

Executing Batch operations in Documentum REST Services 7.2

Overview

The Batches and Batch Capabilities resources are provided since Documentum REST Services 7.2. Through the Batches resource, a client is capable of executing a bunch of operations in one http request with options, e.g., in a transaction or not.

The Batch Capabilities resource is used to tell the client what features are supported by the Batches resource. And what can or cannot be batched.

Two new links are added into the Repository resource:

- <http://identifiers.emc.com/linkrel/batches>
- <http://identifiers.emc.com/linkrel/batch-capabilities>

```
<repository xmlns="http://identifiers.emc.com/vocab/documentum" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>1</id>
  <name>REPO</name>
  <description/>
  <server>...</server>
  <links>
    <link rel="self" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO"/>
    <link rel="http://identifiers.emc.com/linkrel/users" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/users"/>
    <link rel="http://identifiers.emc.com/linkrel/current-user" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/currentuser"/>
    <link rel="http://identifiers.emc.com/linkrel/groups" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/groups"/>
    <link rel="http://identifiers.emc.com/linkrel/cabinets" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/cabinets"/>
    <link rel="http://identifiers.emc.com/linkrel/formats" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/formats"/>
    <link rel="http://identifiers.emc.com/linkrel/network-locations" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/network-locations"/>
    <link rel="http://identifiers.emc.com/linkrel/relations" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/relations"/>
    <link rel="http://identifiers.emc.com/linkrel/relation-types" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/relation-types"/>
    <link rel="http://identifiers.emc.com/linkrel/checked-out-objects" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/checked-out-objects"/>
    <link rel="http://identifiers.emc.com/linkrel/types" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/types"/>
    <link rel="http://identifiers.emc.com/linkrel/dql" hreftemplate="http://127.0.0.1:8080/dctm-rest/repositories/REPO/{?dql}"/>
    <link rel="http://identifiers.emc.com/linkrel/search" hreftemplate="http://127.0.0.1:8080/dctm-rest/repositories/REPO/search{?collections,facet,i
type,page,q,sort,timezone,view}"/>
    <link rel="http://identifiers.emc.com/linkrel/batches" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/batches"/>
    <link rel="http://identifiers.emc.com/linkrel/batch-capabilities" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/batch-capabilities"/>
  </links>
</repository>
```

Batch Capabilities Resource

The Batch Capabilities resource supports the GET operation. The response contains capabilities information about Batch.

```
<batch-capabilities xmlns="http://identifiers.emc.com/vocab/documentum" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <transactions>both</transactions>
  <sequence>both</sequence>
  <on-error>both</on-error>
  <batchable-resources>
    <batchable-resource>batch-capabilities</batchable-resource>
    <batchable-resource>cabinet</batchable-resource>
    <batchable-resource>cabinets</batchable-resource>
    <batchable-resource>checked-out-objects</batchable-resource>
    ...
    <batchable-resource>search</batchable-resource>
    <batchable-resource>type</batchable-resource>
    <batchable-resource>types</batchable-resource>
    <batchable-resource>user</batchable-resource>
    <batchable-resource>users</batchable-resource>
    <batchable-resource>versions</batchable-resource>
  </batchable-resources>
  <non-batchable-resources>
    <non-batchable-resource>batches</non-batchable-resource>
    <non-batchable-resource>content-media</non-batchable-resource>
  </non-batchable-resources>
  <links>
    <link rel="self" href="http://127.0.0.1:8080/dctm-rest/repositories/REPO/batch-capabilities"/>
  </links>
</batch-capabilities>
```

What batch features are supported

What resources are allowed to be put in a batch

What resources can not be batched

Batches Resource

The Batches resource only supports HTTP POST method. The server will execute the batch, and send back the response until the batch execution is done.

REST services provides the following properties for the client to specify how the server executes the batch:

- transactional

Since there can be many operations defined in one batch, the client can decide whether to put all the operations in a transaction, or execute them separately.

Valid values: true (default), false

- sequential

For a non transactional batch, the server will decide the execution sequence according this property. True means the server should execute the operations one by one with the submit sequence. Otherwise, the server may try to execute the operations in parallel despite the submit order. For a transactional batch, this value will be ignored. Valid values: true (default), false

- on-error

This property is applicable only for a sequentially executed and non transactional batch. In this case, if the server fails on one operation, the server has two options to finish the batch: fail immediately, or continue on for unfinished operations.

For a transactional batch, a failed operation will rollback the whole batch. For a non sequential and non transactional batch, since the operations are not executed sequentially, the on-error parameter is ignored.

Valid values: FAIL (default), CONTINUE

- return-request

Whether return the request together with the response.

