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™

opentext™

opentext™

opentext™

What's changed in 24.4?

Please refer to the below sections and their content for what's changed in 24.4.

1. Added a section "Export Audit and Locations".

2. Added a section "Grouping Audit Trails".
3. Added a section "Microsoft Information Protection (MIP)"
4. Added section "Collaborative Editing"
5. Relation section has been modified.
6. Zip and Download section has been modified.
7. Entitlement check (License enforcement)
8. Installation Guide section has been modified.
9. Added a section "Workflow filter configuration"
10. Added a section "Import Rendition service
11. Filter Configuration section has been modified.

What's changed in 24.2?

Please refer to the below sections and their content for what's changed in 24.2.

1. Added a section "Business Administration".
2. Added a sub-section "All Workflows" under section "Workflows".
3. Added a sub-section "All Workflow Status Summary" under section "Workflows".
4. Added a section "Aviator".
5. Added a section "Filter Configuration".
6. Added a section "File Count".
7. Added a section "Object specific permissions API".
8. Updated a section "Collections".
9. Added a section "Share Collections".
10. Added a section "D2 Objects Locations API".
11. Added a section "Audit Trails".
12. Added a section "Retirement of RADL".

What's changed in 23.4?

Please refer to the below sections and their content for what's changed in 23.4.

1. Added a section "Workflow".
2. Added a section "JDK-17 Spring 6 Migration".
3. Added a section "Security Changes".
4. Added a section "Login Logout Audits".
5. Added a section "Encrypted DM Ticket".
6. Added a section "D2 Column Preferences".
7. Added a section "Relations".
8. Added a section "D2 Options API."
9. Added a section "D2 preferences."
10. Added a section "Collections".

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 99999 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-body, in decimal number	Request /Response	Non-negative number

	of OCTETs, sent to the recipient		

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.

opentext™

- 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

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/link rel/ 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/link rel/ 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/link rel/ 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/link rel/ 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/link rel/ 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/link rel/ 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/link rel/ 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/link rel/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/link rel/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/linkr el/ 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/linkr el/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/linkr el/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/linkr el/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.

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

opentext™

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

opentext™

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)

opentext™

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.

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.

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]
```

opentext™

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>&_userna
me=XXXX&_password=XXXXXX&_locale=en&id=<id>&format=pdf&event_name=d2_view_inline&c2_confi
g_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.

```json
{
    "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.

![opentext logo]

```
{
    "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.

```json
{
    "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":"09003042800007df9",
                    "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:

Headers (5)   STATUS 201 Created   TIME 26654 ms

y   Raw   Preview   ▣   ⫤   JSON   XML

```
{
    "name": "object",
    "type": "dm_document",
    "definition": "http://localhost:8080/d2fs/repositories/winsql2/types/dm_document"
    "properties": {
        "object_name": "test90.txt",
        "r_object_type": "dm_document",
        "title": "testing",━━━━━━━━━ Property inherited
        "subject": "",
        "authors": null,
        "keywords": null,
        "a_application_type": "",
        "a_status": "In Development",━━━━━━ Lifecycle applied
        "r_creation_date": "2014-07-03T19:53:16.000+05:30",
        "r_modify_date": "2014-07-03T19:53:41.000+05:30",
        "r_modifier": "Administrator",
        "r_access_date": null,
        "a_is_hidden": false,
        "i_is_deleted": false,
        "a_retention_date": null,
        "r_version_label": [
            "1.1",                  ━━━━━━━ Version label set
            "CURRENT"
        ],
```

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/0900303
98005c5b2/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_20151228205983691613371&_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/0900303
98005c5b2/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_20151229142003204678533S&_docbase=CSAUTO&_username=Administrator&_password=DM_TICKET%3DT0JKIE5VTEwgMAoxMwp2ZXJzaW9uEIlOVCBTIDA
        </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.

