

IBM z/OS Connect EE V3.0

# Developing RESTful APIs for DVM Services



*Lab Version Date: June 20, 2019*

# Table of Contents

---

<b>Overview .....</b>	<b>3</b>
<b>Create Data Virtualization Manger services .....</b>	<b>4</b>
<i>Use the Data Virtualization Manager Studio to create a virtual table .....</i>	<i>4</i>
<i>Use the Data Virtualization Manager Studio to create a web service .....</i>	<i>13</i>
<i>Use the Data Virtualization Manager Studio to deploy the services .....</i>	<i>20</i>
<b>Create z/OS Connect EE APIs .....</b>	<b>25</b>
<i>Connect to a z/OS Connect EE Server.....</i>	<i>25</i>
<i>Create the DVM API Project .....</i>	<i>28</i>
<i>Import the SAR files generated by the DVM Studio .....</i>	<i>30</i>
<i>Compose an API for DVM Rest Services .....</i>	<i>32</i>
<i>Deploy the API to a z/OS Connect EE Server .....</i>	<i>46</i>
<i>Test the DVM APIs.....</i>	<i>48</i>

## Overview

The objective of these exercises is to gain experience with working with the Data Virtualization Manager (DVM) Studio and the z/OS Connect EE API Toolkit. These two products allow the exposure of z/OS resources to JSON clients. More in-depth information about the customization of z/OS Connect EE, z/OS Connect EE security, the use of the API Toolkit and other topics is provided by the 1-day *ZCONNEE – z/OS Connect Workshop*. For information about scheduling this workshop in your area contact your IBM representative.

## *General Exercise Information and Guidelines*

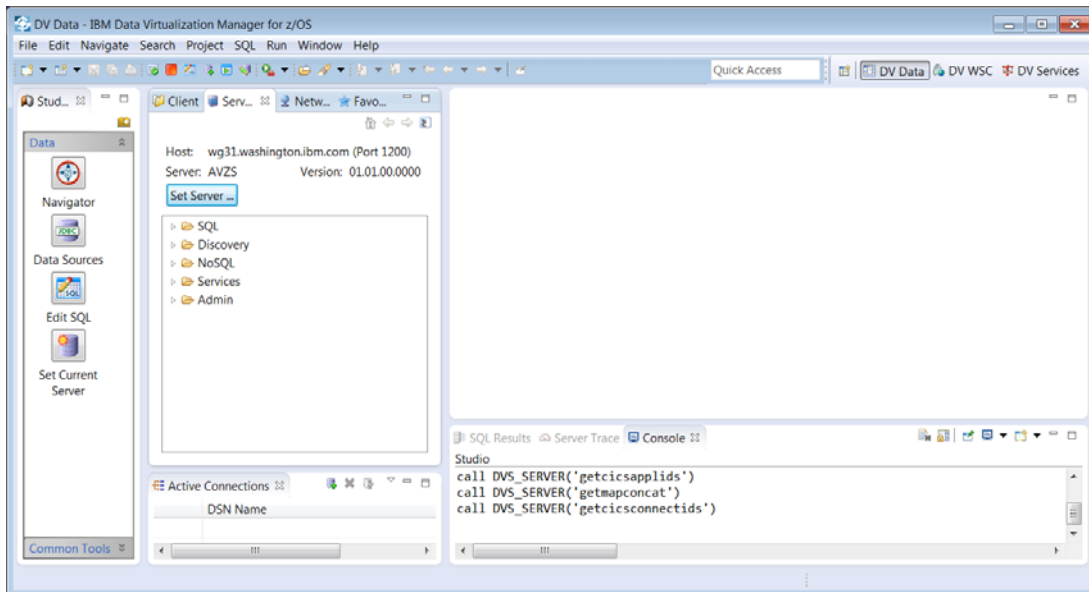
- ✓ This exercise requires using z/OS user identity *USER1*. The password for this user will be provided by the lab instructor.
- ✓ Any time you have any questions about the use of the Data Virtualization Manager Studio, IBM z/OS Explorer, z/OS Connect EE Toolkit features or tools do not hesitate to ask the instructor for assistance.
- ✓ The VSAM data set being used for this exercise is the VSAM data set provided by the CICS Catalog Manager sample application. For details on this VSAM data set see member DFH\$ECAT in the CICS SDFHINST target data set.
- ✓ Text in **bold** and highlighted in **yellow** in this document should be available for copying and pasting in a file named *Development APIs CopyPaste* file on the desktop.

