

Tutorial: Full-text Search with Documentum REST Services 7.2

This document introduces full-text search in *Documentum REST Services 7.2*, which is a new feature of release 7.2.

Part I Full-text Search

The “search engine” of *Documentum REST Services*

From release 7.2, *Documentum REST Services* supports full-text search. Not only meta data but also content are searchable with this service.

You can discover the search resource under repository resource.

```
http://demo-server:8080/dctm-rest/repositories/acme01
```

The search resource under repository acme01 looks like below. The link for search resource is a template containing many URL parameters.

```
{
```

```
"rel": "http://identifiers.emc.com/linkrel/search",
```

```
"hreftemplate": "http://demo-server:8080/dctm-rest/repositories/acme01/search{?collections,facet,include"
```

```
}
```

What to Search in the Tutorial

Assuming there is a brand new repository, some data and content are populated for this tutorial.

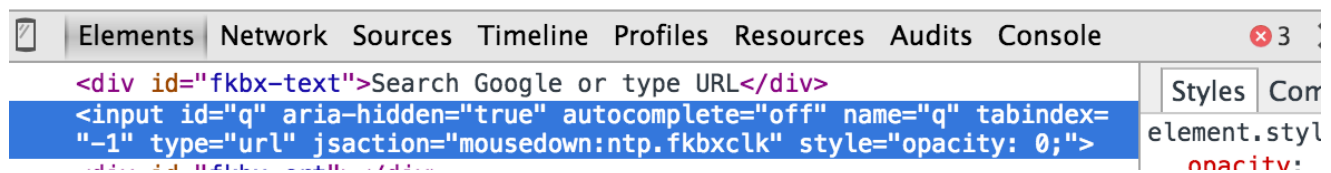
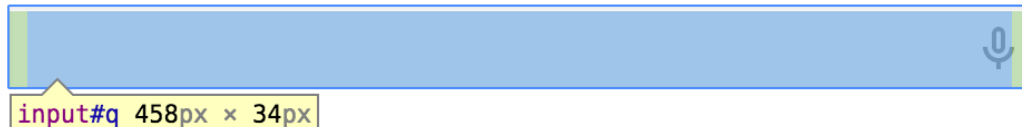
1. New cabinet "tutorial" is created under repository "acme01"
2. Document "documentum" is created under cabinet "tutorial" and it has content "documentum tutorial"
3. Folders "dfc" and "rest" are created under cabinet "tutorial"
4. Documents "dfc tutorial" and "rest tutorial" are created under folders "dfc" and "rest" respectively.

It looks like the structure in the figure below.

```
+++++
+ - acme01
+   |__cabinet-1
+   |__cabinet-2
+   |__...
+   |__tutorial
+   |__documentum (it has content "documentum tutorial")
+   |__dfc
+   |__dfc tutorial
+   |__rest
+   |__rest tutorial
+   |__cabinet-n
+   |__...
+   |__...
+   |__...
+++++
```

Enter a Query

When you look at web page of search engine like google or bing, you can't help to input the something in the search box. If you view the source of the page, you will probably find the search box has "q" as its name or id.



In the google page above, you can see the magic character “q” standing for “query”.

Search Resource of Documentum REST Services also has this “q” - a URL parameter.

First let’s search the key word “tutorial”.

<http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial>

The GET request to this URL will return the search results containing “tutorial”. (4 results)

In fact, we can leverage simple search language to define more complicated search criteria. (1 result for the request below)

<http://demo-server:8080/dctm-rest/repositories/acme01/search?q=rest tutorial>

Simple search language has implicit boolean operator “AND”. The URL above will search objects containing both “rest” and “tutorial”.

What if you want to search the phrase “rest tutorial”? Use double quotes. (1 result)

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q="rest tutorial"
```

Sometimes we have specific conditions for the objects to search. For example, we only care about the objects in our own folders, or some specified kinds of documents.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&locations=/tutorial/rest&object-type=dm_document
```

Another two URL parameters “locations” and “object-type” are for such purpose. In the request above, it will only search key word “tutorial” of type “dm_document” in folder “/tutorial/rest”. (1 result)

Understanding Results

Defining search criteria in URL is quite straightforward, just like using a popular search engine.

The search results are a bit sophisticated. There are two kinds of representation of search results: XML and JSON.

