

**Lab Center – Hands-on Lab**

**Session 3548**

**Session Title Learn how to Build Application with the IBM Cognos TM1, OData Compliant, REST API**

Guido Tejeda Davila, IBM, gtejeda@us.ibm.com

Hubert Heijkers, IBM, hubert.heijkers@nl.ibm.com

**Table of Contents**

[Getting ready 3](#_Toc464137098)

[Introducing the OData compliant RESTful API 4](#_Toc464137099)

[A first peek at TM1’s RESTful API 4](#_Toc464137100)

[Explore the REST API 6](#_Toc464137101)

[A real life HTML/JavaScript based TM1 client app: TM1Top Lite 6](#_Toc464137102)

[Heading 8](#_Toc464137103)

[Heading 8](#_Toc464137104)

[Heading 8](#_Toc464137105)

[We Value Your Feedback! 9](#_Toc464137106)

[Acknowledgements and Disclaimers 10](#_Toc464137107)

# Getting ready

As always there are a couple of late minute changes that need to be made, updates to the examples, instruction documents that aren’t ready in time etc. etc. Happens every time we do one of these Hands-On Labs as we want you to experience the latest and greatest of our software, and the last coolest ideas we’ve come up with.

As such there are a couple of little steps that need to be executed before your machine is ready.

1 – Grabbing the latest files for the update

The latest versions of the files needed on this box, and the sources you’ll be working with in this lab, are all kept together in one GIT repository on github.com.

Not only that, you’ll be working with Go, a.k.a. Golang, during this lab and Go has super doper support for project, dependency and build management build in. So we’ll grab the latest by executing the following command in a command box:

go get github.com/hubert-heijkers/wow2016

After loading the content of the repository you will receive an error message referring to the fact that there are no buildable Go source files in it, but that error can be ignored.

2 – Updating the Virtual Machine

Next we’ll execute a little batch file the updates a bunch of files and does some set up needed for the lab later on. This update can be executed by typing the following command in the command box:

%GOPATH%\src\github.com\hubert-heijkers\wow2016\hol3548\vmupdate\vmupdate.bat

That’s it, enjoy the lab!

# Introducing the OData compliant RESTful API

TM1 Servers, as of version 10.2 RP2, exposes an OData compliant, RESTful API. This was the first PUBLIC version of a RESTful API. In the meantime, a number of fix packs have been released with additional improvements and extensions to the REST API. It is safe to say that, even though still relatively new, it is very mature and the best performing API we have available for TM1. And in case you had any doubt, it will be THE API for the TM1 Server going forward.

Now you might wonder what the being “OData compliant” is all about. Well, ignoring the fact that a bunch of people from various companies, organized in an OASIS technical committee have brought together all the knowledge and experience they had to offer, it is simply said a set of specifications which we obey by that specify how a service describes what is available to a consumer, how a consumer formulates a request for a compliant server and how the service formats the response to the request. So it’s not just saying use the telephone and speak English but, more importantly, having agreed on the ‘topic’, a syntax for any requests, formulation the responses and the dictionary being used in the conversation.

OData, short for Open-Data, has been developed over a number of years as well and the latest version, v4 errata 3, is an OASIS standard. The OData standards has also made it to ISO standard in the meantime as well. For more information about OData standard and the documents describing it please visit the OData.org website at: <http://www.odata.org>. For a quick introduction to the OData standard have a look at the ‘[Understanding OData in 6 steps](http://www.odata.org/getting-started/understand-odata-in-6-steps/)’ webpage.

## A first peek at TM1’s RESTful API

So let’s start with having look at the metadata of the TM1 server first.

