

MOMENTUM 2016

HACK -A- THON

DOCUMENTUM
REST API
LAB

Table of Contents

Overview	3
Introduction to the Documentum REST API	3
Starting the Server	3
Documentum REST Services API Reference	4
Exploring the Documentum REST API with Postman	9
Basic Navigation	9
Get the Home Document	9
Get the Repositories List, Pick a Repository	10
Get the Cabinets List, Pick a Cabinet	13
Get the Folders List for the Cabinet, Pick a Folder	15
Get the Documents List for the Folder	16
Creating, Reading, Updating, and Deleting Documents	17
Create a Document	17
Update a Document	19
Create a Document with Multipart MIME	20
Get Document Content	22
Delete a Document	23
Collections	25
Inlining Collections	26
Showing Totals	26
Paging	27
Sorting	28
Filtering	28
Views	28
Queries and Full-text Search	29
DQL Queries	30
Full-text Search	30

Overview

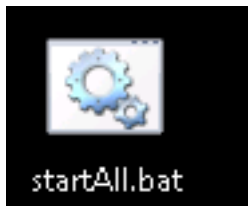
Introduction to the Documentum REST API



This EMC lab will walk you through the basics of the Documentum REST API, using Postman to issue basic commands and inspect the results, and becoming familiar with the API documentation.

Starting the Server

To start the server, click on the startAll.bat icon on the desktop:



To make sure the REST Service is running, go to a web server and open `localhost:8083/dctm-rest/`:



If the REST Service is running you will see the following screen:

Documentum REST Services

EMC Documentum Platform REST Services provides you with a powerful yet simple web service interface to interact with Documentum.

Services »

If you click on the [Services »](#) button it returns the following XML document, which is a **home document** that lists the resources available on this server:

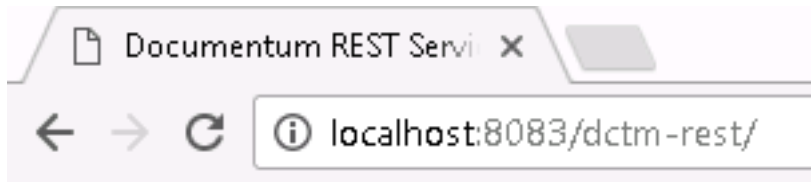
```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<resources xmlns="http://identifiers.emc.com/vocab/documentum" xmlns:dm="http://identifiers.emc.com/linkrel/documentum">
  <resource rel="http://identifiers.emc.com/linkrel/repositories">
    <link href="http://localhost:8083/dctm-rest/repositories"/>
    <hints>
      <allow>
        <i>GET</i>
      </allow>
      <representations>
        <i>application/xml</i>
        <i>application/json</i>
        <i>application/atom+xml</i>
        <i>application/vnd.emc.documentum+json</i>
      </representations>
    </hints>
  </resource>
  <resource rel="about">
    <link href="http://localhost:8083/dctm-rest/product-info"/>
    <hints>
      <allow>
        <i>GET</i>
      </allow>
      <representations>
        <i>application/xml</i>
        <i>application/json</i>
        <i>application/vnd.emc.documentum+xml</i>
        <i>application/vnd.emc.documentum+json</i>
      </representations>
    </hints>
  </resource>
</resources>
```

Documentum REST can provide resources in either XML or JSON. In this tutorial, we will focus on JSON, the format used by most programmers. To do that, we will use Postman, a widely used REST API debugger.

Documentum REST Services API Reference

The Documentum REST Services API reference is a hyperlinked HTML document that describes the REST API. As you work through this tutorial, you can browse the API reference and compare the documentation to the results you see in your web browser or in Postman.

A link to the API reference is available in the lower left-hand corner of the HTML page found at `localhost:8083/dctm-rest/`:



Documentum REST Services

EMC Documentum Platform REST Services provides you with a powerful yet simple web service interface to interact with Documentum.

[Services »](#)

[API Documentation \(experimental\)](#) | (c) 2016 EMC Corporation. All rights reserved.

Click on the blue text highlighted in the above image - you will see a table of contents on the left and a table of states on the right.

Let's explore a few resources now to get a feel for the documentation.

Documentum REST Services

- [States](#)
- [Link Relations](#)
- [Property Groups](#)
- [URI Parameters](#)
- [Media Types](#)
- [Custom Headers](#)
- [Status Codes](#)
- [Resources](#)
- [Authentication](#)

Documentum REST Services

API References for Documentum REST Services 7.3 (Bedrock) Release.

States

State	Transition	Result State
Start	Entry point	home-doc
home-doc	Get product information	product-info
home-doc	Get repositories	repositories
repositories	Find repository	repository
repository	Get cabinets	cabinets
repository	Get checked out objects	checked-out-objects
repository	Create batch	batch
repository	Create batch with multipart/related	batch
repository	Get batch capabilities	batch-capabilities

In a RESTful API, the client enters the service using a published URI. After entering the service, a client uses navigation or queries to progress to other states. A client can cache URIs as they are discovered, but it should not hard code the location of resources or attempt to parse URIs.

In the states table, entering the service using the published URI is represented as the transition from the Start state to the home document. If you click on the **Entry point** transition in the middle of the first row it takes you to a description of this transition:

State: Start

Transitions

Transition	Link Relation	Method	Result State
Entry point	[None]	GET	home-doc

Transition: Entry point

By performing HTTP method **GET** on state **Start**.

Response:

Result States:

- **home-doc**

Media Types:

- [application/home+xml](#) (home-doc)
- [application/home+json](#) (home-doc)

This description says that you can enter the service by doing GET on a known URI, and the result state is **home-doc**, which corresponds to a document returned by the server.