## Create Data Virtualization Manger services

### *Use the Data Virtualization Manager Studio to create a virtual table*

Begin by starting the Data Virtualization Manager Studio toolkit.

1. On the workstation desktop, locate the Data Virtualization Manager Studio icon and double click on it to open the tool. You should automatically be connected to the DVM server running on z/OS, see below.



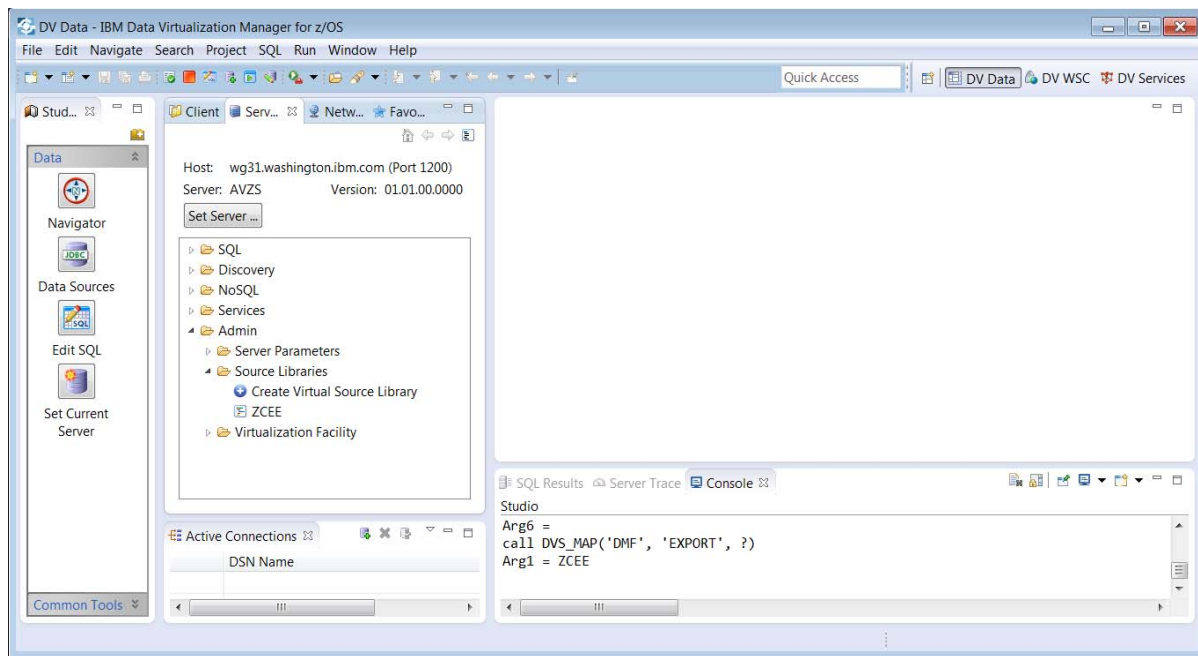
**Tech-Tip:** Eclipse based development tools like DVM Studio; provide a graphical interface consisting of multiple views within a single window.

A view is an area in the window dedicated to providing a specific tool or function. For example, in the window above, *Console*, *Studio Navigator* and *Server*, are views that use different areas of the window for displaying information. At bottom on the right there is a single area for displaying the contents of three views stacked together (commonly called a *stacked views*), *Console*, *SQL Results* and *Server Trace*. In a stacked view, the contents of each view can be displayed by clicking on the view tab (the name of the view).

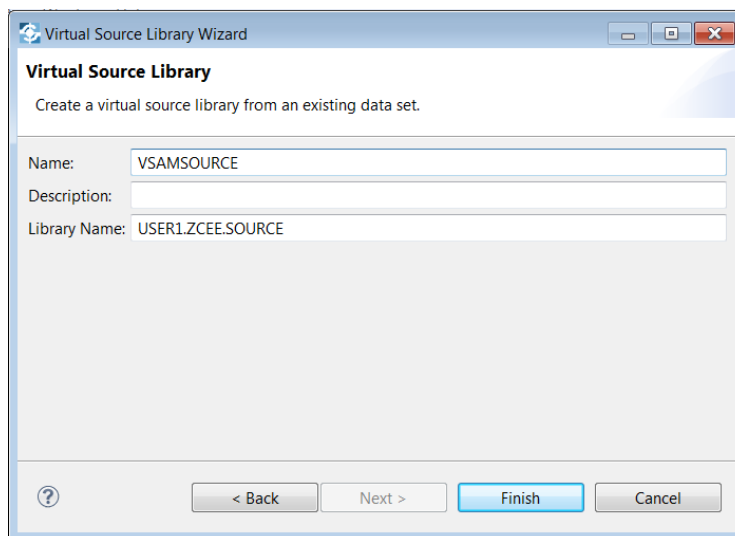
At any time, a specific view can be enlarged to fill the entire window by double clicking in the view's title bar. Double clicking in the view's title bar will be restored the original arrangement. If a DVM Studio view is closed or otherwise disappears, the original arrangement can be restored by selecting **Windows → Reset Perspective** in the window's tool bar.