opentext™

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"
        }
     }
```

opentext™

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.

opentext™

The following diagram illustrates different entities involved and flow of Rest APIs calls.
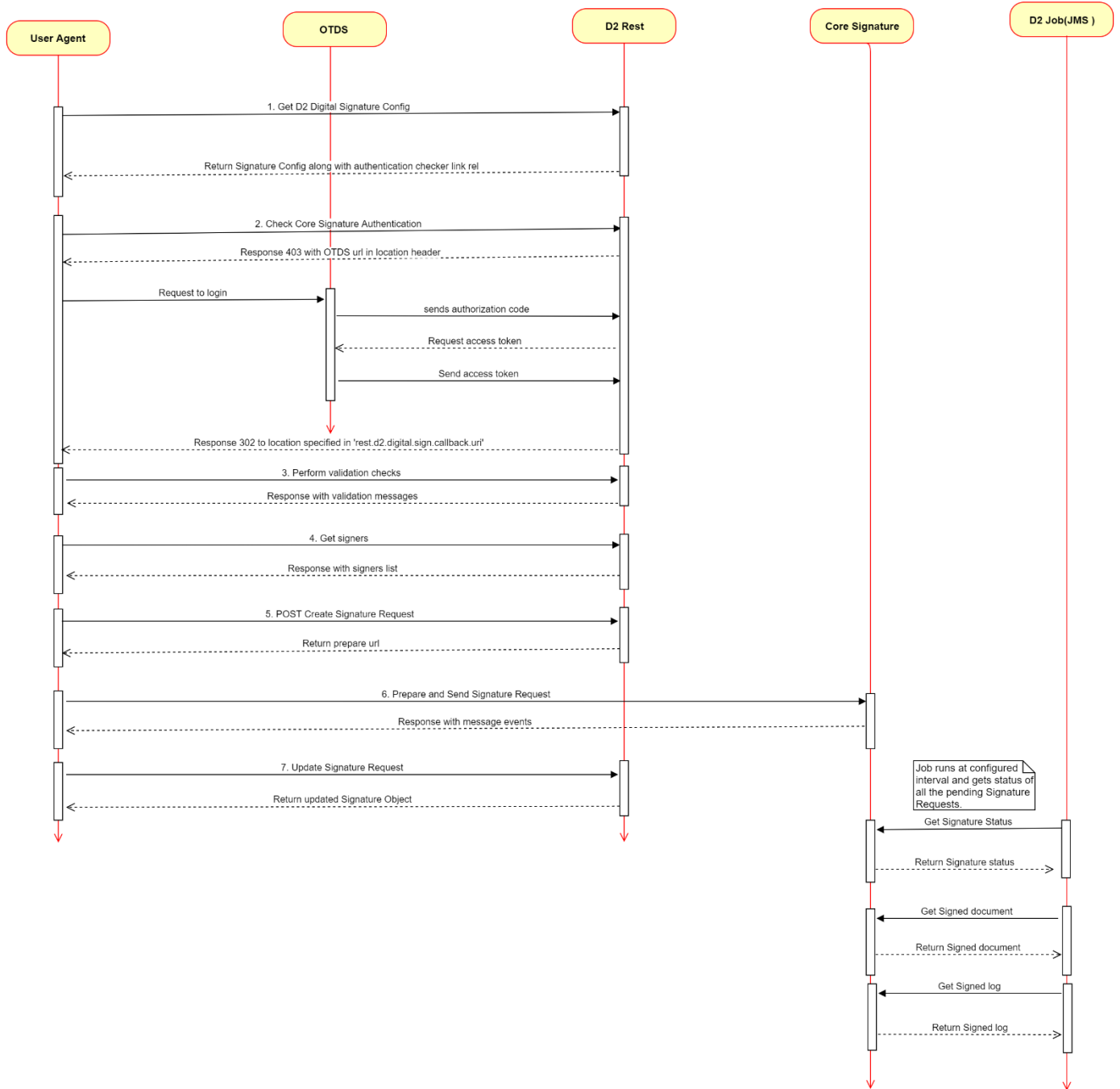
1. **Digital Signature Configuration:** The rest end-point 'd2-task-digital-signature-config' returns the Digital Signature configurations of a D2 workflow task in D2-Config.
The link rel (http://identifiers.emc.com/linkrel/d2-task-digital-signature-config) for this API can be accessed in the response 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](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.

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.

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.

Starting from 24.4 **view** query parameter introduced. This will be honored while generating Manifest file. Manifest file default Columns: **r_object_id, object_name, file_path, status, details** along with this we are going to having column names requested as part of request using query param: **view** or **widget_name** and **widget_type** is used to retrieve the column details.

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.

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.

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.

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:

opentext™

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 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.

Workflow

From 23.4 onwards, the workflow task configuration 'Display Task in Workflow Manager' checkbox is honored in d2-workflow-tasks D2-REST API (link-rel: http://identifiers.emc.com/linkrel/d2-workflow-tasks). Up until D2-REST version 23.2, this config option was not honored for this API.

All Workflows

Any user who has access to the AdminWorkflowWidget Widget (introduced in 24.2) can call the D2-Workflows REST endpoint (link rel.: http://identifiers.emc.com/linkrel/d2-workflows). This D2-REST endpoint is used to get the list of all the Workflows in the system (it is limited by the template selection in the AdminWorkflowWidget). More details on this API can be found in Swagger Documentation.

All Workflow Status Summary

The new D2-REST endpoint (link rel.: http://identifiers.emc.com/linkrel/all-workflows-status-summary) , introduced in 24.2 provides a workflow summary count using the AdminWorkflowWidget Widget name. The count displayed is limited by the template selection in the AdminWorkflowWidget Widget. More details on this API can be found in Swagger Documentation.

Workflow Filter Configuration

This existing REST endpoint (/repositories/<repositoryName>/d2-workflows/<workflowID>/d2-report-tasks) has an additional parameter now called filterName which will get the list of the tasks in a workflow configured for the filtername , from D2-Config. User configures filters with necessary conditions in D2-Config which can be used as a filter in the UI. This should enable the user to filter the tasks of a workflow in UI, depending on the conditions that's been configured.
Linkrel http://identifiers.emc.com/linkrel/d2-report-tasks

Relations

The D2-REST API for relations (link rel: http://identifiers.emc.com/linkrel/d2-relations) allows for fetching relations. However, starting in 23.4, clients can now use this API to fetch custom attributes for custom relations by using the newly introduced request param "object-type" (more information can be found in Swagger documentation).

Starting in 24.4 introduced "Link new content as child" configuration for relation used for inheritance in D2-Config. As per this configuration importing a document by selecting existing document, then the relation between selected and imported document is parent to child respectively for more information refer D2-Admin guide.

JDK-17 Spring 6 Migration

From 23.4 onwards, D2-REST is supported by Spring 6 and should be run on JDK version 17.0.8 or higher. Due to Spring 6 changes, we have introduced a property "rest.requestmapping.trailing.slash.match" in the "rest-api-runtime.properties" file, which allows one to control trailing slash matching in URLs. For example, an API method mapped to "/repositories" will also match "/repositories/" with the default behavior enabled, as the property is set to true by default. However, in the next release, D2-REST will no longer match trailing slashes by default. Users will need to explicitly set the property to true to enable the trailing slash matching. Eventually, trailing slash support will be removed in a future release.

Security Changes

By default, the Documentum REST authentication mechanisms will be available for D2-REST as well. For detailed information about Documentum REST authentication, please refer to the Documentum REST Administrator's Guide.

Currently, D2-REST supports the following authentication modes:

- basic
- basic-ct
- oauth2
- ct-oauth2
- otds_token
- ct-otds_token
- otds_ticket-otds_token
- ct-otds_ticket-otds_token
- otds_password
- ct-otds_password

Additionally, the D2-REST provides a fallback mechanism for the following authentication modes:

- ct-otds_token
- ct-otds_ticket-otds_token
- otds_ticket-otds_token

Starting from version 23.4 onwards, D2-REST will support only full authentication for the oauth2 and ct-oauth2 authentication modes. D2-REST has also removed the source code of the deprecated authentication mechanisms.

Login Logout Audits

D2-REST from 23.4 onwards provides the capability to audit user login and logout for front-end clients like D2-SmartView. The user login entries will be audited with event d2_connect, and the logout entries will be audited with event d2_disconnect, provided "D2 connection audit" is checked in D2-Config under Tools->Documentum audit. An example of an audit entry is given below:

event_name	string_1	string_2	string_3	string_4	string_5	attribute_list
d2_disconnect	D2Smart View	HR	client_ip=162.152.12.45	client_hostname=162.152.12.45	client_location=12.962670064190613, 77.64391671297456	network_location=India;sv_context_UUID =AABH15664879AAP

string_1 value is taken from the request header: X-D2-CLIENT-TYPE
string_2 value is the application name which is taken from header: X-D2-APPLICATION-NAME
string_3 value is the client IP address.
string_4 value is the client hostname. If hostname cannot be resolved, then it is the same as string_3
string_5 is the location (latitude, longitude), which is picked from request header: X-CLIENT-LOCATION
Attribute list is a ';' separated list of attributes. Currently it consists of a network location, which is picked from request header: X-D2-NETWORK-LOCATION and a user-session-specific, globally unique identifier (referred to as sv_context_UUID).

NOTE: Logout entry will not be created in case of session expiry on D2-SmartView.

Encrypted DM Ticket

D2-REST from 23.4 onwards provides the capability to fetch an encrypted DM-ticket via rest API dm-ticket (Example: /repositories/{repositoryName}/dm-ticket). This can be achieved by giving the request param 'secure' as 'true'.

D2 Column Preferences

D2 Column preferences PUT and GET APIs are enhanced to accept and return **d2_sv_widget_state and d2sv_width** as input and output respectively. Definitions of these two attributes are as follows.

d2_sv_widget_state:

{
"table_mode": GRID/LIST/PINNED;(This indicates the mode of the table and there are three possible modes GRID/LIST/PINNED)
"page_size": N, (It's an integer value which indicates the items per page such as 30, 50, 100 etc.)
"left_panel": DEFAULT/TREE/FACET/QUERYFORM (This indicates how the left panel to be displayed. It has three possible values DEFAULT/TREE/FACET/QUERYFORM).

show all version: true/false (This is Boolean value and indicates whether to show all versions or not. false means don't show, and true means show)

"filter": filter name (The name of the active filter which needs to be set)
}

"d2sv_width":123 (An integer value which indicates the width of the column.)

d2-column-preferences GET API

This D2-REST API (link-rel: http://identifiers.emc.com/linkrel/d2-column-preferences) is used for getting the user widget column preferences and is enhanced in 23.4 to return two new attributes, **d2_sv_widget_state and d2sv_width,** in the response.

d2-column-preferences PUT API

This D2-REST API (link-rel: http://identifiers.emc.com/linkrel/d2-column-preferences) is used for updating the user widget column preferences and is enhanced in 23.4 to accept two new attributes, **d2_sv_widget_state and d2sv_width,** in the request payload.

D2 Options API

The D2-Options API (link-rel.: http://identifiers.emc.com/linkrel/d2-options) is an API that lists general D2 option configurations like max file upload size, allowed file extensions, etc. This API has been updated in 23.4 to include default date formats, default datetime formats, and default datetime input formats. Users can use these settings to update their date time preferences.

User Preferences

Logged in user preferences (link-rel.: http://identifiers.emc.com/linkrel/d2-preferences) can be updated using the user preference API. Starting in 23.4, this API can be used to update user date time preferences (which can be fetched using the above mentioned D2 options API).

Collections
Collections are assemblies of documents, forms, cabinets, or folder locations that provide quick access that enable users to perform their work. The following are the REST APIs developed to create and manage collections.

Removing items from a collection:
This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/collection-items) (DELETE) is used to remove objects from a collection. In the request body, you can pass the chronicle IDs of objects to be removed from the collection.

Sample payload for request body:
```
{
   " items_chronicle_id": [
      "id1",
      "id2"
   ]
}
```

Fetching objects from a Collection:

This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/collection-items) (GET) is used to fetch the number of results based on preconfigured value for maxResultSetSize in D2FS.properties. This REST endpoint will always fetch the latest version of the objects present in the collection. It supports filter config name and it is used to apply the filter criteria to this API results. Pagination and sorting is supported. Sort will be applied on all results for dm_sysobject attributes but for custom attributes and dictionary values, sort will be on paginated results only. Below are request parameters can be passed in the URI.

widget-name=WG Favorites&links=true&inline=true

widget-name : widget-name is used to fetch column preferences

links : Boolean value to indicate if linkrels are to be returned in the response.

inline: Boolean value to indicate if all the attributes are to be returned.

sort=<column name> <sort order>

filter=<column name>=<filtervalue>

include-total: Boolean value to indicate Pagination support

Creation of a Collection:

The create collection REST endpoint is used to create a collection (link rel.: http://identifiers.emc.com/linkrel/my-collections). This REST endpoint is used to create an empty collection or a collection with objects by passing object IDs in the request body. This REST endpoint restricts users from creating a collection with the same name.

Sample payload for request body:
1. Creating an empty collection.
   ```
   {
          "properties": {
                  "object_name": "collectionName",
          }
   }
   ```
2. Create a collection and add objects to it.
   ```
       {
   ```

