3/17/2016

EMC²

.NET Rest Model/Controller Framework

Cross platform for Windows, Mac, Linux, Droid, iPhone, and Windows Phone

# Disclaimer

This project is released under Apache License 2.0, January 2004. The full license is contained within the project and can be viewed here:

<https://github.com/Enterprise-Content-Management/documentum-rest-client-dotnet/blob/master/LICENSE>

# Overview

Rest clients are different from DFC clients so while it may be ideal to replicate the same API syntax, it is not always practical. The intent is to make things similar enough that you can easily figure out the differences. The Tester project, included in this solution, is meant to evolve and show distinct use cases and how to code them to use them in your project. If you are wanting to submit your use case to GitHub, please include a use case in the flow for the Tester application. This application not only provides sample code, but provides end to end testing/load for new releases.

All use cases below can be found in the Tester project in the UseCaseTests class. There you can see how use cases flow from one to the next.

# Setting up the Rest Controller and getting a Repository object for Documentum Operations

## Controller Setup

The controller can be initialized in multiple ways to support multiple authentication mechanisms. The idea behind the framework is to make it as extensible as possible without locking anyone into any form of authentication. If other mechanisms not handle by the core framework are needed. Please submit a request and we will try to put it on the roadmap, or write it yourself and submit it for inclusion at GitHub.

Basic Auth with a class that implements the LoggerFacade for outputting console information and/or performance information to the log  
public RestController(string userName, string password, LoggerFacade logger)

Basic auth with username and password, uses default HttpTimeout of 5 minutes  
public RestController(string userName, string password)

Used to setup a RestController with a supplied http timeout parameter. This constructor will use the current process owner’s Kerberos credentials, ensure DCTM Rest is setup to handle Kerberos authentication.   
public RestController(int timeOutMinutes)

This method enables you to get an uninitialized controller.   
RestController()

//Example:

RestController client = new RestController();

HttpClient httpClient = {setup for the authentication you want};  
client.initClient(httpClient, timeout);

## Repository Setup

The repository object is a central base for all operations and is loosely synonymous with a DFC session.

RestController client = new {choose your RestContoller constructor above};  
RestService home = client.Get<RestService>(RestHomeUri, null);

home.SetClient(client);

ProductInfo productInfo = home.GetProductInfo();

Repository repository = home.GetRepository(repositoryName);

Now that you have a repository object, you can start developing your use cases.

## Object Types

To avoid constantly providing the folder and document object types for every create operation, the repository has methods to set the DocumentType and FolderType you want to use. This setting will be kept until you change it and you can change it before each operation you perform as needed.

// Set our default folder and document types.

repository.DocumentType = "dm\_document";

repository.FolderType = "dm\_folder";

If you develop use cases that will span using multiple types, you will need to do a repository.DocumentType or repository.FolderType set before your create operation like so:

// Set our default folder and document types.

repository.DocumentType = "my\_document\_type";

repository.FolderType = "my\_folder\_type ";

RestDocument is synonymous with dm\_sysobject and subtypes.   
Folder is synonymous with dm\_folder and subtypes  
Cabinet is synonymous with dm\_cabinet and subtypes

You can, of course, implement the framework and create your own class types that extend these to add your own functional methods to them but these should provide most all base functions needed.

### Fetching Object(s)

There are a few ways to use DQL to query for objects.   
Straight DQL:

Feed<RestDocument> results = repository.ExecuteDQL<RestDocument>(String.Format("select \* from dm\_document where FOLDER('/Templates') "), new FeedGetOptions { Inline = true, Links = true });

List<RestDocument> docs = ObjectUtil.getFeedAsList<RestDocument>(results, true);

int resultSamples = docs.Count; // To get the count

Fetch a Single Object using repository.getObjectById or repository.getObjectByQualification methods.

## Creating Folders

The RestModelController Repository object contains a method: repository.getOrCreateFolderByPath(“/Cabinet/path/……”) that allows for easy creation of full or partial folder paths. For Example, if you used: repository.getOrCreateFolderByPath(“/Temp/myfolder”), /Temp already exists so it would create myfolder if it did not exist. If the full folder path already exists, it would return the myfolder Folder object.