Eclipse based tools also can display multiple views based on the current role of the user. In this context, a window is known as a perspective. The contents (or views) of a perspective are based on the role the user, i.e., developer or administrator.

2. In the *Server* view expand *Admin* then expand *Source Libraries* to display the *Create Virtual Source Library* wizard, see below.

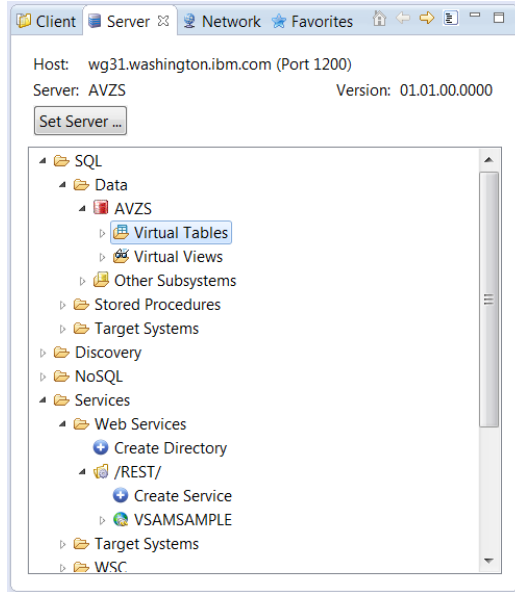


3. Double click the *Create Virtual Source Library* wizard and on the *New - Select a wizard* window select *Data Set* and click **Next** to continue.
4. On the *Virtual Source Library* enter **VSAMSOURCE** as the *Name* of the virtual source library and **USER1.ZCEE.SOURCE** as the *Library Name* and press **Finish** to continue.

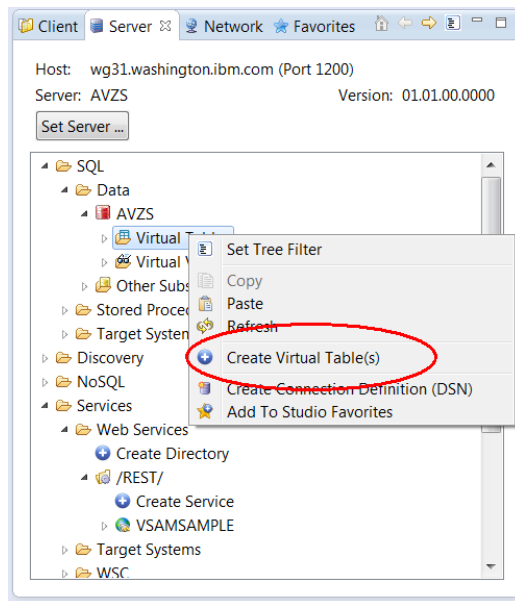


**Important:** The values used for names are somewhat arbitrary, but they do relate to later tasks. If you use the values and cases as supplied then in subsequent windows, results, etc. will be consistent with what is shown in this document.

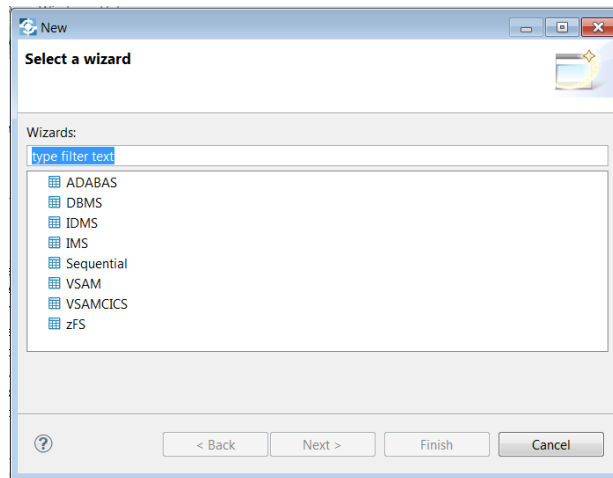
- \_\_\_5. Next expand the *SQL* folder then the *Data* folder and then the *AVZS* folder to display the *Virtual Tables* and *Virtual Views*, see below.