```
            "properties": {

                "object_name": "collectionName",

                "items_object_id": ["id1","id2"]

            }

}
```

Deleting a collection

This REST endpoint is used to delete a collection (link rel.: http://identifiers.emc.com/linkrel/collection)
(DELETE). The r_object_id of the collection to be deleted should be passed in the URI path.

Renaming a collection

This REST endpoint is used to rename the existing collection (link rel.:
http://identifiers.emc.com/linkrel/collection) (PATCH). The r_object_id of the collection to be
renamed should be passed in the URI path.

Sample payload for request body:

```
{

    " properties": {

        "object_name": "New Collection name",

    }

}
```

Fetching collection complete information

This REST endpoint is used to fetch the collection information including a list of inaccessible items
of that collection. Inaccessible items may includes items that are deleted or items that the user no
longer has access to (link rel.: http://identifiers.emc.com/linkrel/collection) (GET).The r_object_id
of the collection to get complete information should be passed in the URI path.

Moving collection items from one collection to other:

This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/collection-items) (POST) is used to move

collection items from one collection to another collection. The r_object_id of the collection to move to

another collection should be passed in the URI path.

Sample payload for request body:

opentext™

```
{
    " items_object_id": [
        "id1",
        "id2"
    ],
"move_from_collection": <id of source collection>
}
```

Adding objects to an existing collection

This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/collection-items) (POST) is used to add objects to an existing collection. In the request body, you can pass multiple object IDs that are to be added into a collection. This REST endpoint does not allow users to add an existing object of the same or different version to a collection.

Sample payload for request body:

```
{
    " items_object_id": [
        "id1",
        "id2"
    ],
"move_from_collection": <>
}
```

Fetching list of collections

This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/my-collections) (GET) will fetch all the collections accessible to the user. Collections will be sorted by last modified date(r_modify_date) in descending order by default.

NOTE: While fetching collections, the user should pass query param **page** as -1 as a query param to get a list of all collections. This is mandatory as currently pagination is not implemented for this API. If page=-1 is not passed, an error response will be sent with the code **E_PAGE_NO_MISSING**.

Share Collections

Shared Collections are the collections that are shared to a user or group by the owner of the collection. The following are the REST APIs developed to create and manage shared collections.

Fetching list of recipients to share collections with:

This REST endpoint (link rel: http://identifiers.emc.com/linkrel/collections-eligible-recipients-for-share) (GET) will fetch a list of all users and groups to share collections to by default.

Instead of returning all the users and groups a query can be configured in D2-Config, under a widget of type '**CollectionWidget**'. For this query to be executed you need to pass the widget name using '**widget-name**' query parameter. If the results returned by query are not of type '**dm_user**' or '**dm_group**' then an error will be thrown.
If the widget name is NOT passed then ALL the results are returned.

Fetching list of shared collections

This REST endpoint (link rel: http://identifiers.emc.com/linkrel/collections-shared-with-me) (GET) will fetch all the shared collections accessible to the user. Shared Collections will be sorted by last modified date(r_modify_date) in descending order by default.

NOTE: While fetching collections, the user should pass the query param **page** as -1 as a query param to get a list of shared collections. This is mandatory as currently pagination is not implemented for this API. If page=-1 is not passed, an error response will be sent with the code **E_PAGE_NO_MISSING**.

Share or Unshare Collection:

The REST endpoint is used to share collection or unshare collection to any user/group. (link rel: http://identifiers.emc.com/linkrel/share-collection)(PUT) .This REST endpoint will take updated list of users/groups info and update collection.

sample payload for request body:

```
{
   "users_and_groups": [
      {
         "recipient_id": "id",
         "recipient_name": "UserName",
         "recipient_role": "reader",
         "recipient_type": "user"
      },
      {
         "recipient_id": "id",
         "recipient_name": "groupName",
         "recipient_role": "reader",
```