1. Start Google Chrome.
2. Retrieve the metadata document by typing the following URL in the address bar: <http://tm1server:8000/api/v1/$metadata>

The metadata for the TM1 server will be shown in your browser. It’s and XML document formatted according to the CSDL specification which is part of the OData standard. It describes all the types, entity and complex types, all entity sets and relationships between entity and complex types in the service. For example, the ‘Dimension’ entity is described as (excluding the documentation annotations):

<EntityType Name="Dimension">  
 <Key>  
 <PropertyRef Name="Name"/>  
 </Key>  
 <Property Name="Name" Type="Edm.String" Nullable="false"/>  
 <Property Name="UniqueName" Type="Edm.String"/>  
 <Property Name="Attributes" Type="ibm.tm1.api.v1.Attributes"/>  
 <NavigationProperty Name="Hierarchies" Type="Collection(ibm.tm1.api.v1.Hierarchy)" Partner="Dimension" ContainsTarget="true"/>  
 <NavigationProperty Name="DefaultHierarchy" Type="ibm.tm1.api.v1.Hierarchy"/>  
 <NavigationProperty Name="LocalizedAttributes" Type="Collection(ibm.tm1.api.v1.LocalizedAttributes)" ContainsTarget="true"/>  
</EntityType>

This is telling us that one of the types that the service exposes is a ‘Dimension’ and that it has a couple of properties among which is its Name, UniqueName and a set of Hierarchies (note: the version of TM1 you are using still only allows one Hierarchy per Dimensions, which has to have the same name then the dimension, but the REST API is ‘future proof’ in that it already supports alternate hierarchies, a feature which is tentatively planned for the upcoming v11 release). The Name is the property that uniquely identifies the Dimensions, and as such acts as the key.

As you scan the metadata file you’ll see all the types available and how they relate to each other and it is this metadata document that consumers of the service will use to understand what is available in the REST API.

Let’s be a consumer for a sec and, knowing what’s available in the service, start retrieving some data from the service. So let’s look at the list of the ‘Dimensions’ that we have in the service and, while at it, ignore those control dimensions (the dimensions starting with the ‘}’ character).

1. Retrieve those dimensions not being control dimensions by typing the following URL: [http://tm1server:8000/api/v1/Dimensions?$filter=not startswith(Name,'}')](http://tm1server:8000/api/v1/Dimensions?$filter=not%20startswith(Name,'%7d'))
2. If this is the first time you are accessing a secured resource, you’ll be challenged for a username and password. If this happens use the famous “admin” and “apple” pair.