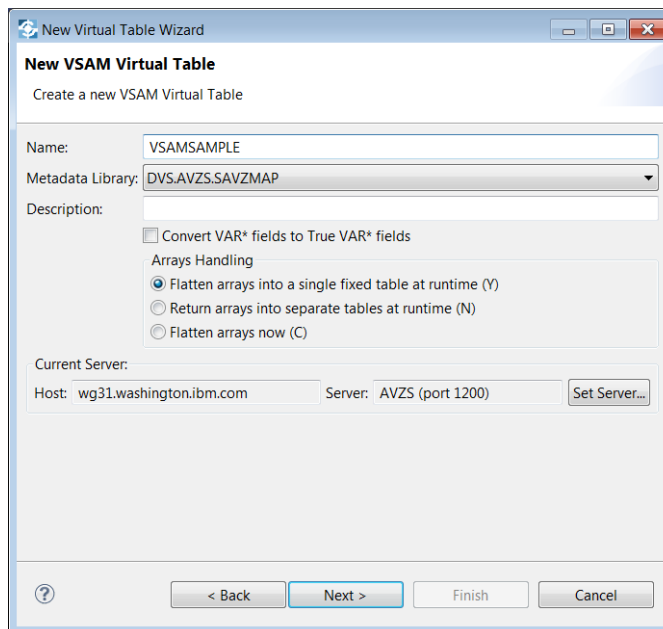
- \_\_\_6. Select the *Virtual Tables* folder and right mouse button click and then select *Create Virtual Table(s)* property, see below.



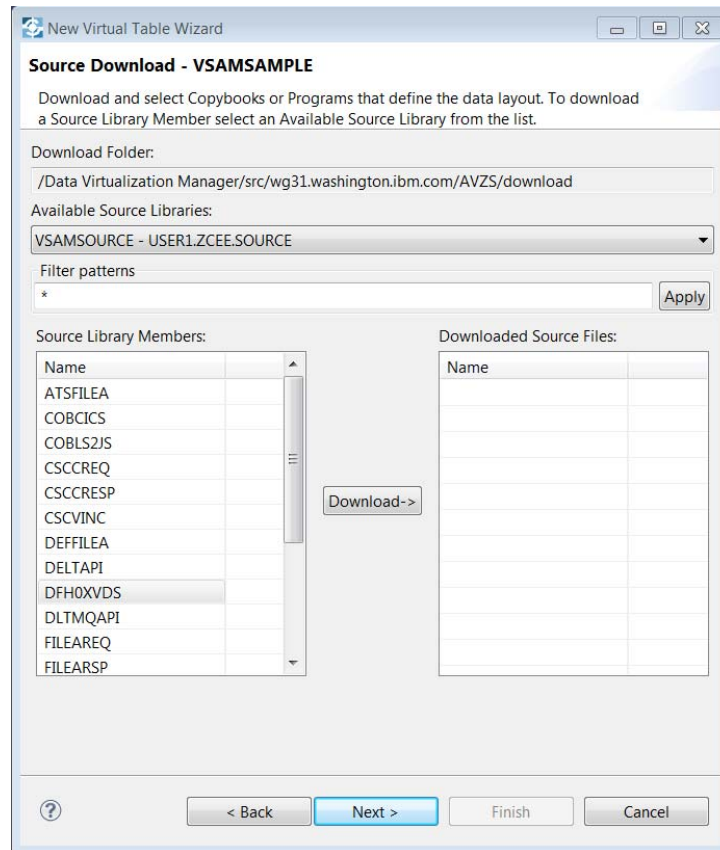
7. On the *Select a wizard* window select *VSAM* and press **Next** to continue.



8. On the *New VSAM Virtual Table* window enter *VSAMSAMPLE* as the name and press **Next** to continue.



9. On the *Source Download – VSAMSAMPLE* window use the pull-down arrow to select the *VSAMSOURCE-USER1.ZCEE.SOURCE* source library (created earlier). This will download a list of the members in this partitioned data set. Select member *DFH0XVDS* and use the **Download** button to have this source downloaded to the workstation.





