

# Learning Documentum REST Services from RADL Documentation

---

Starting from release 7.3, Documentum REST Services provides users with a new API documentation in HTML content format. In this article, I'll walk through this API documentation to help you get started easily. The contents of this article include:

- **Background**
- **Get the documentation**
- **Documentation structure**
- **State transitions**
- **Exploring a resource**
- **A case study**
- **Future work**

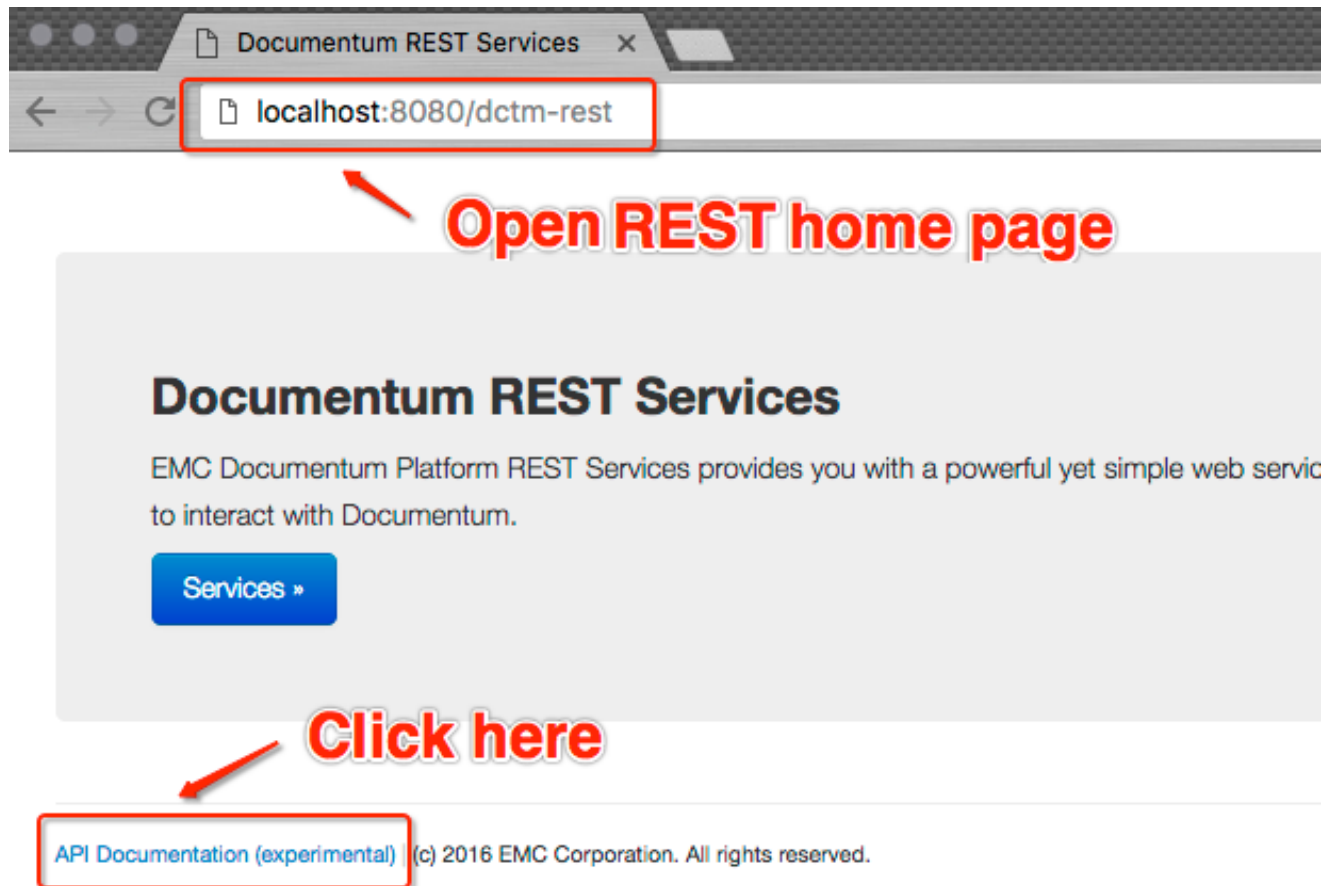
## Background

The first question that comes out to your head may be "what is RADL?". RADL stands for *RESTful API Description Language*. It is an open source specification and tooling to design hypermedia-driven RESTful API. Its source code is available on [GitHub - restful-api-description-language/RADL](#). So not surprisingly, the HTML documentation for Documentum REST Services you will see next is written in RADL. If you already have some experience in REST API development, you may have heard about [Swagger](#) or [Spring REST Docs](#), which are much more popular frameworks to generate REST API documentation. The biggest advantage of RADL is, it is the