Search Results in XML

As the search results is a collection, its representation is an Atom feed, which is same as other collection type resources of *Documentum REST Services*, except there are two special elements in each Atom entry.

```

<feed
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>http://localhost:8080/dctm-rest/repositories/acme01/search</id>
  <author>
    <name>EMC Documentum</name>
  </author>
  <updated>2015-02-05T03:23:50.609+00:00</updated>
  <dm:page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">1
  </dm:page>
  <dm:items-per-page
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">100
  </dm:items-per-page>
  <link rel="self" href="http://localhost:8080/dctm-rest/repositories/acme01/search?q="
  <link rel="search" hreftemplate="http://localhost:8080/dctm-rest/repositories/acme01/search?q="
  <entry>
    <id>090004d2800265e4</id>
    <title>dfc tutorial</title>
    <author>
      <name>dmdadmin</name>
    </author>
    <updated>2015-02-05T01:50:40.000+00:00</updated>
    <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/acme01/object/090004d2800265e4"
    <content type="application/vnd.emc.documentum+xml" src="http://localhost:8080/dctm-rest/repositories/acme01/object/090004d2800265e4"
    <relevance:score
      xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/">1.0
    </relevance:score>
    <dm:terms
      xmlns:dm="http://identifiers.emc.com/vocab/documentum">
      <dm:term>tutorial</dm:term>
    </dm:terms>
  </entry>

```

Element score and terms

In the representation, an atom entry stands for a search result. And we can see the elements “score” and “terms” in each entry.

“score” is defined in open search specification. It means relative assessment of relevance for a particular search result with respect to the search query. “terms” is the matched terms of the search result.

Element content

The other element deserving attention is “content”. Element “content” indicates the details of real object contained in the search result entry.

By default, it has two attributes “type” and “src”. “type” is the content type of current representation; and “src” contains the link for the matched object.

But the URL parameter “inline” will populate details in element “content”.

<http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&inline=true>

```
<entry>
  <id>090004d2800265e4</id>
  <title>dfc tutorial</title>
  <author>
    <name>dmadmin</name>
  </author>
  <updated>2015-02-05T01:50:40.000+00:00</updated>
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/acme01/object/090004d2800265e4">
  <content>
    <dm:item
      xmlns:dm="http://identifiers.emc.com/vocab/documentum" xsi:type="dm_document">
      <dm:properties>
        <dm:r_object_id>090004d2800265e4</dm:r_object_id>
        <dm:object_name>dfc tutorial</dm:object_name>
        <dm:r_object_type>dm_document</dm:r_object_type>
        <dm:r_modify_date>2015-02-05T01:50:40.000+00:00</dm:r_modify_date>
        <dm:r_modifier>dmadmin</dm:r_modifier>
      </dm:properties>
      <dm:links>
        <link rel="self" href="http://localhost:8080/dctm-rest/repositories/acme01/object/090004d2800265e4">
      </dm:links>
    </dm:item>
  </content>
  <relevance:score
    xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0/">1.0
  </relevance:score>
  <dm:terms
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">
    <dm:term>tutorial</dm:term>
  </dm:terms>
</entry>
```

Search Results in JSON

The information in JSON representation search results is same as XML.

```

{
  "id": "http://localhost:8080/dctm-rest/repositories/acme01/search",
  "author": [
    {
      "name": "EMC Documentum"
    }
  ],
  "updated": "2015-02-05T03:24:48.222+00:00",
  "page": 1,
  "items-per-page": 100,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/dctm-rest/repositories/acme01/se
    },
    {
      "rel": "search",
      "hreftemplate": "http://localhost:8080/dctm-rest/repositories/a
    }
  ],
  "entries": [
    {
      "id": "090004d2800265e4",
      "title": "dfc tutorial",
      "author": [
        {
          "name": "dmadmin"
        }
      ],
      "updated": "2015-02-05T01:50:40.000+00:00",
      "links": [
        {
          "rel": "edit",
          "href": "http://localhost:8080/dctm-rest/repositories/a
        }
      ],
      "content": {
        "type": "application/json",
        "src": "http://localhost:8080/dctm-rest/repositories/acme01
      },
      "score": "1.0",
      "terms": [
        "tutorial"
      ]
    },
  ],
}

```


Pagination of Search Results

The size of search results for certain key word is probably big. It's necessary to divide the results into several blocks and control the size of each block. We have two URL parameters to do this.

- items-per-page, specifies the size of each page (block).
- page, specifies which page to return.