10. Click **Next** to continue. This will display the *Virtual Table Layout – VSAMSAMPLE* window. Scroll down and expand the COBOL *WORKFIELDS* structure until the *WS-CAT-ITEM* structure is displayed (see below). This structure represents the actual layout of a record in the VSAM file. Select *WS-CAT-ITEM* and click **Next** to continue.

**New Virtual Table Wizard**

**Virtual Table Layout - VSAMSAMPLE**

Select the starting field that defines the data layout.

Available Files: DFH0XVDS

Source	
01	WORKFIELDS.
03	WS-CURRENT-ITEM-REF PIC 9(4).
03	WS-RESPONSE-CODE PIC 89(8) COMP.
03	WS-LOOP-COUNTER PIC 89(2) COMP.
03	WS-RECORD-COUNT PIC 89(2) COMP.
03	WS-RECORD-COUNT-DISPLAY PIC +9(2) USAGE DISPLAY.
03	<b>WS-CAT-ITEM.</b>
05	WS-ITEM-REF PIC 9(4).
05	WS-DESCRIPTION PIC X(40).
05	WS-DEPARTMENT PIC 9(3).
05	WS-COST PIC ZZ2.99.
05	WS-IN-STOCK PIC 9(4).
05	WS-ON-ORDER PIC 9(3).
05	FILLER PIC X(20).

Start Field: WS-CAT-ITEM

☐ Enable End Field selection

End Field:

< Back Next > Finish Cancel

11. On the *VSAM Data Source Details – VSAMSAMPLE* window enter **USER1.OFFICE.SUPPLIES** as the *Cluster Name* and press the **Validate** button.

**New Virtual Table Wizard**

**VSAM Data Source Details - VSAMSAMPLE**

① Alternate indexes are optional. Use the GET button to populate the table.

Cluster Name:

Post-Read Exit Name:

Pre-Write Exit Name:

Alternate Indexes:

Path Name

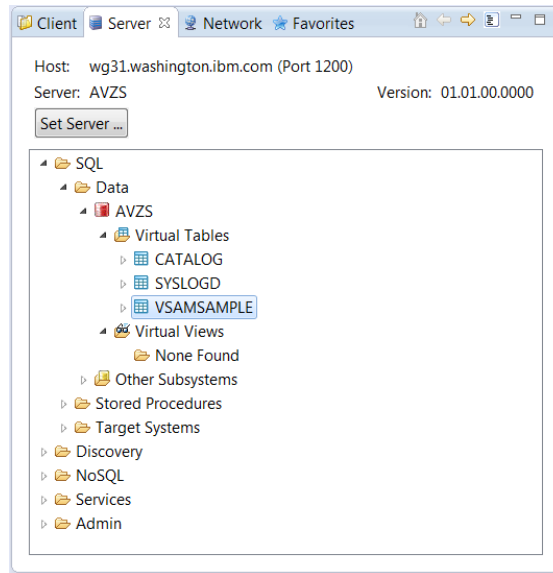
12. This will validate that the VSAM data set does exist and will display the type of VSAM access allowed, e.g., KSDS, ESDS, RRDS.

**Cluster Name Validation**

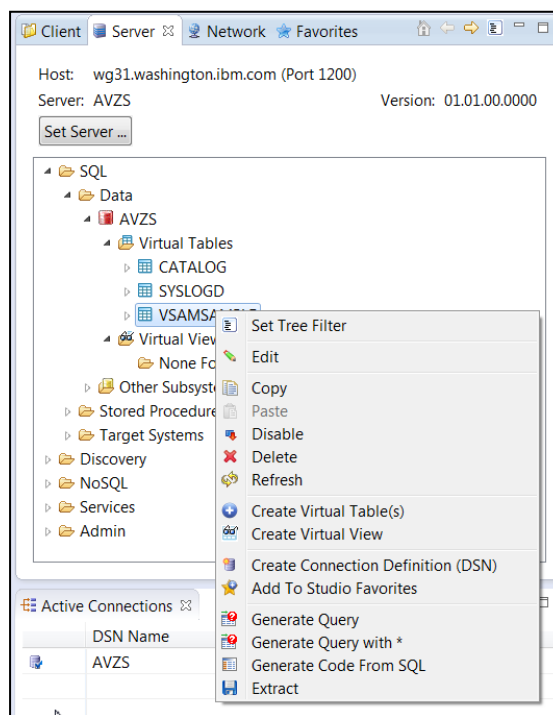
① The Cluster Name is valid, type=KSDS

13. Click the **OK** button and then the **Finish** button to continue.

14. In the list of Virtual Tables an entry for *VSAMSAMPLE* should now appear. Select *VSAMSAMPLE* and right mouse button click.