## Creating Documents

This sample is from the Tester project UseCaseTests class. It shows how a document is created and linked to a folder. You would provide the dctmPath and file then it would import the document in the dctmPath. Highlighted are the specific items needed to create an object in a folder. This is not an optimized use case, it is meant to show you creation, import, and how you can fetch the document to ensure you have the up to date version of the document.

public RestDocument CreateDocument(Repository repository, String dctmPath, FileInfo file)

Folder tempFolder = repository.getOrCreateFolderByPath(dctmPath);

RestDocument newDocument = repository.ImportNewDocument(file, testPrefix + "-" + file.Name, tempPath);

WriteOutput("\t\t[ImportDocument] - RestDocument " + file.FullName + " imported as "

+ newDocument.getAttributeValue("object\_name") + " ObjectID: "

+ newDocument.getAttributeValue("r\_object\_id").ToString());

WriteOutput("[ChangeExistingDocument] - ReFetching and Setting title attribute");

newDocument = newDocument.fetch<RestDocument>();

// Now you can set a bunch of attributes. This is the simplest use case, you can also dig into the ImportNewDocument method code and easily figure out how to create a document with all attributes you want, up front, then import it. This use case was somewhat similar to DFC where you create the object, then set the attributes so it was chosen for familiarity.

newDocument.setAttributeValue("title", "Set properties test");

newDocument.Save();

return newDocument;

}

### Updating an Existing Object

Note: Update DQL statements cannot be currently run using Documentum Rest Services.

Making updates to an existing object is as simple as fetching a RestDocument object, using setAttributeValue (String attributeName, String attibuteValue);

For repeating attributes, when you would pass an Object[] array with the values in it. To update a repeating Value, you could first fetch the attribute value using:  
Object[] repValues = doc.getAttributeValue(repeatingAttributename)

You can then cast the array to the appropriate type, modify it, then use doc.setAttributeValue(attributeName, repValues) to update the object.

### Deleting a Document

Deleting a document is as simple as:  
if (doc.CanDelete()) doc.Delete();

More flexible options for deleting a document are available on the Repository object:  
repository.deleteAllDocumentVersions(doc);

repository.deleteCurrentDocumentVersion(doc);

### Deleting a Folder

Folders have a few options associated with them, but the call is pretty simple, using the Repository object:

repository.deleteFolder(tempFolder, isDescending, isAllLinks);

The two options allow you to delete the folder and all content or just the folder

### Checking for Duplicates

Trusted Content Services and a de-duplicating file store are required for this functionality to work.

Why do I need this if I truly have a de-duplicating file store? Deduplication is at the data layer, while you will only store the content once, a new object is created in Documentum that points to that content. When you search for that document, you will see multiple objects in your search results. De-duplication only keeps you from storing the content on disk multiple times, it does not provide de-duplication of result sets from search and queries. De-duplicating file stores cannot do this for you automatically because there are needs for having duplicates. One example is a document needs to be in a restricted areas for users to access it and also has to be in another project folder. You cannot have two different ACLs for one object, so you must store it twice. What this method does is allows you to know if there is another copy of the same content and lets you decide if you want to link the existing to the new location or go ahead and store the additional copy.

In this method, you can pass null for path and it will check the entire repository for duplicates. If you pass a Documentum path, it will report if there are duplicates in that specific location.

private void CheckDuplicates(Repository repository, RestDocument doc, string path)

{

List<PersistentObject> dupes = repository.CheckForDuplicate((String)doc.getAttributeValue("r\_object\_id"), path);

StringBuilder dupeList = new StringBuilder();

if (dupes.Count != 0)

{

if (printResult)

{

bool first = true;

WriteOutput("\t\t\tDocument: " + doc.getAttributeValue("object\_name") + ":" +  
 doc.getAttributeValue("r\_object\_id"));

foreach (PersistentObject pObj in dupes)

{

WriteOutput(String.Format("DUPLICATE OF: {0}", pObj.getRepeatingValuesAsString("parent\_id",",").ToString()));

if (first)

{

dupeList.Append("'" + pObj.getAttributeValue("parent\_id") + "'");

}

else

{

dupeList.Append(",'" + pObj.getAttributeValue("parent\_id") + "'");

}

}

}

if (path == null)

{

WriteOutput("\t\t\t[DeDuplication] - " + dupes.Count + " duplicates were identified in the SYSTEM. Choosing to allow");

}

else

{

WriteOutput("\t\t\t[DeDuplication] - " + dupes.Count + " duplicates were identified in the destination FOLDER, Choosing to allow.");

}

}

else

{

WriteOutput("\t\t\t[DeDuplication] - No Duplicates of this document found.");

}

}

### Copying Documents

The following is an example of copying an object from one folder to another. You can find this code in the UseCaseTests AssignToFolders method where documents originally created under the /Temp cabinet are then copying to their assigned location.