Now that we may get part of the whole results set, how to tell how many results in all? There is another pagination URL parameter.

- include-total, asks search service to retrieve the hit count.

With these new capabilities, a URL could look like below. (2 results in the page)

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&include-total=true&items-per-page=2&page=2
```

Other Customizations

The same results set may appear in different ways for different purposes.

Sort the Results

For example, we want the results show by lexical order of property "object_name". The URL below will do this.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&sort=object_name ASC
```

- sort, sorts the results for specified property.

Here “ASC” means ascending and “DESC” means descending.

Check the thumbnails

Sometimes, we prefer to check the thumbnail before requesting the whole object.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&thumbnail=true
```

- thumbnail, returns the thumbnail link for each entry in the collection resource.

This URL will ask for thumbnail link in each search result. The thumbnail link has relation “icon” for each entry.

```
<entry>
  <id>090004d2800051cb</id>
  <title>Tues_230PM_Galileo1001 EMC_Bueche.ppt</title>
  <author>
    <name>dmadmin</name>
  </author>
  <updated>2013-05-21T04:07:55.000+00:00</updated>
  <link rel="edit" href="http://localhost:8080/dctm-rest/reposito
  <link rel="icon" href="http://thumbnail-server:8081/thumbsrv/ge
  <content type="application/xml" src="http://localhost:8080/dctm
  <relevance:score
    xmlns:relevance="http://a9.com/-/opensearch/extensions/rele
  </relevance:score>
  <dm:terms
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">
    <dm:term>emc</dm:term>
    <dm:term>EMC</dm:term>
  </dm:terms>
</entry>
```

Customize Attributes in Results

Some users want to retrieve specified attributes in the search result entry instead of kicking off another request to get the object. For example, it could be helpful if the content size or version number are displayed in the search results.

- `view`, specifies the object properties to retrieve. This parameter works only when `inline` is set to `true`.

Below is a request which retrieving attributes “`r_version_label`” and “`r_content_size`”.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?  
q=tutorial&view=r_version_label,r_content_size&inline=true
```

The search result will contains the additional attributes in element “`content`”.

```

<entry>
  <id>090004d2800265e4</id>
  <title>dfc tutorial</title>
  <author>
    <name>dmdadmin</name>
  </author>
  <updated>2015-02-05T01:50:40.000+00:00</updated>
  <link rel="edit" href="http://localhost:8080/dctm-rest/repositories/acm" />
  <content>
    <dm:item
      xmlns:dm="http://identifiers.emc.com/vocab/documentum" xsi:type="dm:item">
      <dm:properties>
        <dm:r_version_label>
          <dm:item>1.0</dm:item>
          <dm:item>CURRENT</dm:item>
        </dm:r_version_label>
        <dm:r_content_size>0</dm:r_content_size>
      </dm:properties>
      <dm:links>
        <link rel="self" href="http://localhost:8080/dctm-rest/repositories/acm" />
      </dm:links>
    </dm:item>
  </content>
  <relevance:score
    xmlns:relevance="http://a9.com/-/opensearch/extensions/relevance/1.0" />
  </relevance:score>
  <dm:terms
    xmlns:dm="http://identifiers.emc.com/vocab/documentum">
    <dm:term>tutorial</dm:term>
  </dm:terms>
</entry>

```

Part II Facet

For search, especially enterprise content search, well-organized search results are important, like invoices during last month, vacation plans of somebody...

Facet search is an approach to classify information.

Facet Search (Navigation)

The first Request for Facet

From Part I, we know how to send a request for full-text search, on which facet navigation is built.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&facet=r_object_type
```

A new URL parameter is introduced in the request.

- facet, specifies the property to be used as facet.

In the request, the face of the cut and polished diamond is property “r_object_type”.

Remember the data in the repository acme01?

1. one cabinet named “tutorial”
2. one object named “documentum” with content “documentum tutorial”
3. two documents named “dfc tutorial” and “rest tutorial”

Exactly match the categories in the facet results.