Every state except for the Start state corresponds to a document retrieved from the REST server. The **Media types** list links to a description of the physical format of the document in each available media type. In this case, it links to a description of the XML and JSON media types for a home document. You can click on this to see what a home document looks like.

Both [application/home+xml](#) and [application/home+json](#) are IANA standard media types. We have already seen the XML representation of a home document, here is the JSON representation:

```
{
  "resources": {
    "http://identifiers.emc.com/linkrel/repositories": {
      "href": "http://localhost:8080/dctm-rest/repositories",
      "hints": {
        "allow": [ "GET" ],
        "representations": [
          "application/xml",
```

```
"application/json",  
    "application/atom+xml",  
    "application/vnd.emc.documentum+json"  
]  
}  
},  
"about": {  
    "href": "http://localhost:8080/dctm-rest/product-info",  
    "hints": {  
        "allow": [ "GET" ],  
        "representations": [  
            "application/xml",  
            "application/json",  
            "application/vnd.emc.documentum+xml",  
            "application/vnd.emc.documentum+json"  
        ]  
    }  
}  
}  
}
```

Each state contains links that identify choices provided to the client by the server. To see the choices available in the **home-doc** state, click on **home-doc** in the results state list. This will take you to the following table:

State: home-doc

Transitions

Transition	Link Relation	Method	Result State
Get product information	<code>about</code>	<code>GET</code>	product-info
Get repositories	<code>http://identifiers.emc.com/linkrel/repositories</code>	<code>GET</code>	repositories

In this table, the **Link Relation** column identifies names associated with the URIs of resources. For instance, in the **Get repositories** transition we will do a GET on the URI labeled with the link relation <http://identifiers.emc.com/linkrel/repositories> to get to the **repositories** state, which corresponds to a document that contains available repositories and the operations a client can perform using this list. If you click on the **Get repositories** transition you will see the details of the API for this transition, including URI parameters that determine whether repositories in the list are inlined.

Transition: Get repositories

By performing HTTP method **GET** on link relation `http://identifiers.emc.com/linkrel/repositories` of state `home-doc`.

Request:**URI Parameters:**

- `inline`
- `links`

Response:**Result States:**

- `repositories`

Media Types:

- `application/atom+xml` (`repositories`)
- `application/vnd.emc.documentum+json` (`repositories`)

Let's compare the information in these tables to the JSON representation of a home document:

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

In this JSON representation, `"resources"` contains a list of link relations. The first one, `"http://identifiers.emc.com/linkrel/repositories"` identifies a list of repositories. The URI used to retrieve this list is in the href attribute: `"href"` property: `"http://localhost:8080/dctm-rest/repositories"`. The `"hints"` property identifies HTTP methods that can be applied to this resource and the available media types. The JSON representation corresponds to the description in the API documentation. The XML representation corresponds to the same description, but using a different format.

Now try clicking around in this file to see how to perform specific tasks. For instance, starting with the home document, how would you find a list of cabinets in a repository? A list of folders in a cabinet? A list of documents in a folder?

Exploring the Documentum REST API with Postman

Postman is a widely used tool for debugging and testing REST APIs. It is also an excellent tool for exploring a REST API hands-on. In this lab we will use the Postman Chrome App, which is already installed on your computers.

Basic Navigation

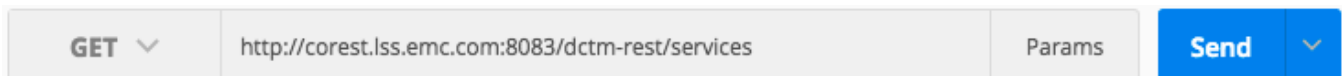
In this section we will do basic navigation in repositories, cabinets, and folders. We will do the following:

- Retrieve the home document
- Use the link relation <http://identifiers.emc.com/linkrel/repositories> to get a list of repositories
- Select the first repository from the list ("entry" [1])
- Use the link relation <http://identifiers.emc.com/linkrel/cabinets> to get the cabinets for the repository
- Select the first cabinet from the list ("entry" [1])
- Use the link relation <http://identifiers.emc.com/linkrel/folders> to get the folders in the cabinet

We will start with the home document.

Get the Home Document

After starting Postman, enter <http://localhost:8083/dctm-rest/services> into the address bar and press the blue send button:



Postman displays the result of the HTTP request. The status code is shown in the upper right corner, the body is formatted and displayed:

```

1  {
2    "resources": {
3      "http://identifiers.emc.com/linkrel/repositories": {
4        "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories",
5        "hints": {
6          "allow": [
7            "GET"
8          ],
9          "representations": [
10             "application/xml",
11             "application/vnd.emc.documentum+json",
12             "application/atom+xml",
13             "application/json"
14           ]
15         }
16       },
17       "about": {
18         "href": "http://corest.lss.emc.com:8080/dctm-rest/product-info",
19         "hints": {
20           "allow": [
21             "GET"
22           ],
23           "representations": [
24             "application/xml",
25             "application/vnd.emc.documentum+json",
26             "application/vnd.emc.documentum+xml",
27             "application/json"
28           ]
29         }
30       }
31     }
32   }

```

To see the response headers, choose the **Headers** tab:

```

Content-Type → application/home+json;charset=UTF-8
Date → Mon, 24 Oct 2016 13:31:19 GMT
Server → Apache-Coyote/1.1
Transfer-Encoding → chunked

```

Note the content type. This is a JSON Home document, a media type defined in <https://mnot.github.io/I-D/json-home/>. It functions like a “table of contents” for the service, identifying the top-level resources. In the following excerpt, <http://identifiers.emc.com/linkrel/repositories> is a **link relation**, a label that identifies the kind of resource. Do not attempt to apply HTTP methods to this URI, search for it when you need to find a link to the repositories. The associated href property is the physical location of the resource.

```

{
  "resources": {
    "http://identifiers.emc.com/linkrel/repositories": {
      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories"
    }
  }
}

```

Get the Repositories List, Pick a Repository

Click on the URI underlined in the above image. Postman copies it into the address bar. Hit the Send button.