Folder destinationDir = repository.getOrCreateFolderByPath(“/Cabinet/Folder”); // Use your own /Cabinet/Folder path

RestDocument docToCopy = repository.getObjectById<RestDocument>(assignDoc.DocumentId);

// To copy the document, we need to get a reference object

RestDocument copiedDoc = destinationDir.CreateSubObject<RestDocument>(docToCopy.GetCopy<RestDocument>(), null);

### Moving Documents

To move a document from one folder to another, you can call the moveDocument method on the Repository object. You need to source (moveFrom) and destination folder path (moveTo).

RestDocument docToMove = repository.getObjectById<RestDocument>(aDoc.DocumentId);

Folder moveFrom = = repository.getFolderByQualification(

String.Format("dm\_folder where r\_object\_id = '{0}'",

docToMove.getRepeatingString("i\_folder\_id", 0).ToString(), new FeedGetOptions

{ Inline = true, Links = true });

Folder moveTo = repository.getFolderByQualification(

String.Format("dm\_folder where any r\_folder\_path = '{0}'", folderPath,),

new FeedGetOptions { Inline = true, Links = true });

repository.moveDocument(docToMove, moveFrom, moveTo); // Refetch the document if you want to work with it more

### Exporting/Viewing a Document

This is an example from the UseCaseTests class. It shows how you can export a document to a file. You can also get a stream, see the comment in the below code.

public void ViewDocument(Repository repository, String path, string objectId, bool openDocument)

{

RestDocument doc = repository.getObjectById<RestDocument>(objectId);

ContentMeta contentMeta = doc.getContent();

if (contentMeta == null)

{

WriteOutput("!!!!!!!!!!!!!!!!VIEW TEST FAILURE!!!!!!!!!!!!!!!!!!!!");

return;

}

// To get a stream instead, call by calling contentMeta.DownloadContentMediaStream();  
FileInfo downloadedContentFile = contentMeta.DownloadContentMediaFile();

if (!Directory.Exists(path))

{

Directory.CreateDirectory(path);

}

downloadedContentFile.MoveTo(path + Path.DirectorySeparatorChar + objectId + "-" + downloadedContentFile.Name);

WriteOutput("\t\t[GetFileForView] - RestDocument file is located: " + downloadedContentFile.FullName);

if (openDocument) System.Diagnostics.Process.Start(downloadedContentFile.FullName);

}

### Export Folder With All SubFiles and Folders to a Zip Archive

The Repository object can export a folder with all of its contents to a zip file. This example is a method from the UseCaseTests class

private void ExportParent(Repository repository)

{

string parentPath = “/Cabinet/folder go here”;

FileInfo zipFile = repository.ExportFolder(parentPath, testDirectory + Path.DirectorySeparatorChar + parentFolderId + ".zip");

WriteOutput("[ExportFolderToZip] Export Folder completed and stored: " + testDirectory + Path.DirectorySeparatorChar + parentFolderId + ".zip");

}

### Create a Document from a Template

Creating a document from a template is not different that the Copy Document use case, you fetch a template document, then copy it to the location you want to create the new document. Here we get a list of all templates using the ExecuteDQL method of the Repository object like one might do to display a list of templates to a client user interface. Here, we choose a random template to copy. This example is from the UserCaseTests class, CreateFromTemplate method.

Random rnd = new Random();

//get list of templates

Feed<RestDocument> results = repository.ExecuteDQL<RestDocument>(String.Format("select \* from dm\_document where FOLDER('/Templates') "), new FeedGetOptions { Inline = true, Links = true });

List<RestDocument> docs = ObjectUtil.getFeedAsList<RestDocument>(results, true);

int resultSamples = docs.Count;  
RestDocument template = docs[rnd.Next(0, resultSamples)];  
RestDocument newDoc = repository.copyDocument(template.getAttributeValue("r\_object\_id").ToString(), ProcessBasePath + assignPath);  
newDoc.setAttributeValue("subject", "Created From Template: " + template.getAttributeValue("object\_name"));

string documentName = ObjectUtil.NewRandomDocumentName("FROMTEMPLATE");

newDoc.setAttributeValue("object\_name", documentName);

newDoc.Save();

### Is Document Checked out/Check out/Check in/Cancel Checkout

You can check if a document is checked out by calling the IsCheckedOut method of the document object.

