-permission-set) is used for getting the permission set of the object (including restriction list, required groups, and required group sets if MACL is enabled) with the following additional attributes and capabilities:

1. Gets an additional Boolean attribute *r_is_group* for each permission object.
2. Additional query parameter accessor-id (r_object_id of dm_user or dm_group) is introduced in the request to filter permission of a user or a group. If the accessor-id is passed then all the ancestor groups which are part of the permission set are returned in the result set.

Note: For 23.2 and newer, D2-REST (with the help of D2-Config) supports MACL (Mandatory Access Con Control List). This means the security template in D2-Config now supports Permission, Restriction, Required Group, and Required Group Set. Prior to 23.2, D2-Config only supported Permissions. All the D2-REST endpoints honor the MACL, wherever applicable. Refer to D2-Config guide for more details.

opentext™

Installation Guide

Follow below guidelines for deploying REST .war depending on the app server.

WebLogic 12.1.3

Disable web service annotation scan for WebLogic

Append below line to Java options of WebLogic startWeblogic.bat file.

-Dweblogic.servlet.DIDisabled=true

Since WebLogic has service provider hook for jaxb implementation, duplicate implementations for jaxb related jars results in a linkage error. Follow the manual changes to the lib folder:

Remove following jar files from lib folder of deployment .war.

jaxb-api-2.1.jar
jaxb-impl-2.1.6.jar
jsr173_api-20060801.jar
stax-api-1.0-2.jar
stax2-api-3.1.1.jar
xml-apis-1.3.04.jar
xmlParserAPIs-2.6.2.jar

Download below jars and add to lib folder:

http://central.maven.org/maven2/com/sun/xml/bind/jaxb-impl/2.2.11/jaxb-impl-2.2.11.jar

http://central.maven.org/maven2/xalan/xalan/2.7.0/xalan-2.7.0.jar

WebSphere Installation instructions

1) Manually remove the 3 following .jar files from d2fs-rest-web.war/lib folder:
   a) javax.servlet-api-3.0.1.jar
   b) xml-apis-1.3.04.jar
   c) xmlParserAPIs-2.6.2.jar

2) Change the class loader order at **Enterprise Applications > d2fs-rest-web_war > Manage Modules > d2fs-rest-web.war** class loader with local class loader first (parent last)

opentext™

About OpenText

OpenText enables the digital world, creating a better way for organizations to work with information, on-premises or in the cloud. For more information about OpenText (NASDAQ: OTEX, TSX: OTEX), visit opentext.com.

Connect with us:

OpenText CEO Mark Barrenechea's blog

Twitter | LinkedIn | Facebook