HACKATHON - DOCUMENTUM REST API

A Postman interface showing a GET request to the URL `http://corest.lss.emc.com:8080/dctm-rest/repositories`. The interface includes a dropdown for the HTTP method (set to GET), a text field for the URL, a 'Params' tab, and a blue 'Send' button.

The result of the above GET contains a list of repository resources. Choose the first entry in the list:

```
"entries": [  
  {  
    "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO",
```

Click on the URI underlined in the above image. Before you hit send you need to configure authentication. Choose basic authentication, which relies on a user name and a password:

The Postman interface with the 'Authorization' tab selected. The 'Type' dropdown is set to 'Basic Auth'. Below it, the 'Username' field contains `{{user}}` and the 'Password' field contains `{{password}}`. A 'Show Password' checkbox is checked.

In Postman, `{{user}}` and `{{password}}` refer to variables defined in the environment. Make sure that the dropdown in the upper right contains the name "Documentum", an environment configured with the correct password. You can click on the eye icon to display the values bound in this environment. `{{user}}` evaluates to "dmdadmin", `{{password}}` evaluates to "password".

A screenshot of the Postman environment configuration for 'Documentum'. The configuration includes the following variables:

Variable	Value
User	dmdadmin
Password	password
Root	<code>http://{{host}}:{{port}}/dctm-rest</code>
Repository	self
Host	corest.lss.emc.com
Port	8080

Now that authentication is enabled, press the Send button. The result contains a large number of links. Here is a partial list:

X`

```

13  "links": [
14    {
15      "rel": "self",
16      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO"
17    },
18    {
19      "rel": "http://identifiers.emc.com/linkrel/cabinets",
20      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets"
21    },
22    {
23      "rel": "http://identifiers.emc.com/linkrel/checked-out-objects",
24      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/checked-out-objects"
25    },
26    {
27      "rel": "http://identifiers.emc.com/linkrel/current-user",
28      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/currentuser"
29    },
30    {
31      "rel": "http://identifiers.emc.com/linkrel/current-user-preferences",
32      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/currentuser-preferences"
33    },
34    {
35      "rel": "http://identifiers.emc.com/linkrel/users",
36      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/users"
37    },
38    {
39      "rel": "http://identifiers.emc.com/linkrel/groups",
40      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/groups"
41    },

```

These links correspond to the transitions associated with a repository, which you can see in the API Reference:

Transition	Link Relation	Method	Result State
Get cabinets	http://identifiers.emc.com/linkrel/cabinets	GET	cabinets
Get checked out objects	http://identifiers.emc.com/linkrel/checked-out-objects	GET	checked-out-objects
Create batch	http://identifiers.emc.com/linkrel/batches	POST	batch
Create batch with multipart/related	http://identifiers.emc.com/linkrel/batches	POST	batch
Get batch capabilities	http://identifiers.emc.com/linkrel/batch-capabilities	GET	batch-capabilities
Execute DQL query	http://identifiers.emc.com/linkrel/dql	GET POST	dql-query
Get users	http://identifiers.emc.com/linkrel/users	GET	users
Get groups	http://identifiers.emc.com/linkrel/groups	GET	groups
Get current user	http://identifiers.emc.com/linkrel/current-user	GET	current-user
Get current user preferences	http://identifiers.emc.com/linkrel/current-user-preferences	GET	current-user-preferences
Get formats	http://identifiers.emc.com/linkrel/formats	GET	formats
Get relation types	http://identifiers.emc.com/linkrel/relation-types	GET	relation-types
Get relations	http://identifiers.emc.com/linkrel/relations	GET	relations
Get network locations	http://identifiers.emc.com/linkrel/network-locations	GET	network-locations
Get types	http://identifiers.emc.com/linkrel/types	GET	types
Execute full-text search	http://identifiers.emc.com/linkrel/search	GET	search
Get aspect types	http://identifiers.emc.com/linkrel/aspect-types	GET	aspect-types
Get saved searches	http://identifiers.emc.com/linkrel/saved-searches	GET	saved-searches
Get search templates	http://identifiers.emc.com/linkrel/search-templates	GET	search-templates
Get acls	http://identifiers.emc.com/linkrel/acls	GET	acls

Get the Cabinets List, Pick a Cabinet

One of the links in the Repository is associated with the list of cabinets for the repository. In the body of the response, look for the link with the link relation <http://identifiers.emc.com/linkrel/cabinets> and click on the value of the associated href:

```

18 {
19   "rel": "http://identifiers.emc.com/linkrel/cabinets",
20   "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets"
21 },

```

HACKATHON - DOCUMENTUM REST API

Make sure basic authentication is still enabled, and press Send:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets	Params	Send ▾
-------	---	--------	--------

You will see a list of cabinets.

```
"entries": [
  {
    "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105",
    "title": "Administrator",
    "author": [
      {
        "name": "REPO_ADMIN",
        "uri": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/users/REPO_ADMIN"
      }
    ],
    "summary": "dm_cabinet 0c00000580000105",
    "updated": "2016-10-24T07:00:53.000+00:00",
    "published": "2015-03-03T10:15:33.000+00:00",
    "links": [
      {
        "rel": "edit",
        "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105"
      }
    ],
    "content": {
      "type": "application/vnd.emc.documentum+json",
      "src": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105"
    }
  },
  ...
]
```