```
        "recipient_type": "group"
    }

  ]
}
```

Fetching shared details of collections:

This REST endpoint (link rel: http://identifiers.emc.com/linkrel/share-collection)(GET) will fetch all the users and groups that the collection shared with. The r_object_id of the shared collection should be passed in the URI path.

Business Administration

Admin users can now define users as BAs (Business Administrators). Follow the configuration directions given in the D2 Administration Guide to setup a user as a BA. An Admin can also configure multiple landing pages for users, including BA users. For a BA user, one landing page can have widgets and tiles related to the logged in user's normal everyday functionality, and the other landing page can contain widgets specific to BA related functionality. When logging in, a BA user can switch to either landing page.

If the logged in user has more than one landing page configured, they can choose to store their default landing page in preferences (use attribute: sv_landing_page).

The actions that a BA user can perform on objects not owned by them are limited to the list of BA actions in the applicable BA action config in D2-Config. For example - If a BA user wants to cancel checkout on an object on which the lock was acquired by some other user, then the BA action config (applicable to this BA user) will determine if the BA can perform cancel checkout on the object. Similarly, for Workflows the BA can perform actions such as PAUSE or RESUME only if BA action config allows him to do so.

All Actions performed by a BA (Pause Workflow, Cancel Checkout, Resume Workflow, Change Workflow Supervisor and Abort Workflow) are audited in dm_audit. The audit event names are given below:

S.no.	Action	Audit event name
1	WF Pause	d2_ba_workflow_halted
2	WF Resume	d2_ba_workflow_resumed
3	WF Abort (i.e. HALT)	d2_ba_workflow_aborted
4	Cancel Checkout	d2_ba_cancel_checkout

S.no.	Action	Audit event name
5	Update WF Supervisor	d2_ba_workflow_update_supervisor

* Multiple workflows related D2-REST endpoints now support AdminWorkflowWidget name (widget introduced in 24.2. More details can be found in the D2 Administration Guide) apart from WorkflowOverviewWidget name. The Swagger documentation for such D2-REST endpoints has been updated.

Get all checkout out objects.

This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/d2-all-checked-out-objects) (GET) fetches all the checked-out documents as per the query configured in AdminCheckedOutDocumentWidget. This query is constructed using DQL content and minimum permit level configured in the widget. If no query is configured, then the API fetches all the checked out documents in the system.

Cancel checkout by Business Administrator

This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/checkout (DELETE) cancels the document checked out by other users. Before cancelling the document, it first evaluates the condition configured in "Business Admin Actions" for "Cancel checkout" action. If this action is evaluated to be true then it proceeds to cancel or it throws an error to indicate action is forbidden.

Aviator

OpenText Content Aviator brings the power of generative AI and large language models (LLMs) into OpenText content services platforms, including Documentum. An ideal scenario on how to use Aviator would involve the following steps:

1. Get the Aviator Global Config. and Enable Aviator Global Config. (requires system admin or superuser privileges)
2. List the Aviator enabled folders. (requires system admin or superuser privileges)
3. Enable Aviator on a folder. This has certain limitations like the number of documents in the folder. This is a time taking process since each document inside the folder will be processed. (requires system admin or superuser privileges)
4. Check if the Aviator is enabled on the folder.
5. Get the file count for the folder.
6. Once the documents are processed, select a document inside the folder and initiate the chat.

Enable/Disable Aviator on Folders

This D2-REST API (link-rel.: http://identifiers.emc.com/linkrel/aviator-marker) is used to enable/disable Aviator on folders. The logged in user must be a superuser or a system admin to call this REST endpoint. This API can be used to enable/disable Aviator on multiple folders at once. Refer Swagger for more information.

List Aviator enabled folders.

This D2-REST API (link-rel.: http://identifiers.emc.com/linkrel/aviator-enabled-folders) is used to list all the Aviator enabled folders. The logged in user must be a superuser or a system admin to call this REST endpoint. Any folder with dcis_folder_aspect Aspect attached and dcis_folder_aspect.embd_enabled=true can be qualified as Aviator enabled folder. For the execution of this API refer Swagger documentation.

Chat on Aviator enabled folder.

This D2-REST API (link-rel.: http://identifiers.emc.com/linkrel/aviator-chat) is used for chatting on Aviator enabled folders. There are 3 main request parts for the chat request: *where*, *context* and *chat.*
The "*where*" section will be very similar to the expression-set in the quick search, advanced search etc. But here it will be limited to the minimal expression. For example, the simple value expression will only support EQUAL operator. The *context* section will be empty for the initial chat and for subsequent chat it will use the context from the response. The *chat* section should contain the actual chat query. For the question from the user, the *author* key should be populated as *user*. The content should be question.
For the execution of this API refer to Swagger documentation.

Get Aviator Global Configuration

This D2-REST API is used to get the Aviator global configuration. As a part of this REST endpoint, it will retrieve whether aviator is enabled or not, who enabled/disabled it, when was it enabled/disabled, the embedding URL, and the chat URL. Link rel.: http://identifiers.emc.com/linkrel/aviator-configuration

Enable/Disable Aviator Global Config

This D2-REST API is used to enable or disable the Aviator service globally. System administrators and Superusers will be able to enable or disable Aviator functionality in the Aviator dedicated UI screen. Link rel.: http://identifiers.emc.com/linkrel/aviator-global-status

Check if list folders/documents are aviator enabled.

This endpoint will check if a list of folders/documents are aviator enabled or not. Link rel.: http://identifiers.emc.com/linkrel/aviator-enabled-checker . For more details, refer to Swagger Documentation.

Filter Configuration

Get My Filter Configs

This D2-REST API is used to get filter configs applicable for the logged in user. This endpoint will fetch all the matrix evaluated D2-Config filters. To fetch filter configs specific to a particular folder or custom extended folder type pass the request parameter named folder_id, filter configs are evaluated based on type context. If the folder id is not provided, then filter configs are evaluated based on user context mapping. When no filter configs are mapped, All and Current configs are returned. Link rel.: http://identifiers.emc.com/linkrel/my-filter-configs.

Filter config support in object facets API

The object facets API (link rel.: http://identifiers.emc.com/linkrel/object-facets) now supports filter config name. This filter config name is used to apply the filter criteria to this API results. The end user should first fetch the list of filter configs applicable to the logged in user. Then you can send this filter config name to the object facets API to apply the filter. You can also choose to save this filter config name in your preferences. If a filter config is not provided, the filter config present in this user's preferences against the provided widget name and type is applied.

If you do not pass the filter config name to this API or have any filter name configured in your preferences for this widget-name and widget-type combination, then the first default config will be applied (from the list of filter configs marked in user's context + the list of filter configs marked in default context ordered by filter config name).

If you do not pass the filter config name to this API or have any filter names configured in your preferences for this widget-name and widget-type combination and have no default filter configs in your context or the default context then the first config (from the list of filter configs marked in user's context + the list of filter configs marked in default context) is taken and applied.