15. Select option *Generate Query* and click **Yes** on the *Execute Query?* pop up window.



16. This will access the VSAM data set and display the contents of this VSAM data set in the view on the lower right-hand side

SQL Results							
Server Trace Console							
	WS_ITEM_REF	WS_DESCRIPTION	WS_DEPARTMENT	WS_COST	WS_IN_STOCK	WS_ON_ORDER	
0	10	Ball Pens Bla...	10	002.90	135	0	
1	20	Ball Pens Blu...	10	002.90	6	50	
2	30	Ball Pens Red...	10	002.90	106	0	
3	40	Ball Pens Gre...	10	002.90	80	0	
4	50	Pencil with e...	10	001.78	83	0	
5	60	Highlighters ...	10	003.89	13	40	
6	70	Laser Paper 2...	10	007.44	102	20	
7	80	Laser Paper 2...	10	033.54	25	0	
8	90	Blue Laser Pa...	10	005.35	22	0	
9	100	Green Laser P...	10	005.35	3	20	
10	110	IBM Network P...	10	169.56	12	0	
11	120	Standard Diar...	10	025.99	7	0	
12	130	Wall Planner:...	10	018.85	3	0	
13	140	70 Sheet Hard...	10	005.89	84	0	
14	150	Sticky Notes ...	10	005.35	36	45	
15	160	Sticky Notes ...	10	009.75	67	30	
16	170	Sticky Notes ...	10	007.55	64	30	
17	180	Highlighters ...	10	003.49	88	10	
18	190	Highlighters ...	10	003.49	76	20	
19	200	12 inch clear...	10	002.12	14	10	
20	210	Clear sticky ...	10	004.27	73	0	

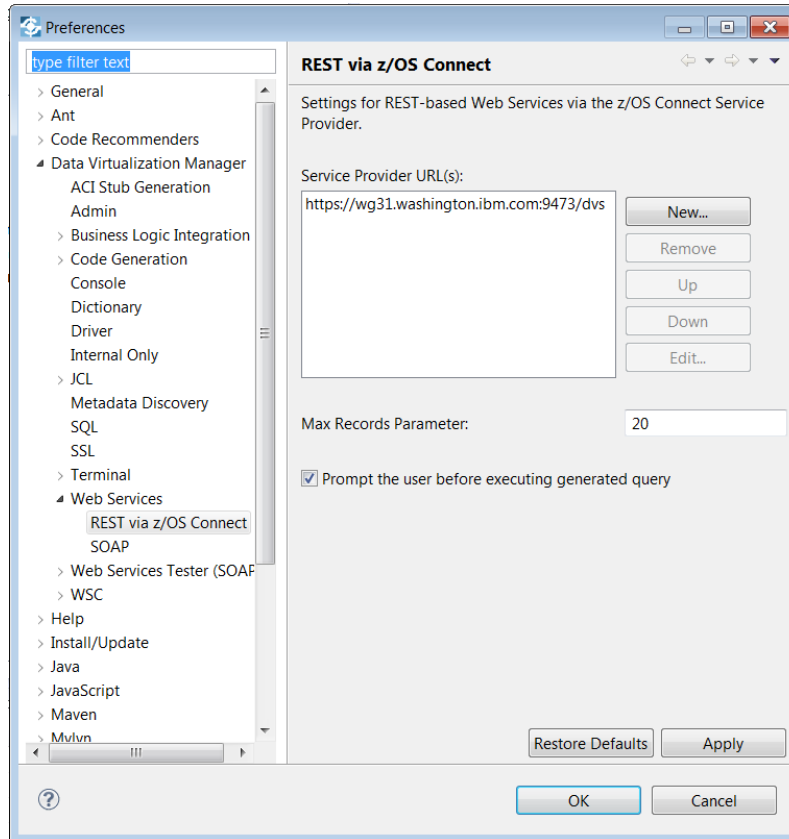
Columns Group: 1 of 1 Columns per group: 25

21 rows SQL Messages

This completes the creation of the virtual tables for this VSAM data set.