Click on the **id** property of the first entry:

```
"entries": [
  {
    "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105",
    "title": "Administrator",
    ...
  },
  ...
]
```

Now hit Send:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105	Params	Send ▾
-------	--	--------	--------

The response contains the representation of a cabinet. The links in the response correspond to the transitions for a cabinet in the API Reference. Here is a subset of these links:

```

75  "links": [
76    {
77      "rel": "self",
78      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105"
79    },
80    {
81      "rel": "edit",
82      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105"
83    },
84    {
85      "rel": "http://identifiers.emc.com/linkrel/delete",
86      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105"
87    },
88    {
89      "rel": "http://identifiers.emc.com/linkrel/folders",
90      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0c00000580000105/folders"
91    }
92  ],

```

Get the Folders List for the Cabinet, Pick a Folder

In the links for the Cabinet, look for the link relation <http://identifiers.emc.com/linkrel/folders>, which identifies the list of folders for this cabinet, click on the value of the associated href property, and press Send:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0c00000580000105/folders	Params	Send ▾
-------	---	--------	--------

Each entry in the folders list corresponds to a folder:

```

18  "entries": [
19    {
20      "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1",
21      "title": "Copy Of Copy Of Workspace Customizations",
22      "author": [
23        {
24          "name": "dmadmin",
25          "uri": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/users/dmadmin"
26        }
27      ],
28      "summary": "dm_folder 0b000005800048b1",
29      "updated": "2016-10-24T08:22:59.000+00:00",
30      "published": "2016-10-24T08:22:59.000+00:00",
31      "links": [
32        {
33          "rel": "edit",
34          "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1"
35        }
36      ],
37      "content": {
38        "type": "application/vnd.emc.documentum+json",
39        "src": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1"
40      }
41    },

```

Choose the first folder in this list:

HACKATHON - DOCUMENTUM REST API

```
18  "entries": [  
19    {  
20      "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1",  
21      "title": "Copy Of Copy Of Workspace Customizations",
```

Now hit Send to retrieve the folder:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1	Params	Send ▾
-------	---	--------	--------

The folder contains links for the transitions associated with a folder:

```
72  "links": [  
73    {  
74      "rel": "self",  
75      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1"  
76    },  
77    {  
78      "rel": "edit",  
79      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1"  
80    },  
81    {  
82      "rel": "http://identifiers.emc.com/linkrel/delete",  
83      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1"  
84    },  
85    {  
86      "rel": "http://identifiers.emc.com/linkrel/parent-links",  
87      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/objects/0b000005800048b1/parent-links"  
88    },  
89    {  
90      "rel": "parent",  
91      "title": "0c00000580000105",  
92      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0c00000580000105"  
93    },  
94    {  
95      "rel": "http://identifiers.emc.com/linkrel/folders",  
96      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1/folders"  
97    },  
98    {  
99      "rel": "http://identifiers.emc.com/linkrel/documents",  
100     "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b000005800048b1/documents"
```

Get the Documents List for the Folder

In the links for the Folder, look for the link relation <http://identifiers.emc.com/linkrel/documents>, which identifies the list of documents for this folder, click on the value of the associated href property, and press Send:

```
98  {  
99    "rel": "http://identifiers.emc.com/linkrel/documents",  
100    "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents"  
101  },
```

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents	Params	Send ▾
-------	---	--------	--------

The result contains metadata about the collection, but the collection has no entries. In the lab, your version may or may not contain entries depending on what other students have placed in the folder. The following collection is empty; it contains metadata for the collection, but no documents:


```

1 {
2   "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents",
3   "title": "Documents under folder 0b0000058000022e",
4   "author": [
5     {
6       "name": "EMC Documentum"
7     }
8   ],
9   "updated": "2016-10-25T00:12:22.399+00:00",
10  "page": 1,
11  "items-per-page": 100,
12  "links": [
13    {
14      "rel": "self",
15      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents"
16    }
17  ]
18 }

```

We will need the documents list for the next section, so keep a copy of this URI where you can find it.

Creating, Reading, Updating, and Deleting Documents

In this section we will learn the basics of creating, reading, updating, and deleting documents. If you just completed the previous section, use the collection of documents for the folder from that section. If not, find a collection of documents by navigating to it.

Create a Document

In Documentum, a document resource contains both document metadata (such as author, creation date, etc.) and document content (the document that you would see in a word processor or printed out). The most basic document you can create requires a name and a type:

```

{
  "type": "dm_document",
  "properties" : { "object_name" : "Aardvark" }
}

```

(Actually, if you leave out the name a name will be created for you, so only type is strictly necessary, but you should usually provide a name so you can find the document again.)

In most REST APIs, you can create an item in a collection using a POST on that collection. We will create a new document by doing a POST to our documents collection:

POST ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents	Params	Send ▾
--------	---	--------	--------

But first we have to provide some information. First, click on the **Body** tab and input the name and type as raw text:

HACKATHON - DOCUMENTUM REST API

POST

http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents

Params

Send

Authorization Headers (2) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary Text

```
1 {
2   "type": "dm_document",
3   "properties" : { "object_name" : "Aardvark" }
4 }
```

Now we have to provide a content type so that the server knows what kind of data we are sending and can parse it appropriately. The content type is **application/vnd.emc.documentum+json**, so we will set the Content-Type header for the request accordingly:

Authorization Headers (2) Body Pre-request Script Tests

Content-Type

application/vnd.emc.documentum+json

Authorization

Basic ZG1hZG1pbjpwYXNzd29yZA==

Now it's time to push the Send button:

POST

http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents

Params

Send

The response contains the document that was created. It has a great deal more metadata than the data we specified – this metadata was created automatically by Documentum:

```
1 {
2   "name": "document",
3   "type": "dm_document",
4   "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_document",
5   "properties": {
6     "object_name": "",
7     "r_object_type": "dm_document",
8     "title": "",
9     "subject": "",
10    "authors": null,
11    "keywords": null,
12    "a_application_type": "",
13    "a_status": "",
14    "r_creation_date": "2016-10-25T19:55:15.000+00:00",
15    "r_modify_date": "2016-10-25T19:55:15.000+00:00",
16    "r_modifier": "dmadmin",
17    "r_access_date": null,
18    "a_is_hidden": false,
19    "i_is_deleted": false,
20    "a_retention_date": null,
21    "a_archive": false,
22    "a_compound_architecture": "",
23    "a_link_resolved": false,
24    "i_reference_cnt": 1,
25    "i_has_folder": true,
26    "i_folder_id": [
```

The response header contains the URI of the newly created resource:

Body	Cookies	Headers (11)	Tests
Cache-Control → no-cache, no-store, max-age=0, must-revalidate			
Content-Type → application/vnd.emc.documentum+json;charset=UTF-8			
Date → Tue, 25 Oct 2016 21:12:54 GMT			
Expires → 0			
Location → http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/documents/0900000580004919			

Save the tab with the POST request and open up a new tab using the + sign:

http://corest.lss.emc.c
X
+

POST
http://co

Copy the **Location** header into the address bar and hit Send to get the new resource:

GET
http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/documents/090000058000491a
Params
Send

Another way to see the newly created resource is to retrieve the collection that contains it. Go back to the tab that contains the POST. Go back to the tab with the POST request and change the POST to a GET using the dropdown control next to POST:

POST
http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents

GET

The entries in this collection will contain all the documents we created in it.

Update a Document

To modify a document, POST the properties you wish to change to the document's URI. For instance, we can GET the document we just created:

GET
http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/documents/090000058000491a
Params
Send

Press Send:

```

1 {
2   "name": "document",
3   "type": "dm_document",
4   "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_document",
5   "properties": {
6     "object_name": "Aardvark",
7     "r_object_type": "dm_document",
8     "r_creation_date": "2016-10-25T21:40:12.000+00:00",
9     "r_modify_date": "2016-10-25T21:40:12.000+00:00",

```

HACKATHON - DOCUMENTUM REST API

Let's change the HTTP method to POST:

The screenshot shows the Postman interface with the HTTP method dropdown menu open. The 'POST' option is highlighted with a red box. The URL is `http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/documents/090000058000491c`. The 'Headers (10)' and 'Tests' tabs are visible below the URL bar.

Now let's specify the new properties. We will change the object name from "Aardvark" to "Aardvark Bohemian":

The screenshot shows the 'Body' tab selected in Postman. The content type is set to 'Text'. The JSON body is as follows:

```
1 {  
2   "type": "dm_document",  
3   "properties": { "object_name": "Aardvark Bohemian" }  
4 }
```

Don't forget to set the content type:

The screenshot shows the 'Headers (2)' tab selected in Postman. A header is added with the key 'Content-Type' and the value 'application/vnd.emc.documentum+json'.

Now hit Send:

The screenshot shows the Postman interface with the 'POST' method and the same URL. The 'Send' button is highlighted in blue.

In the response, the object name has been updated:

The screenshot shows the response body in Postman, which is a JSON object:

```
1 {  
2   "name": "document",  
3   "type": "dm_document",  
4   "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_document",  
5   "properties": {  
6     "object_name": "Aardvark Bohemian",  
7     "r_object_type": "dm_document",
```

Create a Document with Multipart MIME

The documents we have created so far contain nothing but metadata, like an envelope with addresses but no contents. Now let's create a document that has both metadata and contents. We will do this two different ways. The first approach is to use Multipart MIME that we specify as text in Postman, posting it to the same URI we used to create documents earlier:

The screenshot shows the Postman interface with the 'POST' method and a new URL: `http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/folders/0b0000058000022e/documents`. The 'Send' button is highlighted in blue.

In the request headers, the content type indicates that this is Multipart MIME and specifies the boundary that separates metadata from content:

Authorization

Headers (2)

Body

Pre-request Script

Tests

✓ Authorization

Basic ZG1hZG1pbjpwYXNzd29yZA==

✓ Content-Type

multipart/form-data; boundary=vFKDeJ2jOu0ZQqKu9BGfHj9otiMjQ44EwXfIV

Each section of the text starts with a boundary, then identifies the content disposition and the content type, followed by the data itself. The boundary is an arbitrary string that is unlikely to occur in content. The *Content-Type* specifies both the MIME type `multipart/form-data` and the string that is used as a boundary. In this example, the boundary is:

vFKDeJ2j0u0ZQqKu9BGfHj9otiMJq44EwXfiV

Make sure that this boundary is used in the Content-Type header. Use the following text:

```
--vFKDeJ2j0u0ZQqKu9BGfHj9otiMJq44EwXfiV
Content-Disposition: form-data; name="metadata"
Content-Type: application/vnd.emc.documentum+json; charset=UTF-8
{"type": "dm_document", "properties": {"title": "Document
Time", "object_name": "testdocument.txt"}}

--vFKDeJ2j0u0ZQqKu9BGfHj9otiMJq44EwXfiV
Content-Disposition: form-data; name="binary"
Content-Type: plain/text; charset=UTF-8
sample text content

--vFKDeJ2j0u0ZQqKu9BGfHj9otiMJq44EwXfiV--
```