If you do not pass the filter config name to this API or have any filter names configured in your preferences for this widget-name and widget-type combination and have no filter configs in your context or the default context then no filter is applied.

Fetching Facet data with Filter config

The following use case has been handled for processing facets with Filter Config.

1. If the active D2 Filter Config has a DQL criteria and if the resultant document list size is beyond maxResultsize then facets will not process and a flag ("facet-processed": false) will be returned to indicate the same.
2. If the active D2 Filter Config has a DQL criteria and if the resultant list size is below maxResultsize then facets will be processed, in this case "facet-processed" flag will be true.
3. If the filter criteria is empty, then facet will be processed irrespective of the resultant list size.

Note: maxResultsize for doc-list widget can be configured in D2-Config (under Maximum Results Filtering section in Widget Config) or in D2FS.properties.

Sorting enhancement in object facets API

The object facets API now sorts attributes by executing DQL instead of Java In-memory sorting which was done earlier.
Note: Sorting on repeating attributes is not supported and may return irrelevant data.

Filter config support in collection items API

The collection items API (link rel.: http://identifiers.emc.com/linkrel/user-defined-collection-items) now supports filter config name. This filter config name is used to apply the filter criteria to this API results. If a filter config is not provided, you apply the filter config present in the user's preferences against the provided widget name and type.

Filter config Show Folder Option Support in Object Facets API
The filter config now has "Show Folder" option based on which folders will be added /shown irrespective of the DQL condition. The Object Facets API has been modified to support the "Show Folder" option.

File Count

The file count API (link rel.: http://identifiers.emc.com/linkrel/file-count) fetches the file count for a particular folder/cabinet. It takes a Boolean request param "descend "which determines if the count should be fetched from sub-folders. If it is true, then the API will recursively fetch the file count from sub-folders. If false, then only the immediate file count will be fetched. The default value is false.

Object specific permissions API.

There is a REST API which takes object-ids and permission-levels and checks if the object/s have that level permission or not. There was a limitation for the GET endpoint, due to which from 24.2 we have added another variant with POST endpoint. Here we take permission-level as request param. and the object-ids are passed in the request body. Refer to Swagger Documentation for more details.

Retirement of RADL

D2-REST documentation was migrated to Swagger in 23.2.

Previously RADL documentation URL was:
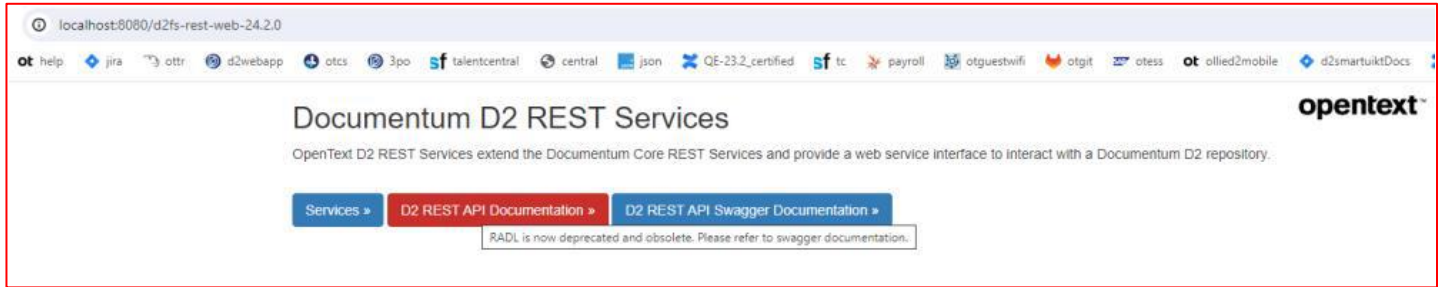
http://<host>:<port>/<war_file_name>/static/d2-rest/radl/index.html

And now it's changed to:

http://<host>:<port>/<war_file_name>/static/d2-rest/**radl-deprecated**/index.html

In the d2fs-rest-web (24.2) war file, the button D2 REST API Documentation >> is now changed to red color, with tool tip text: *RADL is now deprecated and obsolete. Please refer to swagger documentation.*

Please note that starting in 23.4 Swagger is the official D2-REST API documentation and RADL is no longer being updated. For API specs and details, please refer to Swagger.

D2 Objects Locations API

API Changes in D2 Locations:

This REST endpoint (link rel.: http://identifiers.emc.com/linkrel/d2locations) (GET) is modified in 24.2 to retrieve folders and/or VD locations if a single object is passed in the request. For multiple objects, it only retrieves folder locations. It accepts multiple objects in the request.

The object-ids of the object should be passed in the URI path and location_type can be passed in URI, this is optional.

object-ids - The object id(s) of the object whose actions are required. One or more Object IDs can be passed as part of this parameter. Object IDs can be passed in the format - ?object-ids=0903684f80198367&object-ids=0903684f802c0eef

location_type=FOLDER|VD (New query param added in 24.2 to facilitate fetching of parent VD locations)

If the value is set to FOLDER, then it retrieves the folder locations of the document.

If the value is set to VD, then it retrieves the Virtual Document locations of the document.

If the value is set to FOLDER|VD, then it retrieves both the Folder and Virtual Document locations of the document.

If this query parameter is not passed in the request, it retrieves all the folder locations for the document.

If the value of the location_type query param is set to 'VD' or 'FOLDER|VD' and multiple objects are passed in the request, it throws an exception saying, 'Requested Locations for multiple objects is not supported.'.

opentext™

Export Audit and Locations API

Export Audit API

This REST endpoint (/repositories/{repositoryName}/objects/{id}/audit-export) (GET) will export Audit data into an Excel file. Link rel is "object-audit-export".

This API will accept object_id(supports workflow/object id for audit) and export all the Audit related to it. We can export specific version Audit data by sending all_versions = false.
Assumptions: We always export audit related to document.
The below query params are used to pass in URI.
all_versions - Request param to be used to export Audit events of a document for all versions/specific version. The default value is true.
export_type - Request param to be used to mention the format type of audit export. The default value is XLSX. The allowed value is XLSX only. Other format types can be expected in the future.
filter - Results can be filtered by any attribute ex: filter = audited_obj_id eq '<object_id>' and contains(event_name, ''<name>'')
file_name [Optional] - Request param used to pass a custom file name for audit export document.

Export Locations API

This REST endpoint (/repositories/{repositoryName}/objects/{id}/locations-export) (GET) will export locations data into an Excel file. Link rel is "object-locations-export ".

This API will accept object_id and export all the Locations related to it. We can export all the location types such as the FOLDER and VD the object is linked to.
The below query params are used to pass in URI.
export_type - Request param to be used to mention the format type of locations export. The default value is XLSX. The allowed value is XLSX only. Other format types can be expected in the future.
file_name [Optional] - Request param used to pass a custom file name for locations export document.

Grouping Audit Trails API

Get Audit Trail Facets by relative date and count of Audit in each group

This REST endpoint (/repositories/{repositoryName}/objects/{id}/audit-trail-facets-by-relative-date) (GET) will retrieve Audit related count data grouped by fixed date facets. Link rel is "audit-trail-facets-by-relative-date ".
This API will return the groups by date and total count of Audits in each group for a given document and the start & end dates of each facet.
Getting groups by date is dynamic w.r.t day the audit is viewed. Monday to Sunday is considered as a week and grouping will be done accordingly. Facet with no audit will not be populated in the response.
Following are the relative date facets applicable as per the implementation.

TODAY, YESTERDAY, CURRENT_WEEK, LAST_WEEK, TWO_WEEKS_AGO, THREE_WEEKS_AGO, FOUR_WEEKS_AGO, EARLIER_THIS_MONTH, LAST_MONTH and OLDER

event_source = Automatic/Manual/All. Specifies whether event_source is automatic, manual or all. The default value is all, which retrieves both Automatic and Manual events.

Get Audit Trail Facets by event source and count of Audit in each group

This REST endpoint (/repositories/{repositoryName}/objects/{id}/ audit-trail-facets-by-event-source) (GET) will retrieve Audit related count data grouped by event source type. Link rel is "audit-trail-facets-by-event-source".

This API will return the groups by event source such as Automatic and Manual and total count of Audits in each group for a given document.

The below query params are used to pass in URI.