```

</entry>
<dm:facet
  xmlns:dm="http://identifiers.emc.com/vocab/documentum">
  <dm:facet-id>facet_r_object_type</dm:facet-id>
  <dm:facet-label>Type</dm:facet-label>
  <dm:facet-value>
    <dm:facet-value-id>dm_document</dm:facet-value-id>
    <dm:facet-value-count>2</dm:facet-value-count> 2 documents
    <dm:facet-id>facet_r_object_type</dm:facet-id>
    <dm:facet-value-constraint>dm_document</dm:facet-value-constraint>
    <link rel="search" href="http://demo-server:8080/dctm-rest/
      repositories/acme01/search?q=tutorial&facet=r_object_type&
      facet-value-constraints=dm_document"/>
  </dm:facet-value>
  <dm:facet-value>
    <dm:facet-value-id>dm_cabinet</dm:facet-value-id>
    <dm:facet-value-count>1</dm:facet-value-count> 1 cabinet
    <dm:facet-id>facet_r_object_type</dm:facet-id>
    <dm:facet-value-constraint>dm_cabinet</dm:facet-value-constraint>
    <link rel="search" href="http://demo-server:8080/dctm-rest/
      repositories/acme01/search?q=tutorial&facet=r_object_type&
      facet-value-constraints=dm_cabinet"/>
  </dm:facet-value>
  <dm:facet-value>
    <dm:facet-value-id>dm_sysobject</dm:facet-value-id>
    <dm:facet-value-count>1</dm:facet-value-count> 1 object
    <dm:facet-id>facet_r_object_type</dm:facet-id>
    <dm:facet-value-constraint>dm_sysobject</dm:facet-value-constraint>
    <link rel="search" href="http://demo-server:8080/dctm-rest/
      repositories/acme01/search?q=tutorial&facet=r_object_type&
      facet-value-constraints=dm_sysobject"/>
  </dm:facet-value>
</dm:facet>
</feed>

```

Dig the Facet Results and Navigate

The facet results are easy-understanding. In each facet, there are several elements of facet-value, which represents one classification against the faceted property. In the sample, there are three categories for object type: dm_sysobject, dm_document and dm_folder.

The facet-value element contains information like id, count and constraint. The most important element in the facet-value is the link, which is the entrance for each category.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&facet=r_object_type&facet-value-constraints=dm_document
```

Clicking the link above will navigate us to the `dm_document` category. The combination of “`facet=r_object_type`” and “`facet-value-constraints=dm_document`” can be explained as `r_object_type` is `dm_document`.

- `facet-value-constraints`, specifies a property constraint expression.

Customize facet Search

Since `facet-value-constraints` can define the conditions for a facet classification, we may leverage it to customize our own facet category.

For example, we just want to get the search results of either `dm_document` or `dm_folder`. Unfortunately, search resource of *Documentum REST Services* doesn't return such category.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&facet=r_object_type&facet-value-constraints=dm_document|dm_folder
```

In the request, we customize the value of parameter `facet-value-constraints`: `dm_document|dm_folder`. This is a boolean expression, which means `dm_document` or `dm_folder`. Three results (1 folder and 2 documents) will be returned.

- `|`, OR
- `+`, AND

For the request below, only one result will be returned, the cabinet tutorial, which is of both `dm_sysobject` and `dm_folder`.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&facet=r_object_type&facet-value-constraints=dm_sysobject+dm_folder
```

How many results will the below URL return? Three (2 documents and 1 folder).
dm_docuemnt|dm_sysobject+dm_folder here is actually dm_docuemnt|(dm_sysobject+dm_folder). As dm_sysobject is parent of dm_folder, the expression equals to dm_document|dm_folder.

```
http://demo-server:8080/dctm-rest/repositories/acme01/search?q=tutorial&facet=r_object_type&facet-value-constraints=dm_docuemnt|dm_sysobject+dm_folder
```

The two operators has same precedence, but the expression is calculated from right to left.

- $A+B|C$ means $A+(B|C)$
- $A|B+C$ means $A|(B+C)$

Summary

Class is over. It's time to review what's covered in this tutorial.

Firstly, we learn how to define a search criteria. The URL parameters introduced in this part includes "q", "locations" and "object-type", which will decide the results set.

Once the results are given, we need to understand them. It's the precondition if we want to build an application based on search resource of *Documentum REST Services*. Obviously, the pagination options("items-per-page", "page" and "include-total") work for search results set as it is actually a collection.

Take one step further. Sometimes it's helpful if the results show up in a specific way. The URL parameters "sort", "inline", "view" and "thumbnail" can do this job. These options won't change the results set, but customize the way the results appear. This article is a 10 minutes tutorial.

For more details, refer the material *EMC® Documentum® Platform REST Services Version 7.2 Reference Guide*. simple search language how to get xml or json representation.

1. Don't forget to encode the character "|" and "+" in URL.