Copy it into the request body in Postman:

```

1 --vFKDeJ2j0u0ZQqKu9BGfHj9otiMJq44EwXfiV
2 Content-Disposition: form-data; name="metadata"
3 Content-Type: application/vnd.emc.documentum+json;charset=UTF-8
4
5 {"type":"dm_document","properties":{"title": "Document Time","object_name":"testdocument.txt"}}
6 --vFKDeJ2j0u0ZQqKu9BGfHj9otiMJq44EwXfiV
7 Content-Disposition: form-data; name="binary"
8 Content-Type: plain/text; charset=UTF-8
9
10 sample text content
11 --vFKDeJ2j0u0ZQqKu9BGfHj9otiMJq44EwXfiV--

```

Now hit Send. The response body contains the document – the “envelope” – with metadata similar to what we have seen in other examples:

```

1 {
2   "name": "document",
3   "type": "dm_document",
4   "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_document",
5   "properties": {
6     "object_name": "testdocument.txt",
7     "r_object_type": "dm_document",
8     "title": "Document Time",
9     "subject": "",
10    "authors": null,
11    "keywords": null,
12    "a_application_type": "",
13    "a_status": "",
14    "r_creation_date": "2016-10-26T01:44:16.000+00:00",
15    "r_modify_date": "2016-10-26T01:44:16.000+00:00",
16    "r_modifier": "dmadmin",

```

The response headers contain the content type of the above document:

Body	Cookies	Headers (11)	Tests
Cache-Control → no-cache, no-store, max-age=0, must-revalidate			
Content-Type → application/vnd.emc.documentum+json;charset=UTF-8			
Date → Wed, 26 Oct 2016 01:44:10 GMT			
Expires → 0			
Location → http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/documents/090000058000491d			

So what happened to the document content? We will see that in the next section, but first let's discuss a way to upload files directly from the file system using Postman, which is often convenient.

Get Document Content

The document we created in the last section has two link relations that point to the document's content:

HACKATHON - DOCUMENTUM REST API

```
125 {
126   "rel": "contents",
127   "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/objects/090000058000491d/contents"
128 },
129 {
130   "rel": "http://identifiers.emc.com/linkrel/primary-content",
131   "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/objects/090000058000491d/contents/content"
132 },
```

The contents link relation allows a single document to have multiple enclosures, each containing document content. The primary-content link relation refers to the primary content of the document. Let's choose the href associated with primary-content:

```
129 {
130   "rel": "http://identifiers.emc.com/linkrel/primary-content",
131   "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/objects/090000058000491d/contents/content"
132 },
```

Now let's get the associated resource:

GET Params

The response contains metadata that describes the enclosure. Let's look for the enclosure link relation, which contains a link to the enclosure itself. Click on the value of the href attribute:


```
30 {
31   "rel": "enclosure",
32   "title": "ACS",
33   "href": "http://10.37.14.190:9080/ACS/servlet/ACS?command=read&version=2.3&docbaseid=000005&basepath
          =R%3A%5CDocumentum%5Cdata%5CREPO%5Ccontent_storage_01%5C00000005&filepath=80%5C00%5C07%5C80&objectid
          =090000058000491d&cacheid=dAAEAgA%3D%3DgAcAgA%3D%3D&format=unknown&pagenum=0&signature
          =FBS3eJGhqFLE9rHR7U89IB77m%2FRbTXine4807nw8vwtJA6ALytuVsT9qC5ajDYIJ6HW%2BoDHLyuVnMjfmtMeCfGZeCJZaw%2FEDj
          m%2BXLruYDyQazHgG3IxnDfeR5iXFWP1REQpLCpuv9WmIHjMdyASBrzvaHXzTW5ZYL0%3D&servername=RESTCS72GAACS1&mode
          =1&timestamp=1477450430&length=19&parallel_streaming=true&expire_delta=360"
34 },
```

Now get the content of the enclosure:

GET Params

The body of the response contains the original document:

Body Cookies Headers (9) Tests

Pretty Raw Preview Text 

```
1 sample text content
```

Delete a Document

Documents are deleted using the HTTP DELETE method. Let's use that now to delete the last document we created. First, let's get the document to make sure we have the right URI:

GET Params

HACKATHON - DOCUMENTUM REST API

If the response corresponds to the document you want to delete, change GET to DELETE:

The screenshot shows a REST client interface. The top bar displays the method 'GET' with a dropdown arrow and the URL 'http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/documents/090000058000491d'. Below this, a menu on the left lists HTTP methods: GET, POST, PUT, PATCH, and DELETE. The 'DELETE' method is highlighted with a red box. To the right of the menu, there are tabs for 'Headers (1)', 'Body', 'Pre-request Script', and 'Tests'. Below these tabs, there is a 'Basic Auth' dropdown menu.

Now hit Send.

There is no message body in the result. Check the status code. A 204 indicates a successful delete:

The screenshot shows the response of the DELETE request. The top bar displays the status 'Status: 204 No Content' and the time 'Time: 1389 ms'. Below this, there are tabs for 'Body', 'Cookies', 'Headers (8)', and 'Tests'. The 'Body' tab is selected, and the response body is empty.

Let's verify that by changing the HTTP method back to GET and trying to read the resource we just deleted:

The screenshot shows the REST client interface with the method 'GET' selected. The URL is 'http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/documents/090000058000491d'. There is a 'Params' tab and a 'Send' button.