all_versions - Request param to be used along with audited_obj_id to get Audit events of a document for all versions. Default value is true.

filter - can send any filter conditions. API will return the count of the result set after performing the filter condition.

Audit Trails

Starting 24.2 it is mandatory to select Audit config for Recently Accessed widgets.

API Changes in Audit Trails

This existing REST endpoint (link rel.: http://identifiers.emc.com/linkrel/audit-trails)(GET) is modified in 24.2 to fetch all the Audit trails. audited_obj_id, all_versions, view, filter, sort can be passed in the URI path. Audits specific to documents can also be retrieved by sending 'audited_obj_id' query parameter. This API will honor Audit configuration and other D2 config if audited_obj_id is sent either in filter or request parameter. The below query params are used to pass in URI to fetch the audit trails.

filter - Results can be filtered by any attribute ex: filter = audited_obj_id eq '<object_id>' and contains (event_name, ''<name>'')

sort - Default sort order is by time_stamp. Results can be sorted ascending or descending order by any attribute ex: sort=event_name desc

view - view=:all can be sent if all attributes are to be returned. view=<list of attributes> can be used to return any specific list of attributes

audited_obj_id = Optional parameter when sent in request gets audit events specific to document. If audited_obj_id is sent in request param, then it is assumed that filter will contain other fields than

audited_obj_id. If audited_obj_id is not sent in request param and the filter contains 'audited_obj_id', then all_versions request param is not considered.

all_versions - Request param to be used along with audited_obj_id to get Audit events of a document for all versions. The default value is true.

Paging paramaters
   items-per-page - Number of items per page. If no pagination parameters are sent, 'rest.paging.default.size' property configured in rest-api-runtime properties file is considered for page count. Items per page cannot exceed 'rest.paging.max.size' property configured in rest-api-runtime properties
   page - Page number can be sent to retrieve respective page.
   include-total = true returns total count of Audits as per filter specified in URL

Send inline=true to fetch details of Audits

send links=true to get link rels

NOTE: Existing D2-Rest Audits API (/repositories/{repositoryname}/objects/{id}/audits) is deprecated in 24.2 Release.

Microsoft Information Protection (MIP)

D2-REST now supports MIP labelled content for PDF rendition creation provided MIP config has been created and correctly setup by Administrator through MIP config API.

Create MIP configuration

This REST endpoint (link-rel.: http://identifiers.emc.com/linkrel/mip-configuration) is used to create MIP global config. The 'is_mip_enabled' flag is used to determine if the MIP is enabled globally. 'mip_client_credentials' should contain information about tenant id, client id and client secret. The 'mip_application_settings' should contain a list of feature names like SEARCH and THUMBNAIL. It should also contain MIP labels allowed for SEARCH and THUMBNAIL feature. For MIP protected document features like Search/Thumbnail will only work if the content is protected by 1 of the labels present in the MIP label list for search/thumbnail. This API can only be accessed by a user with minimum system administrator permissions.

GET MIP configuration

This REST endpoint (link-rel.: http://identifiers.emc.com/linkrel/mip-configuration) is used to get MIP config. This API returns the MIP global flag, the application settings and the client credentials (client secret cannot be retrieved). This endpoint can be accessed by a system administrator or superuser only.

opentext™

MIP connection validation

Before creating the MIP configuration, you can use this API (link rel.: http://identifiers.emc.com/linkrel/mip-connection-validator) to test if we have correct values for tenant_id, client_id and client_secret. These 3 attributes should be sent in the request body (POST) for this call.

List available MIP Sensitivity labels

This API will fetch the latest available sensitivity label from Microsoft Purview portal for the Azure tenant (link rel.: http://identifiers.emc.com/linkrel/mip-sensitivity-labels). To reflect the newly created Sensitivity label in this API, it might take up to 24 hours.

Support aspect attribute for Objects Facets API

This object Facets API (link rel.: http://identifiers.emc.com/linkrel/object-facets) now returns Aspect attributes as well. For an MIP labelled object , MIP aspect attributes are returned in response. The aspect attribute can be added to columns either through the widget config or through the column preference API.

Support Aspect types in Display Types API

The Display Types API (link rel: http://identifiers.emc.com/linkrel/d2-display-types) now returns the configured aspect type's data along with other system/custom types. This configuration can be done in Display Configuration under Interface in D2-Config.

Support Aspect data retrieval for Quick Search

The Quick Search API (Endpoint: /repositories/{repositoryName}/quickSearch (POST)) now returns data for the configured aspect attribute along with other attributes. To index and search based on the aspect data, some configuration needs to be done on the search engine side (xPlore/DCTM Search) which can be found in their respective documentation.

Support Aspect data retrieval for DQL based Query form Search

The Query form API (Endpoint: /repositories/{repositoryName}/d2-saved-searches/query-forms/{queryFormId}/results (POST)) now returns the configured aspect attribute data along with other attributes data in the response for DQL Query based Config. In Query Form Config, the aspect attribute can be added either directly in the DQL along with other attributes or through user preferences.

Collaborative Editing

Get native annotations of a document

This REST endpoint(/repositories/{repositoryName}/objects/{id}/native-annotations-for-collaboration-edit) is used to retrieve content and native annotations details of the document. If it is a task, it retrieves the annotation details of each associated document. Link rel: http://identifiers.emc.com/linkrel/collaborative-edit. Use query param inline=true to get the annotation details.

Import Renditions service

This import rendition API (link rel: http://identifiers.emc.com/linkrel/d2-import-renditions-urls) can be used to import renditions for a selected document.

Lifecycle (LC) development

LC API to fetch Lifecycle Config for the given transition

This rest endpoint is used to fetch Lifecycle Config, for a given transition, by accepting object ids and transition target state. The Link Rel (http://identifiers.emc.com/linkrel/d2-objects-lifecycle-configs) can be accessed in response of this LC API (/repositories/<repo>/d2-objects-lifecycle-configs).

LC API to initialize Lifecycle on a document (legacy flow)

The API must accept object id and target state of the transition. And make the state transition to the target state. This flow is like D2 classic. PUT request on this resource will make the lifecycle transition to the target state. The API will evaluate the entry conditions and transition conditions for each object before applying lifecycle state transition. The Link Rel (http://identifiers.emc.com/linkrel/d2-objects-lifecycle-state
) can be accessed in response of this LC API (/repositories/<repo>/d2-objects-lifecycle-state). The same API is used to initialize the LC on the document using the target state value as the starting state of the LC. The API will support esign , property page update and multi document flow in the future updates.

LC API to evaluate Entry conditions and transition conditions

This LC API is used to evaluate the entry and transition conditions of document object(s) passed in the request and then status is returned as the response for each object passed in the request. The API accepts multiple object IDs and a target state. It returns the localized error message for each of the failure cases. The Link Rel
(http://identifiers.emc.com/linkrel/d2-objects-lifecycle-conditions-checker
) can be accessed in response of the LC API (/repositories/<repo>/d2-objects-lifecycle-conditions-checker) .

OTDS Support for Life Cycle Change State E-Signoff

