

# Documentum REST Extensibility Tutorial (2): Create A Sample Project

Version 14

created by William Zhou on Feb 28, 2015 2:27 AM, last modified by William Zhou on Sep 18, 2016 8:28 PM



*Previous:* [Documentum REST Extensibility Tutorial \(1\): Install Documentum REST Artifacts](#)

This document is for the continuous series for Documentum REST Extensibility tutorials. Previously, we talked about how to install Documentum REST artifacts into the Maven repository. In this document, we will talk about how to create your first sample project for custom REST development.

## Preliminary

Before diving into this tutorial, you must complete the software and Maven artifacts installation described by [Documentum REST Extensibility Tutorial \(1\): Install Documentum REST Artifacts](#).

## Creating your first sample REST project

As an example, we will create a REST extension project which contains the out of the box Documentum REST services and the extended [alias sets services](#). The sample project adds two sample resources for the alias set collection and the single alias set, respectively, to Documentum REST Services.

Resource	URI	HTTP Method	Mime Type
Alias Set Collection	/repositories/{repositoryName}/alias-sets{?view,filter,links,inline,page,items-per-page,include-total,sort}	GET	application/atom+xml application/vnd.emc.documentum+json
Alias Set	/repositories/{repositoryName}/alias-sets/{alias-set-id}{?view,filter,links}	GET	application/vnd.emc.documentum+xml application/vnd.emc.documentum+json

The first step is to install the Maven archetype for Documentum REST Services.

## Create the project using Documentum REST SDK

The Documentum REST SDK provides a [Maven archetype](#) project for you to create the first sample project for custom REST development. You need to install the Maven archetype project to the repository first.

### For REST SDK 7.2 Patch 03 and later versions

Since the version **7.2 Patch 03**, Documentum REST SDK provides users the single command-line script to create the REST project.

- (1) Enter to the SDK folder `<SDK_ROOT>/maven-kit/`
- (2a) For Windows users, run the batch script in console: `dctm-rest-getstarted.bat`
- (2b) For Linux/Mac users, run the bash script in terminal: `bash dctm-rest-getstarted.sh`

The above script will create and build a sample Documentum REST extension project under the directory `<SDK_ROOT>/maven-kit/generated/acme-rest/**`

## For REST SDK 7.2 GA, Patch 01 and Patch 02

For SDK users earlier than 7.2 Patch 03, it is **highly recommended** to upgrade to the latest patch version!

For SDK users of REST SDK 7.2 GA, Patch 01 and Patch 02, the steps can be divided into:

1. Install the Maven archetype
2. Create the sample project
3. Build the project

The SDK provides a readme file under `<SDK_ROOT>/maven-kit/archetype-install-guide.txt` to illustrate how to execute the steps manually.

## Deploy the WAR

With the previous step, the custom WAR file is built at the location `<SDK_ROOT>/maven-kit/generated/acme-rest/acme-rest-web/target/acme-web-1.0.0-SNAPSHOT.war`

Before deploying it into a web container, you will need to update `dfc.properties` under `<WAR>/WEB-INF/classes`. This is the only file need to modify for the deployment. For instance,

```
01. dfc.docbroker.host[0]= 192.168.1.100
02. dfc.docbroker.port[0]= 1489
03. dfc.globalregistry.repository= REPO
04. dfc.globalregistry.username= dave
05. dfc.globalregistry.password= password
```

It is not very convenient to update a file in an archived WAR file. The other option is to modify the root `pom.xml` and **rebuild** the war. The build will populate the DFC properties to the WAR file during the build. The root `pom.xml` is at `<SDK_ROOT>/maven-kit/generated/acme-rest/pom.xml`.

```
01. <properties>
02.     <!-- TODO: update below test properties for unit testing -->
03.     <test.docbroker.host>192.168.1.100</test.docbroker.host>
04.     <test.docbroker.port>1489</test.docbroker.port>
05.     <test.repository>REPO</test.repository>
06.     <test.username>dave</test.username>
07.     <test.password>password</test.password>
08. </properties>
```

Assuming that we change the WAR file name `acme-rest.war` to when deploying it to the web container.

## Run the test

The web container system requirement for the custom WAR file is same to the out-of-the-box Documentum REST Services. From the [repository resource](#), you will find a new link relation "<http://identifiers.emc.com/linkrel/alias>" its links collection. This is the new link relation we added by this project, which points to the new developed alias set collection resource.

```
01. GET /acme-rest/repositories/REPO.json HTTP/1.1
```



```
{
  id: 1,
  name: "REPO",
  description: "",
  - servers: [
    - {
      name: "REPO",
      host: "contentserver72",
      version: "7.2.0000.0106 Win64.SQLServer",
      docbroker: "contentserver72"
    }
  ],
  - links: [
    - {
      rel: "self",
      href: "http://localhost:8080/acme-rest/repositories/REPO.json"
    },
    - {
      rel: "http://identifiers.emc.com/linkrel/alias",
      href: "http://localhost:8080/acme-rest/repositories/REPO/alias-sets.json"
    },
    - {
      rel: "http://identifiers.emc.com/linkrel/users",
      href: "http://localhost:8080/acme-rest/repositories/REPO/users.json"
    }
  ],
}
```

You can then navigate down to the alias sets collection via this link relation.

01. | GET /acme-rest/repositories/REPO/alias-sets.json HTTP/1.1

In next post, we will walk through the code of this sample project.

*Next: Documentum REST Extensibility Tutorial (3): Explore The Sample Project*

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