only REST API description language that describes how state transfer works in a REST API. You will see how we use RADL to describe Documentum REST state transfer if you are still with me.

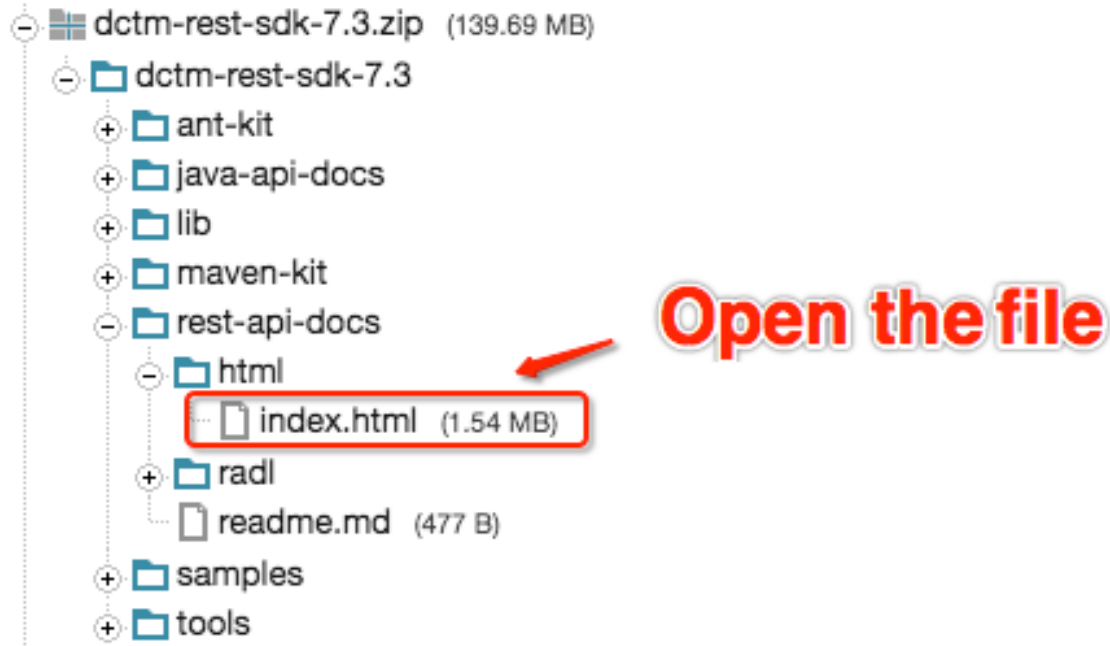
## Get the documentation

I assume that you already know where to down Documentum REST binaries. If not, here is the link to Documentum REST Services product resources on DELL EMC support site, [https://support.emc.com/products/32689\\_Documentum-REST-Services](https://support.emc.com/products/32689_Documentum-REST-Services) (require login).

## Get API documentation from a running REST server



## Get API documentation from the Documentum REST SDK



Please note that the API documentation is marked as *experimental*. The reason is we don't intend to immediately replace *Documentum REST Services Reference Guide* (pdf version) by this HTML documentation since the pdf version still has its unique values for customers.

## Documentation structure

Now if you open the HTML file, you will see the content structure from the left sidebar. The name of each section clearly tells what its content is about. I just give you a quick overview of outlines here.