Valid values: true, false (default)

The batch representation contains all the information for a batch request. Besides the parameters described above, the information of each operation also need be provided, e.g., the operation URI, the http method, or even the operation request body.

```
<?xml version="1.0" encoding="UTF-8"?>
<batch xmlns="http://identifiers.emc.com/vocab/documentum">
  <description></description>
  <transactional></transactional>
  <sequential></sequential>
  <on-error></on-error>
  <return-request></return-request>
  <operations>
    <operation id="">
      <description></description>
      <request method="" uri="">
        <header name="" value=""/>
        <entity></entity>
        <attachment>
          <xop:Include xmlns:xop=http://www.w3.org/2004/08/xop/include href=""/>
        </attachment>
      </request>
    </operation>
  </operations>
```

```
{
  "description" : "",
  "transactional" : true,
  "sequential" : true,
  "on-error" : "",
  "return-request" : false,
  "operations" : [ {
    "id" : "",
    "description" : "",
    "request" : {
      "method" : "",
      "uri" : "",
      "headers" : [ {
        "name" : "",
        "value" : ""
      } ],
      "entity" : "",
      "attachment" : {
        "Include" : {
          "href" : ""
        }
      }
    }
  } ]
}
```

1. Each 'operation' must have an unique id within the whole batch.
2. The 'operation' must have a 'request' defined, and the 'request' must have 'method' and 'uri'.
3. If the 'request' has specific headers, they can be defined in the 'header'.
4. The content in 'entity' is the request body of this 'request'.
5. The 'attachment' defines the attachment content id for the multipart batch, which can embed the binary content in the single batch request.

After execution, the server will return the batch representation for the response. The response batch representation returns the request information, as well as the result of the batch operations.

The following is a sample batch request and response in XML:

- Request

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <batch xmlns="http://identifiers.emc.com/vocab/documentum">
3   <description>a sample batch</description>
4   <transactional>false</transactional>
5   <sequential>true</sequential>
6   <on-error>continue</on-error>
7   <return-request>true</return-request>
8   <operations>
9     <operation id="op1">
10       <description>create a document operation</description>
11       <request method="POST" uri="/repositories/acme/folder/0c00000180000107/documents">
12         <header name="Content-Type" value="application/vnd.emc.documentum+xml" />
13         <header name="Accept" value="application/vnd.emc.documentum+xml" />
14         <entity>
15           <![CDATA[
16             <document type="dm_document">
17               <properties>
18                 <object_name>sample doc 1</object_name>
19                 <title>for sample</title>
20               </properties>
21             </document>
22           ]]>
23         </entity>
24       </request>
25     </operation>
26     <operation id="op2">
27       <request method="POST" uri="/repositories/REPO/folders/0c00208080000107/objects">
28         <header name="Content-Type" value="application/vnd.emc.documentum+xml" />
29         <header name="Accept" value="application/vnd.emc.documentum+xml" />
30         <entity>
31           &lt;object>&lt;properties>&lt;object_name>sample object 2&lt;/object_name>&lt;/properties>&lt;/object>&#xD;
32         </entity>
33       </request>
34     </operation>
35   </operations>
36 </batch>

```

Batch parameters

Operation must have the unique id

Request must has method and uri

Any specific header can be defined for each request

Entity contains the request body. It can be put in the cdata or escaped by xml. Or double quote escaped for json.

- Response

Executing Batch operations in Documentum REST Services 7.2

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <batch xmlns="http://identifiers.emc.com/vocab/documentum">
3   <description>a sample batch</description>
4   <transactional>false</transactional>
5   <sequential>true</sequential>
6   <on-error>continue</on-error>
7   <return-request>true</return-request>
8   <state>FINISHED</state>
9   <submitted>2014-07-03T16:37:32.147+08:00</submitted>
10  <started>2014-07-03T16:37:32.147+08:00</started>
11  <finished>2014-07-03T16:37:32.246+08:00</finished>
12  <owner>dmdadmin</owner>
13  <operations>
14    <operation id="op1">
15      <description>create a document operation</description>
16      <state>FINISHED</state>
17      <started>2014-07-03T16:37:32.147+08:00</started>
18      <finished>2014-07-03T16:37:32.158+08:00</finished>
19      <request method="POST" uri="/repositories/acme/folder/0c00000180000107/objects">
20        ...
21      </request>
22      <response status="201">
23        <header name="Location" value="http://127.0.0.1:8080/dctm-rest/repositories/REPO/documents/0900000180015693.xml"/>
24        <header name="Content-Type" value="application/vnd.emc.documentum+xml; charset=UTF-8"/>
25        <entity><?xml version='1.0' encoding='UTF-8'?><document xmlns="http://identifiers.emc.com/vocab/documentum" xmlns:xsi="
26        ...
27      </response>
28    </operation>
29    <operation id="op2">
30      <state>FINISHED</state>
31      <started>2014-07-03T16:37:32.158+08:00</started>
32      <finished>2014-07-03T16:37:32.244+08:00</finished>
33      <request method="POST" uri="/repositories/REPO/folders/0c00208080000107/objects">
34        ...
35      </request>
36      <response status="201">
37        <header name="Location" value="http://127.0.0.1:8080/dctm-rest/repositories/REPO/documents/0900000180015694.xml"/>
38        <header name="Content-Type" value="application/vnd.emc.documentum+xml; charset=UTF-8"/>
39        <entity><?xml version='1.0' encoding='UTF-8'?><document xmlns="http://identifiers.emc.com/vocab/documentum" xmlns:xsi="
40        ...
41      </response>
42    </operation>
43  </operations>
44 </batch>
```