Both the response status and the response message body clearly tell us the resource was successfully deleted and can no longer be retrieved:

The screenshot shows the response of the GET request. The top bar displays the status 'Status: 404 Not Found' and the time 'Time: 851 ms'. Below this, there are tabs for 'Body', 'Cookies', 'Headers (10)', and 'Tests'. The 'Body' tab is selected, and the response body is displayed in a JSON format. The JSON object contains the following fields: 'status' (404), 'code' ('E_RESOURCE_NOT_FOUND'), 'message' ('The operation failed because the resource specified by ID, name, type, or path cannot be found.'), 'details' ('(DM_API_E_EXIST) [DM_API_E_EXIST]error: \\'Document/object specified by 090000058000491d does not exist.\\';(DM_SYSOBJECT_E_CANT_FETCH_INVALID_ID) [DM_SYSOBJECT_E_CANT_FETCH_INVALID_ID]error: \\'Cannot fetch a sysobject Invalid object ID : 090000058000491d\\''), and 'id' ('df7e073d-c324-49e3-92f9-ea79c301e4dc').

Now let's try to get the document's primary content, which was stored as a separate resource:

The screenshot shows the REST client interface with the method 'GET' selected. The URL is 'http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/objects/090000058000491d/contents/c'. There is a 'Params' tab and a 'Send' button.

It has also been deleted, as we can see in the response:


```

Body    Cookies    Headers (10)    Tests
Pretty  Raw    Preview    JSON
1 {
2   "status": 404,
3   "code": "E_CONTENT_NOT_FOUND",
4   "message": "The specified content of the object cannot be found.",
5   "details": "(DM_API_E_NO_MATCH) [DM_API_E_NO_MATCH]error: \"There was no match in the docbase for the qualification: 090000058000491d\"",
6   "id": "edcca084-9c41-4ae1-9fc8-f9b3accf5af7"
7 }

```

Collections

We have seen collections of repositories, collections of cabinets, collections of folders, and collections of documents. Collections are common and important in REST APIs, and the Documentum REST API allows you to view or filter collections in a variety of ways using URI parameters such as these:

- Inlining: `?inline=true`
- Showing totals: `?include-total=true`
- Paging: `?items-per-page=2&page=2`
- Sorting: `?sort=object_name`
- Filters: `?filter=contains(object_name, 'Backyard')`
- Views: `?inline=true&view=object_name,r_object_type`

In this lab we will explore these parameters by using them with the list of cabinets for a repository. First, let's look at the list of cabinets without any URI parameters (this URI will be different on your machine):

GET

http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets

Params

Send

Hit Send and you will see a result like this:

```

1 {
2   "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets",
3   "title": "Cabinets",
4   "author": [
5     {
6       "name": "EMC Documentum"
7     }
8   ],
9   "updated": "2016-10-25T01:01:30.133+00:00",
10  "page": 1,
11  "items-per-page": 100,
12  "links": [
13    {
14      "rel": "self",
15      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets"
16    }
17  ],
18  "entries": [
19    {
20      "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105",
21      "title": "Administrator",
22      "author": [
23        {
24          "name": "REPO_ADMIN",
25          "uri": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/users/REPO_ADMIN"
26        }
27      ]
28    }
29  ]
30 }

```

Inlining Collections

The items in a collection can be retrieved inline. This can simplify code and reduce the number of REST requests needed in an application. The URI parameter `inline` takes the values `true` or `false`. Let's set it to `true`:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?inline=true	Params	Send ▾
-------	---	--------	--------

Hit Send. Now the collection's representation contains the properties of each item inlined in the entry for that item:

```

18 ▾ "entries": [
19 ▾ {
20   "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105",
21   "title": "Administrator",
22   "author": [
23   ▾ {
24     "name": "REPO_ADMIN",
25     "uri": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/users/REPO_ADMIN"
26   }
27 ],
28   "summary": "dm_cabinet 0c00000580000105",
29   "updated": "2015-03-03T10:15:33.000+00:00",
30   "published": "2015-03-03T10:15:33.000+00:00",
31   "links": [
32   ▾ {
33     "rel": "edit",
34     "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets/0c00000580000105"
35   }
36 ],
37   "content": {
38     "name": "cabinet",
39     "type": "dm_cabinet",
40     "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_cabinet",
41     "properties": {
42       "r_object_id": "0c00000580000105",
43       "object_name": "Administrator",
44       "title": "Super User Cabinet",
45       "owner_name": "REPO_ADMIN",
46       "owner_permit": 7,
47       "group_name": "docu",
48       "group_permit": 5,

```

Showing Totals

Because computing the total number of items in a collection adds overhead, the Documentum REST API omits totals by default. You can ask for totals using the URI parameter `?include-total=true`:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?include-total=true	Params	Send ▾
-------	--	--------	--------

Now the total number of items in the collection is returned:

```

1  {
2    "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets",
3    "title": "Cabinets",
4    "author": [
5      {
6        "name": "EMC Documentum"
7      }
8    ],
9    "updated": "2016-10-25T01:16:39.086+00:00",
10   "page": 1,
11   "items-per-page": 100,
12   "total": 17,
13   "links": [
14     {
15       "rel": "self",
16       "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?include-total=true"
17     }
18   ],

```

Paging

Documentum collections can contain a very large number of items, so paging can be helpful. URI parameters can specify the number of items on a page or specify a specific page that should be returned. For instance, the parameters `&items-per-page=2&page=2` specify two items per page, and request the second page, which would have items 3-4:

GET Params

The metadata for the collection displays the current page and the number of items per page. As you can see, it corresponds to our request:

```

1  {
2    "id": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets",
3    "title": "Cabinets",
4    "author": [
5      {
6        "name": "EMC Documentum"
7      }
8    ],
9    "updated": "2016-10-25T01:20:41.305+00:00",
10   "page": 2,
11   "items-per-page": 2,
12   "total": 17,

```

In addition, the links contain link relations for paging to the previous, next, first, or last page in the collection:

```

13  "links": [
14    {
15      "rel": "self",
16      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?include-total=true&items-per-page=2&page=2"
17    },
18    {
19      "rel": "previous",
20      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?include-total=true&items-per-page=2&page=1"
21    },
22    {
23      "rel": "next",
24      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?include-total=true&items-per-page=2&page=3"
25    },
26    {
27      "rel": "first",
28      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?include-total=true&items-per-page=2&page=1"
29    },
30    {
31      "rel": "last",
32      "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?include-total=true&items-per-page=2&page=9"
33    }
34  ],

```

Sorting

Collections can be sorted using the `sort` URI parameter:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?sort=title	Params	Send ▾
-------	--	--------	--------

In the result, note that the items are sorted by title.