## 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

- **States** - States refer to the letter **S** in the term **REST**. Resources in an application have their states, for instance, a locked document, a purged group, etc. A REST API essentially describes the state transitions of resources in an application.
- **Link Relations** - Link relations are the hypermedia markups on resource presentations to link two independent resources together with an explicit relationship, for instance, a locked document and its lock owner. It's the key factor to make a web API RESTful.
- **Property Groups** - Property groups are a set of properties to describe the data transfer objects in REST API request and response bodies, for instance, which properties are mandatory for a new folder creation. It does NOT define which representation format (XML or JSON) the payload should be in.
- **URI Parameters** - URI parameters are applied on the URIs of REST API calls as the query string. They provide additional options for the REST API calls, for instance, ordering a folder collection or not.
- **Media Types** - Media types define the format of resource representations in request and response bodies. JSON and XML are supported.

- **Custom Headers** - Custom headers are HTTP headers applied to request and responses of REST API calls. For instance, *Content-Type* and *Accept* headers are used to negotiate content media types.
- **Status Codes** - Status codes define the response result status in HTTP standard codes, for instance, 201 for created. It's the most straightforward way to identify whether a REST API call is successful or failed.
- **Resources** - Resources define all available URI-identified resources and their operations in the REST API. Resources are the wheels of the state transition engine in an application.
- **Authentication** - Authentication defines all supported authentication schemes on the REST API.

## State transitions

I am pretty confident that you know other parts well, except the states. So please give me some time to explain how states describe the REST API. The state definition is comprised of three elements: a **source state**, a **transition**, and a **result state**.

State	Transition	Result State
Start	Entry point	home-doc
home-doc	Get product information	product-info
home-doc	Get repositories	repositories

A source state assumes where the client currently stands in an application process. A transition describes what available business operation is done from the source state. A result state describes the result of this transition. If you have the chance to check the RADL XML definition from Documentum REST SDK, you'll find the state definition part like this:

```
<state name="home-doc">    <transitions>        <transition name="Get product information" to="product-info"/>        <transition name="Get product information" to="product-info"/>    </transitions></state>
```

**Start** is a dummy state in the API documentation, which only means to the starting point. **Home document** is the entry point for the entire Documentum REST Services. If you click on *Entry point* of the HTML document, you'll see how you get to home document in the REST API. It's done by performing a GET method on the billboard URI.

## 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)

The result of this transition is **home-doc**, which stands for home document. It has two media types for its representation, XML (*application/home+xml*) and the JSON (*application/home+json*). If you click their media type links, you'll see the XML and JSON samples for home document, respectively.

## Representation: home-doc

### Example:

```
<resources xmlns="http://identifiers.emc.com/vocab/documentum">
  <resource rel="http://identifiers.emc.com/linkrel/repositories">
    <link href="http://localhost:8080/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>
```

## Representation: home-doc

### Example:

```
{
  "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"
        ]
      }
    }
  }
}
```

If you click on the result state **home-doc**, you'll see what further state transitions can be operated on home document. For instance, the default REST API has two available

transitions on **home-doc**, which are **Get product information** and **Get repositories**, respectively. The transition **Get product information** is done by performing the GET method on the href of link relation **about** from state **home-doc**.

## State: home-doc

### Transitions

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

By now, you must have gotten the idea that with states and transitions. By state transitions, we can reach out to all available resources and their operations in Documentum REST Services. Therefore, you can take the the entire state transfer of Documentum REST Services as a bidirectional graph with a starting point. It's bidirectional because with link relations, resources can navigate to each other. Below is a tree view of state transitions in Documentum REST Services in a gif (*Click on the image to view its animation*).



*(The tree view code is out of the REST binaries so far; we will contribute it to RADL open source project in the future.)*

## Exploring a resource

By now, you have gotten the idea about state transfer in Documentum REST Services. Different with states, resources are the URI identified endpoints for the REST programming. They are the wheels to transit states. Next, let's explore a sample resource in this documentation to see what information it gives you for the REST API consumption. Let's look at a **document resource**.

document	The document resource models a single document in the repository.	<div>GET</div> <div>POST</div> <div>DELETE</div>
----------	---	--

*Note: You may find that resource URI for document resource is not documented on the API documentation. The reason is, as a RESTful API, we don't encourage clients to hard-code resource URIs except the billboard URI on home document resource. If clients hard code all resource URIs in client codes, the REST API loses its benefits for server driven state transfer, and clients become tightly bind to specific REST servers. But if developers want to reference resource URIs for prototyping or debugging, they can find resource URIs in RADL XML definition.*

```
<resource name="document" status="complete" public="true">    <documentation>The document resource models a single document
```

A document resource supports GET (read), POST (update) and DELETE (remove) methods. We only look at the GET method here.