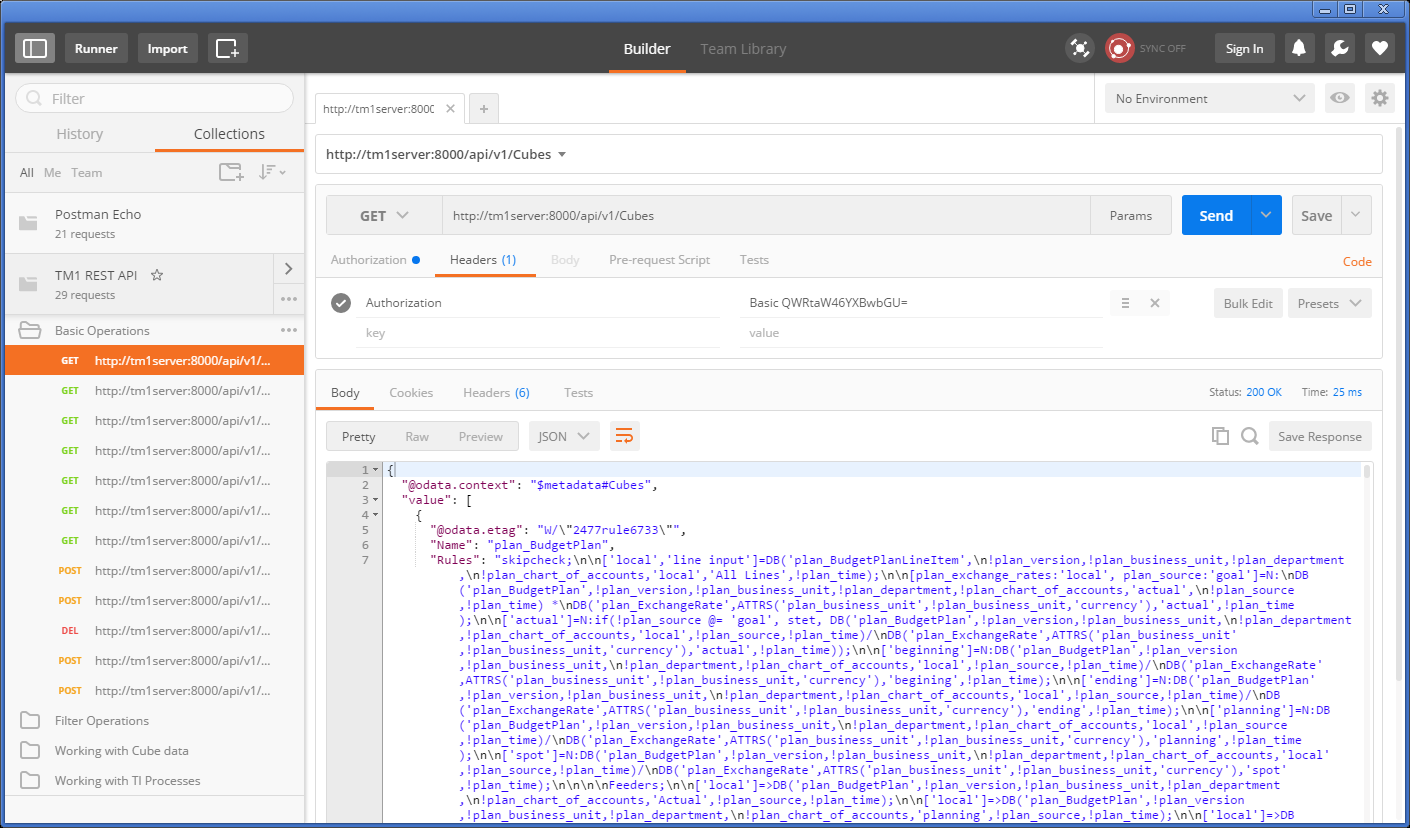
You’ll get the list of dimensions available shown in your browser nicely formatted because we installed the JSONView plug-in for Chrome.



If you want to see what went over ‘the wire’ you can start Fiddler, by clicking the  icon in the taskbar. Once Fiddler is up it’ll start recording HTTP traffic and you can look at the requests going to and the responses returned by the server. This way you’ll see for example that the JSON going over the wire is pretty compact and that we, provided the client supports it, apply compression to the response.

## Explore the REST API

Ok, it’s time for some more examples. To make it easier to interact with our, any for that matter, HTTP/REST based service we use Postman. Click on the  icon in the taskbar to start Postman.



After starting Postman you’ll find, under the Collections tab on the left, a collection named ‘TM1 REST API’. A bunch of examples have been included in this collection to give you an initial feel of what the REST API can do for you and how it works.

Note: If you don’t see the ‘TM1 REST API’ collection, not to worry, hit the ‘Import’ button on the top, open a file explorer, locate the ‘C:\HOL-TM1SDK\postman\_collections’ folder and drop the ‘TM1%20REST%20API.json.postman\_collection’ file in the screen that opened up.

After selecting an example, you can see the definition of the request on the right. Hitting the ‘Send’ button will execute the request after which the response will be shown to you in the output window. Don’t forget to look at the Cookies and Headers tabs in the output pane to see what more is being send forth and back between the client, Postman in this case, and the TM1 Server.