When user initiates a D2 LC transition, if the state is configured to obtain users e-sig and if the new external user is a member of the group defined for IDP external user; then the user that requires e-signature will need to see the IDP electronic flow for e- signature panel and not the legacy D2 e-signature flow .
To support this , the following changes are made in the configuration for the LC API (/repositories/<repo>/d2-objects-lifecycle-configs) :

API Changes in D2 Lifecycle Config Fetch

The changed Response detail (External Sign off)

url - This URL will be the OTDS login URL. Redirect URL should be appended to this URL by the UI.

Ex: http://otds.otblr.com:6060/otdsws/login?response_type=id_token&client_id=otblr_d2rest&prompt=login&authc ontext=sign&scope=openid&nonce=MlCnP6fw0AF2UbO4eKxOPrA5bscuIPgC8f9qKTWDGq1tqEJbcWsX8JewqB_EH mdI&redirect_uri=http://d2vm.otblr.com:8080/d2sv/ui

nonce_token - This will be the unique id for the OTDS login request. The same value will be needed when e-sign off for lifecycle change state

API Changes in LifeCycle Init | LifeCycle Apply API

The changed Request Body attribute (signoff-inputs) –
signoff_login - For OTDS authentication (ct-otds_token mode), the signoff_login will be empty
signoff_password - This should be the OAuth token that would be getting from the OTDS machine.
nonce_token - The nonce token is the unique id that will be getting from the LifeCycle Config API

D2 REST - As a user I should be able to send email to the list of users in the email-list when lifecycle succeeds.

When the lifecycle action with send email is applied an email will be sent to the users available in the email-list. This is the existing behavior in classic and the same feature parity is maintained between SmartView/Rest and Classic when user applies the LC state change configuring "send email" action type for the selected the LC. If Send Email Action Type is set in LC config as follows, then email will be sent to all the email ids configured in selected "emailing list". If no emailing-list is configured for "send email" action, then we don't do anything. The LC API that's been used for this feature to work is **repositories/d2repo/d2-objects-lifecycle-state?inline=true**. It calls the D2FS code and hence 'send email' action will work fine when LC is applied.

D2-Config: As D2 PMs we want to provide an OOTB default email template that can be used for LC state changes send mail mailing list.

In some cases, especially when the bulk processing option for LC state changes is enabled and the processing takes a long time, the user will need to be informed of the LC state outcomes via an email. In these cases, the customer is expected to configure the LC state change to have a Send mail action and indicate a mailing list to use. So as per this feature, sending email is for single as well as multiple file LC state change. It just depends on the value of the action type "send email". If it contains the "Default D2 Lifecycle State Change report" emailing list, then an email notification with the report as per the mock in the Jira should be sent out irrespective of whether its single file or multiple file LC state change. The LC API that's been used for this feature to work is **repositories/d2repo/d2-objects-lifecycle-state?inline=true**

LC API to apply LC state transition in homogenous flow with EC & TC

This LC API will accept array of object ids and target state of the transition. [along with e-sign, property bag etc.]. And make the state transition to the target state. This Link REL (http://identifiers.emc.com/linkrel/d2-objects-lifecycle-state) can be accessed in response of the LC API (**/repositories/<repo>/d2-objects-lifecycle-state**).

LC API to honor mass update properties bag during LC state transition

The LC change state API will accept a property bag and update the properties of the objects. Properties bag is a common set of properties which shall be applied to all the objects passed as an input to this API. This is a list of individual object properties. The user will send a list of properties along with the object ids, which he/she wants to update during the LC state transition. This API will accept Properties Bag/ Properties List. If user passes both (Properties bag and list) API will respond with a proper error message. This Link REL (http://identifiers.emc.com/linkrel/d2-objects-lifecycle-state) can be accessed in response of the LC API (/repositories/<repo>/d2-objects-lifecycle-state) .

LC state transition homogeneous flow asynchronously to send email

This API will accept array of object ids and target state of the transition. [along with e-sign, property bag etc.]. And make the state transition to the target state. For homogeneous and asynchronous lifecycle state transition "/d2-objects-lifecycle-state" API will send mail to the Email List config as configured in LC configuration in D2-Config. To make the LC config Asynchronous and Set the summary mail config, in Life Cycle Config Mark Allow Bulk transitions and Skip UI processing and set the Summary mail config. This Link REL (http://identifiers.emc.com/linkrel/d2-objects-lifecycle-state) can be accessed in response of the LC API (**/repositories/<repo>/d2-objects-lifecycle-state**)

Lifecycle API to evaluate homogeneous flow

This API will evaluate whether the requested Lifecycle state transition is homogeneous or not. The API must accept multiple object ids and a target state. It must evaluate if the transitions requested constitute a homogeneous flow or a heterogeneous flow. It must also return for each object id the Link REL for the LC config API. This Link REL (http://identifiers.emc.com/linkrel/d2-objects-lifecycle-configs) can be accessed in response of the LC API (**/repositories/<repo>/d2-objects-lifecycle-configs**).

Menu action label is required as part of config data for LC

In Smart UI menu label name is showing as dialog title for confirmation view and property page view. When the user performing heterogeneous LC, multiple configurations have different menu labels, user want to see proper menu label name as title for each file. This Link REL (http://identifiers.emc.com/linkrel/d2-objects-lifecycle-configs) can be accessed in response of the LC API **(/repositories/<repo>/d2-objects-lifecycle-configs)**.

D2-REST: As D2 Admin I expect user to see a single e-sig dialog & IDP login during multi-file LC state change actions if user is using external IDP when LC state change requires e-signature.

Initially , When user initiates a multi-file D2 LC transition, if the files share the same LC state change configuration and if the state is configured to obtain users e-sig, and the user is required to use an external IDP for e-signatures, then the user will only be asked to provide e-sig once for all the documents electronic signature panel in runtime D2SV UI. So as per this feature, following changes are made in the configuration:

API For D2 Lifecycle Config Fetch

Following changes are done for the API /repositories/<repo>/d2-objects-lifecycle-configs

url - This URL will be the OTDS login URL. Redirect URL should be appended to this URL by the UI.

Ex: http://otds.otblr.com:6060/otdsws/login?response_type=id_token&client_id=otblr_d2rest&prompt=login&authcontext=sign&scope=openid&nonce=MlCnP6fw0AF2UbO4eKxOPrA5bscuIPgC8f9qKTWDGq1tqEJbcWsX8JewqB_EHmdI&redirect_uri=http://d2vm.otblr.com:8080/d2sv/ui

nonce_token - This will be the unique id for the OTDS login request. The same value will be needed when e-sign off for lifecycle change state

API Details For LifeCycle Init | LifeCycle Apply API

Following changes are done for the API /repositories/<repo>/d2-objects-lifecycle-state

The changed Request Body attribute (signoff-inputs) : signoff_login - For OTDS authentication (ct-otds_token mode), the signoff_login will be empty
signoff_password - This should be the OAuth token that would be getting from the OTDS machine.
nonce_token - The nonce token is the unique id that will be getting from the LifeCycle Config API

Installation Guide

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

opentext™

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

Starting 24.4 backward compatibility for trailing slash (/) has been removed hence this property "**rest.requestmapping.trailing.slash.match**" no longer supported by D2

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

    1) 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