## Method **GET**

Get the specified document.

### Transitions:

- Find document : [folder-child-documents](#) → [document](#)
- Get canonical document : [object](#) → [document](#)

### Request:

#### URI Parameters:

- [links](#)
- [view](#)

### Response:

#### Result States:

- [document](#)

#### Media Types:

- [application/vnd.emc.documentum+xml](#) ([document](#))
- [application/vnd.emc.documentum+json](#) ([document](#))

By definition, the **document** state can be reached out by performing a GET method on two source states, a **folder-child-documents** state, and an **object** (sysobject) state. Please note that almost every state can circle to itself by a "self" link relation and this is not explicitly documented in RADL.

On this GET method, there are two query parameters that can be applied, **links** and **view**. By clicking on their links, you'll get their definitions.

**links**

Datatype	Values	Default Value	Documentation
boolean	true, false	true	<p>Ensures whether link relations to be returned for this object representation.</p> <ul style="list-style-type: none"> <li><i>true</i> - return links for the object</li> <li><i>false</i> - do not return links for the object</li> </ul>

**view**

Datatype	Default Value	Documentation
string	:default	<p>Properties to return. The pattern is like: <code>?view=( :view-name )?( ,column )*</code></p> <ul style="list-style-type: none"> <li><i>( :view-name )</i> and <i>( ,column )*</i> must be mutually exclusive</li> <li>names of predefined views start with colon (':'). The following view-name are defined: <ul style="list-style-type: none"> <li><i>:all</i></li> <li><i>:default</i></li> </ul> </li> <li>if no view-name is specified, names of properties or predefined views should be returned, separated by comma (',').</li> </ul>

By definition, this GET request does not ask for a request body. By observing the response part, we can see that the result is a **document** state, which can be represented in both XML and JSON representations. If you click on their links, you'll reach out to their sample messages.

In summary, this section tells you that from resource perspective, we can learn what resource methods, request parameters, request body and response body are applicable on a resource.

## A case study

With all the above introduction, let us look at a case of using RADL documentation to help you programming with the REST API. **Move** is an operation that changes the parent folder of a document or a sysobject from one location to the other. There is an article [Tutorial: Copy, Move, Link and Unlink Objects in REST Services](#) telling how to move a document in Documentum REST Services. In this section, I will show you how to **Move a document** by RADL documentation instead.

The start point is the **States** table, because state transitions is exactly mapping to business operations. Here is the methodology:

- In the States table, search the transition keyword **Move**

parent-folder-link	Move SysObject	parent-folder-link
--------------------	----------------	--------------------

- Next, search in the States table about how to reach to state **parent-folder-link**

parent-folder-links	Find parent folder link	parent-folder-link
---------------------	-------------------------	--------------------

- Same to above, search the transition to **parent-folder-links**

document	Get parent links	parent-folder-links
----------	------------------	---------------------

By these research, we get to know that the state transition to move a document is by the following sequence:

**document** -- *Get parent links* --> **parent-folder-links** -- *Find parent folder link* --> **parent-folder-link** -- *Move sysobject* --> **parent-folder-link (other)**

Each transition definition tells you at least two key points to complete the transition:

1. What **link relation** to use on current state representation
2. What **HTTP method** to apply on the link relation

For example, on the transition **Get parent links**, the definition clearly tells the above points:

### Transition: Get parent links

By performing HTTP method **GET** on link relation <http://identifiers.emc.com/linkrel/parent-links> of state **document**.

Therefore, following the transitions step by step, you are able to move a document from one location to the other. The RADL documentation tells you all information required to complete a transition and I will not repeat them in the article.

I am stopping here. The next is for you to explore the REST API documentation after downloading the REST binaries. Hopefully, it gives you the better experience on learning Documentum REST Services.

## Future work

We are working on both RADL specification improvement and Documentum REST API documentation improvement. And we are looking forward to seeing your feedbacks on using the new Documentum REST Services documentation.

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