Filtering

Filters can be used to return only the items of a collection that satisfy one or more conditions. For instance, this specifies objects with names that contain "Cabinet": `?filter=contains(object_name, "Cabinet")`.

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?filter=contains(object_name, "Cabinet")	Params	Send ▾
-------	---	--------	--------

In the result, note that the items satisfy this condition.

Views

<http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?inline=true>

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?inline=true	Params	Send ▾
-------	---	--------	--------

```

37  "content": {
38    "name": "cabinet",
39    "type": "dm_cabinet",
40    "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_cabinet",
41    "properties": {
42      "r_object_id": "0c00000580000105",
43      "object_name": "Administrator",
44      "title": "Super User Cabinet",
45      "owner_name": "REPO_ADMIN",
46      "owner_permit": 7,
47      "group_name": "docu",
48      "group_permit": 5,
49      "world_permit": 3,
50      "acl_domain": "REPO_ADMIN",
51      "acl_name": "dm_4500000580000d0a",
52      "r_object_type": "dm_cabinet",
53      "r_creation_date": "2015-03-03T10:15:33.000+00:00",
54      "r_modify_date": "2015-03-03T10:15:33.000+00:00",
55      "i_antecedent_id": "0000000000000000",
56      "i_chronicle_id": "0c00000580000105",
57      "i_cabinet_id": "0c00000580000105"
58    },

```

GET

http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/cabinets?inline=true&view=object_name,r_object_type

Params

Send

```

37  "content": {
38    "name": "cabinet",
39    "type": "dm_cabinet",
40    "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_cabinet",
41    "properties": {
42      "object_name": "Administrator",
43      "r_object_type": "dm_cabinet"
44    },

```

Queries and Full-text Search

A repository contains links with URI templates for queries and full text search. To see them, get a repository:

GET

http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO

Params

Send

These links contain link relations for queries specified using DQL or full-text search:

```

66  {
67    "rel": "http://identifiers.emc.com/linkrel/dql",
68    "hreftemplate": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/{?dql}"
69  },
70  {
71    "rel": "http://identifiers.emc.com/linkrel/search",
72    "hreftemplate": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/search{?collections,facet,include-total,inline,items-per-page,locations,object-type,pag,q,sort,timezone,view}"
73  },

```

The curly braces indicate content to be supplied by the client. For instance, a client might replace `{?dql}` `select * from dm_folder` to create the following URI:

http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO?dql=select * from dm_folder

Similarly, a client might replace `{?collections, facet, include-total, inline, items-per-page, locations, object-type, page, q, sort, timezone, view}` with `q=test` to create the following URI:

<http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/search?q=test>

We will use these queries in the following sections.

DQL Queries

DQL is a SQL-like query language for Documentum. The following URI contains a query that searches for folders in the repository:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO?dql=select * from dm_folder	Params	Send ▾
-------	---	--------	--------

Each item in the result has the type `dm_folder`:

```

38 ▾  "content": {
39     "json-root": "query-result",
40     "type": "dm_folder",
41     "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_folder",
42 ▾   "properties": {
43     "r_object_id": "0b00000580000108",
44     "object_name": "Methods",
45     "title": "Methods Folder",
46     "subject": "Folder containing all dm_method objects.",

```

To find out how many folders there are, change the query to this: `select count(*) from dm_folder`:

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO?dql=select count(*) from dm_folder	Params	Send ▾
-------	---	--------	--------

Hit [Send] and look at the result:

```

24 ▾  "content": {
25     "json-root": "query-result",
26     "definition": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/types/dm_folder",
27 ▾   "properties": {
28     "dm_attr_0001": 566
29     }

```

At the time of this search, there were 566 folders in the repository.

Full-text Search

Documentum also supports full-text search using the `q` URI parameter. For instance, the following query looks for items containing the string "test":

GET ▾	http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/search?q=test	Params	Send ▾
-------	---	--------	--------

Hit [Send]. Look at the result. In the response body, each result is a document for which full-text search matched the string "test". The score for the match and the corresponding terms are returned in the entry's metadata:

```
22  "entries": [  
23  {  
24    "id": "0c000005800047ee",  
25    "title": "rest_my_test_cabinet",  
26    "author": [  
27      {  
28        "name": "dmdadmin"  
29      }  
30    ],  
31    "updated": "2016-02-16T03:47:42.000+00:00",  
32    "links": [  
33      {  
34        "rel": "edit",  
35        "href": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/objects/0c000005800047ee"  
36      }  
37    ],  
38    "content": {  
39      "type": "application/vnd.emc.documentum+json",  
40      "src": "http://corest.lss.emc.com:8080/dctm-rest/repositories/REPO/objects/0c000005800047ee"  
41    },  
42    "score": "1.0",  
43    "terms": [  
44      "test"  
45    ]  
46  },  
47  ],  
48  ],  
49  }
```