Check out is as simple as calling the check out method on a document. Once you check out a document, the document is locked for use by the session user.

To cancel a checkout, you can call the cancel checkout method.

The concept of check in is up to the implementer to decide. To mimic what WDK and some other client applications do you could do what is outlined in the Create/Import as New Version section (below), passing in the same content as was on the previous object.

RestDocument doc = repository.getObjectById<RestDocument>(objectId);

if(!doc.IsCheckedOut()) doc = doc.Checkout();  
doc = doc.CancelCheckout();

### Create/Import As a New Version

To create a new version of an object, you call the repository.ImoortNewVersion method. There are a couple of overloads for this method for convenience. If you pass null for GenericOptions, it will use defaults of format=sameFormatAsPreviousVersion, page=0 , primary=true, versionLabel=”” (CURRENT label will still be added). You also pass a Stream of content. In this case a FileInfo object of file is used and a stream is opened from the file. A sample use case can be found in the UseCaseTests class in the ImportAsNewVersion method.

GenericOptions checkinOptions = new GenericOptions();

checkinOptions.SetQuery("format", ObjectUtil.getDocumentumFormatForFile(file.Name));

checkinOptions.SetQuery("page", 0);

checkinOptions.SetQuery("primary", true);

checkinOptions.SetQuery("version-label", "ImportAsNewVersion");

repository.ImportDocumentAsNewVersion(doc, file.OpenRead(), ObjectUtil.getMimeTypeFromFileName(file.Name), checkinOptions);

### Viewing Audit/Document History

This use case can be found in the checkDocumentHistory of the UseCaseTest class of the Tester application. This snippet shows how to get the document audit history and loop through displaying a couple of attributes on the AuditEntry object.

RestDocument doc = repository.getObjectById<RestDocument>(aDoc.DocumentId);  
WriteOutput("\t" + aDoc.DocumentId + ":" + doc.getAttributeValue("object\_name").ToString() + " - RestDocument History:");

Feed<AuditEntry> auditInfo = repository.getDocumentHistory(HistoryType.THISDOCUMENTONLY, doc);

List<AuditEntry> entries = ObjectUtil.getFeedAsList(auditInfo,true);

foreach (AuditEntry history in entries)

{

WriteOutput("\t\tEvent:" + history.getEventName() + " Description:" + history.getEventDescription()

+ " ObjectName:" + history.getObjectName() + " Time:" + history.getTimeStamp());

}

### Search the Repository

The following example not only show you how to search, but also how to use paged result sets. First you create a Search object, then you set search parameters using methods on the search object.

Search search = new Search();

search.Query = "document";

search.Locations = "/SystemA/Process/" + parentFolderId; // childList[new Random().Next(0, childList.Count)];

search.ItemsPerPage = 20;

int totalResults = 0;

double totalPages = 0d;

WriteOutput("[SearchResults] Return a list of documents using search criteria: " + search.Query + " Location: '" + search.Locations + "'");

SearchFeed<RestDocument> feedResults = repository.ExecuteSearch<RestDocument>(search);

if (feedResults != null)

{

totalResults = feedResults.Total;

totalPages = feedResults.PageCount;

int docProcessed = 0;

for (int i = 0; i < totalPages; i++) // Allows us to iterate through the total pages in our result set

{

foreach (SearchEntry<RestDocument> result in feedResults.Entries)

{

WriteOutput("\t[SearchResults] Search - RestDocument: " + result.Content.getAttributeValue("object\_name").ToString() + " Summary: " + result.Summary

+ " Score: " + result.Score + " Terms: " + String.Join(",", result.Terms));

docProcessed++;

}

if (totalResults > docProcessed) feedResults = feedResults.NextPage();

WriteOutput("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

WriteOutput("Page:" + (i + 1) + " Results: " + docProcessed + " out of " + totalResults + " Processed");

WriteOutput("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

WriteOutput("\n\n");

}

}

WriteOutput("[SearchResults] Result Count: " + totalResults + " Pages: " + totalPages + " Processed in " + ((DateTime.Now.Ticks - tStart) / TimeSpan.TicksPerMillisecond) + "ms");

### Querying the Repository and Processing Pages Result Sets

For getting and processing page result sets, you must specify the ItemsPerPage and IncludeTotal parameters. You can then iterate through the total number of pages through the list of objects on each page. The following example is take from the DqlQueryTest class in the Tester project:

Feed<PersistentObject> queryResult = repository.ExecuteDQL<PersistentObject>(query, new FeedGetOptions() { ItemsPerPage = itemsPerPage, IncludeTotal = true});

String[] attributes = query.Substring(7, query.IndexOf("from")-7).Replace(" ","")  
.Split(new String[] {","}, StringSplitOptions.RemoveEmptyEntries);

if (queryResult != null)

{

int totalResults = queryResult.Total;

double totalPages = queryResult.PageCount;

int docProcessed = 0;

for (int i = 0; i < totalPages; i++)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PAGE " + (i + 1) + "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<PersistentObject> objs = ObjectUtil.getFeedAsList(queryResult, true);

foreach (PersistentObject obj in objs)

{

StringBuilder values = new StringBuilder();

bool first=true;

Console.WriteLine(String.Format("\tName: {0} \t\tID: {1}",

GetAttr(obj, new string[] {"object\_name", "user\_name", "group\_name", "name"}),

GetAttr(obj, new string[] {"r\_object\_id"})));

foreach (string attribute in attributes)

{

if(first) {

values.Append(attribute).Append("=")  
.Append(obj.getAttributeValue(attribute));

} else {

values.Append(",").Append(attribute).Append("=")  
.Append(obj.getAttributeValue(attribute));

}

}

Console.WriteLine(values.ToString());

docProcessed++;

}

if (totalResults != docProcessed) queryResult = queryResult.NextPage();

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Page:" + (i + 1) + " Results: " + docProcessed + " out of " + totalResults + " Processed");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\n\n");

if (pauseBetweenPages)

{

Console.WriteLine("Press any key for next page");

Console.ReadKey();

}

}

}

### Adding a Rendition

Under most circumstances, a rendition server will be automatically creating renditions for document, but you can add your own rendition content using Rest Services. Here you get your RestDocument object, get a file, find its mimeType, then set the Options to pass when adding the rendition. “format” and “page” “primary” (this should be false for a rendition) are required. You can also set the “modifier” parameter which allows you to import renditions of the same format more than once, as long as they have a unique “modifier”.

RestDocument doc = repository.getObjectById<RestDocument>(objectId);

FileInfo file = ObjectUtil.getRandomFileFromDirectory(randomFilesDirectory);

String mimeType = ObjectUtil.getMimeTypeFromFileName(file.Name);

// Upload the content as a new rendition

GenericOptions rendOptions = new GenericOptions();

String format = ObjectUtil.getDocumentumFormatForFile(file.Extension);

rendOptions.SetQuery("format", format);

rendOptions.SetQuery("page", page);

// If you want to allow multiple renditions of the same format, the modifier must be set, this makes the rendition unique in the list

// the "modifier" is more like a label/tag for the rendition in the list.

rendOptions.SetQuery("modifier", "Test");

// With primary false, will be added as a rendition

rendOptions.SetQuery("primary", isPrimary);

ContentMeta renditionMeta = doc.CreateContent(file.OpenRead(), mimeType, rendOptions);

### doc = doc.fetch<RestDocument>(doc); // To refetch and get the latest document with renditions.

### Viewing a Rendition

To view a rendition, you must first have the RestDocument object for the document you wish to view the rendition(s) of. There are three methods you can use to get a rendition:

doc.getRenditionByFormat(dmFormatName);  
doc.getRenditionByFormatAndModifier(dmFormatName, modifierName);  
doc.getRendtiionByModifier(modifierName);

This is case if outlined in the UseCaseTests class of the Tester project as the ViewRenditions method. The following show how to get a rendition by format:

RestDocument doc = repository.getObjectById<RestDocument>(objectId);

ContentMeta renditionMeta = doc.getRenditionByFormat("pdf");

FileInfo downloadedContentFile = renditionMeta.DownloadContentMediaFile();

if (!Directory.Exists(renditionDirectory))

{

Directory.CreateDirectory(renditionDirectory);

}

downloadedContentFile.MoveTo(renditionDirectory + Path.DirectorySeparatorChar + objectId + "-" + downloadedContentFile.Name);

WriteOutput("\t\t[ViewRendition] - Rendition file is located: " + downloadedContentFile.FullName);

if(openDocument) System.Diagnostics.Process.Start(downloadedContentFile.FullName);