The batch execution info

The operation execution info

If set 'return-request' to true, then the request is returned as well

The operation response, including the status code, the response header and response body

If one or more operations need to send a binary content (for example, creating an object with content), the batch request must be sent as a multipart request. The request batch representation itself is in a part, and other binary contents are in the following parts with the same sequences as they defined in the batch request operations.

The relative batch request content-type are:

- Multipart/Related;boundary=boundary-line;type="application/xop+xml";start="batch";start-info="application/vnd.emc.documentum+xml"
- Multipart/Related;boundary=boundary-line;type="application/jop+json";start="batch";start-info="application/vnd.emc.documentum+json"

A sample multipart batch in JSON can be found below:

Executing Batch operations in Documentum REST Services 7.2

```
1 --the-part-boundary-line
2 Content-Type: application/jop+json; type="application/vnd.emc.documentum+json"
3 Content-ID: batch
4 Content-disposition: form-data; name=batch
5
6 {
7   "operations" : [ {
8     "id" : "id-100",
9     "request" : {
10      "method" : "POST", "uri" : "/repositories/REPO/folders/0c00208080000107/objects",
11      "headers" : [ { "name" : "Content-Type", "value" : "application/vnd.emc.documentum+json" },
12                    { "name" : "Accept", "value" : "application/vnd.emc.documentum+json" } ],
13      "entity" : "{\n\"properties\":{\n\"object name\":\n\"my test object\"}\n}",
14      "attachment" : { "Include" : { "href" : "cid:id-100-content" } } }
15   }, {
16     "id" : "id-101",
17     "request" : {
18      "method" : "POST", "uri" : "/repositories/REPO/folders/0c00208080000107/objects",
19      "headers" : [ { "name" : "Content-Type", "value" : "application/vnd.emc.documentum+json" },
20                    { "name" : "Accept", "value" : "application/vnd.emc.documentum+json" } ],
21      "entity" : "{\n\"properties\":{\n\"object name\":\n\"my test object\"}\n}",
22      "attachment" : { "Include" : { "href" : "cid:id-101-content" } } }
23   } ]
24 }
25 }
26 }
27
28 --the-part-boundary-line
29 Content-Type: text/plain
30 Content-ID: id-100-content
31 Content-disposition: form-data; name=id-100-content
32
33 i'm the content of id-100
34
35 --the-part-boundary-line
36 Content-Type: text/plain
37 Content-ID: id-101-content
38 Content-disposition: form-data; name=id-101-content
39
40 i'm the content of id-101
41
42 --the-part-boundary-line--
```

The batch part

Specify the content id, must be the same with following parts Content-ID header

The first content part, must have the same Content-ID as defined in the previous batch operation 'attachment'

The second part

To send a batch request to the Batches resource, you may reuse any client lib or tools or even existed code. The batch operation request has the same format as sending the request to that resource directly. You just need to put the single request headers in the "header" sections of each batch operation request, and put the whole single request body as a string value in the "entity" section. Vice versa, the operation response's headers are put in the "header" sections of each batch operation response, together with the response body in the "entity" as the string value.

However, for a multipart batch request, you have to compose the request body by yourself. For example, if you are using Spring RestTemplate as a client lib, you have to implement your own `HttpMessageConverter` to handle the request, because the media type "multipart/related" are not supported by default. Or you can use any technology, only if it follows both the multipart protocol, and the batch representation requirement.

Trouble Shooting

Executing Batch operations in Documentum REST Services 7.2

1. Check the Batches resource response for the error messages, and the server log. Or even turn on the trace log for more detailed information
2. Following the develop guide to change the batch configurations
3. Change the batch properties, e.g. change the non-transactional and non-sequential batch to transactional or sequential
4. For a multipart batch, trying to resend as a simple batch without the attachments
5. Reduce the batch operations count for a single batch

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