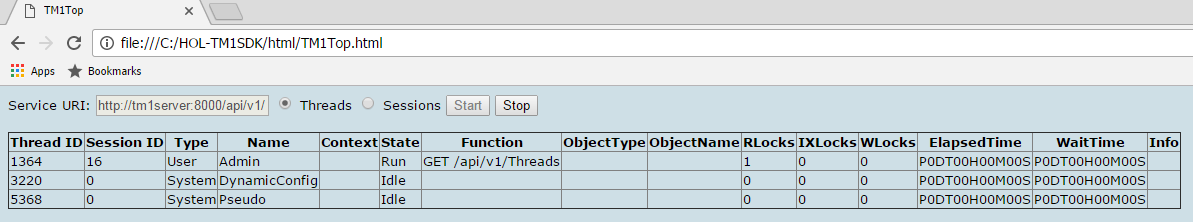
Postman is a very convenient tool to test requests. If you haven’t done so already we’d advice you to download it and install it in your environment and have a go at it. Want the collection of tests from this lab, don’t hesitate to contact any of the presenters and we’ll send it to you. Have fun!

## A real life HTML/JavaScript based TM1 client app: TM1Top Lite

To illustrate how quick and easy it is to build client applications using the new TM1 REST API, have a look at our TM1Top “Lite” sample application. It’s a simple, standalone, web client that periodically retrieves the active threads and inserts them into a table. It’s obviously not pretty but it is functional. And, since we are using a recent enough version, it is capable of showing the threads by session too.

You can find the sample at [file:///C:/HOL-TM1SDK/html/TM1Top.html](file:///C:\HOL-TM1SDK\html\TM1Top.html). Open it using Chrome. If you’re curious to see how it’s implemented, take a look at the source code by right-clicking anywhere in the web page and click “View page source”.

If you wonder what all the fiddling with security modes is about, have a look at the ‘[Using CAM Authentication with TM1’s, OData compliant, REST API](https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W181f1083f3dd_455f_b2f8_f63c4a9c8010/page/Using%20CAM%20authentication%20with%20TM1's%2C%20OData%20compliant%2C%20RESTful%20API)’ article on [developerWorks TM1 SDK community](https://www.ibm.com/developerworks/community/groups/service/html/communitystart?communityUuid=94a0a656-48ea-436d-ac5c-c711caf02e85).



HTML/JavaScript is only one of the many ways to consume the new TM1 REST API. In the next section, we’ll show you how to build applications that connect to TM1 using C#, C++ and Java. These are simply examples, you can choose to build your applications with your choice of language/environment running on any OS as long as it supports making HTTP requests and you have means to compose and parse JSON.

# Building your first model using the REST API

Consuming data and metadata thru the REST API is one, and likely what most consumers will end up doing but it doesn’t stop there. Obviously one can create, update and delete objects, like dimensions and cubes, as well. So in this section we’ll use the REST API to build a complete, one cube, model from scratch. As the source for this model we are going to use the [NorthWind](http://services.odata.org/V4/Northwind/Northwind.svc/) database, hosted on the [OData.org website](http://www.odata.org/), which itself is also a, relational, OData compliant, database.

One of the things we can’t do, yet, is create a complete new model (read: server). So we’ll start with doing that the old fashion way, which means:

* Creating a data directory that is going to contain all the data for our model
* Create a tm1s.cfg file in that directory with the configuration for our new model
* Create a shortcut to start the new TM1 server representing our new model
* Start it!

On our lab VM machine we are storing the data for our TM1 models in the C:\HOL-TM1SDK\models folder. We are going to call our new service ‘NorthWind’ as per the data source name, so we’ll start with creating a new directory in the C:\HOL-TM1SDK\models folder called ‘NorthWind’.

In this newly create NorthWind folder we’ll create a new text file named tm1s.cfg

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

# Heading

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

# Heading

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

------------------------------------------------------------------ Text -----------------------------------------------------------------

# We Value Your Feedback!

* Don’t forget to submit your World of Watson session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
* Access the World of Watson Conference Connect tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.

# Acknowledgements and Disclaimers