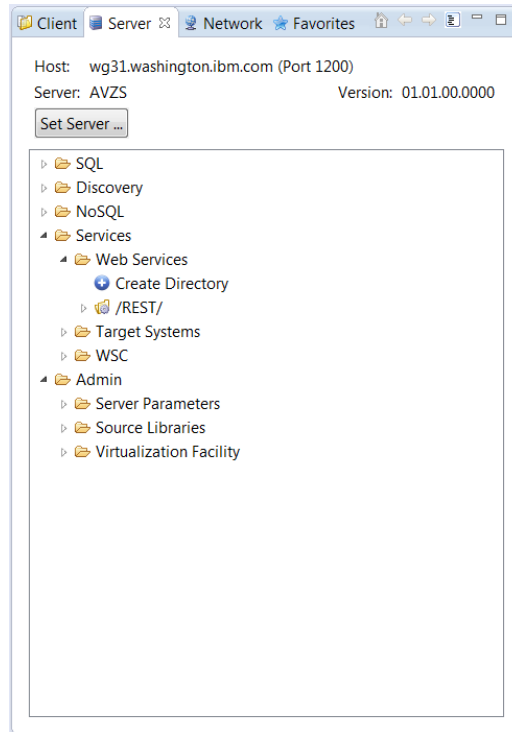
## *Use the Data Virtualization Manager Studio to create a web service*

1. Confirm that the DVM studio is properly configured to communicate with the z/OS Connect EE server which has the DVM service provider installed. On the DVM Studio toolbar click on *Windows* then *Preferences* to display the Eclipse *Preferences* window. Expand *Data Virtualization Manager* and then *Web Services* to select *REST via z/OS Connect*, see below:

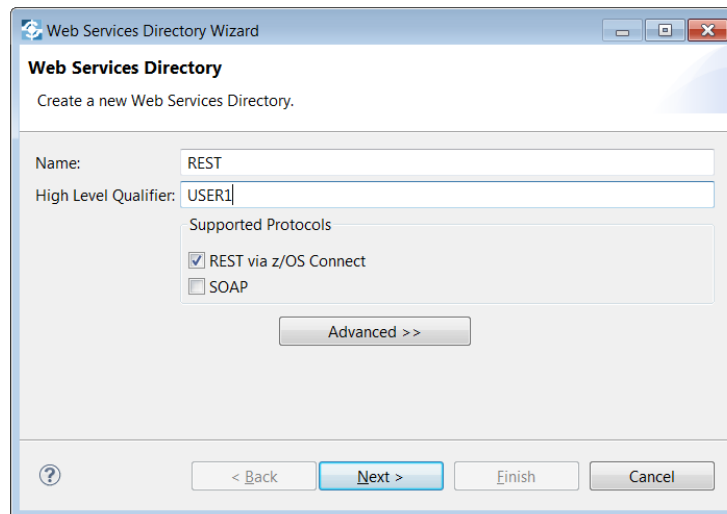


2. Ensure the *Service Provider URL(s)* is set to **<https://wg31.washington.ibm.com:9473/dvs>**. If not, select the provider and use the **Edit** button to set it correctly.

3. Back in the *Server* view expand the *Services* and then the *Web Services* folders

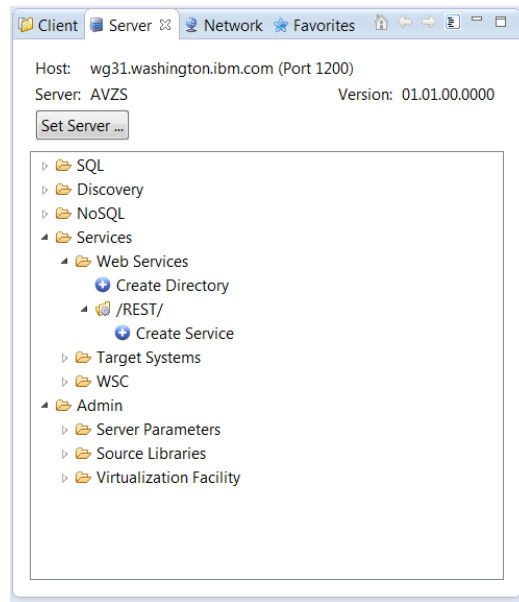


4. If the */REST/* web services directory does not exist use the Create Directory wizard to create the directory and a Microflow Library as shown in the initial window below:



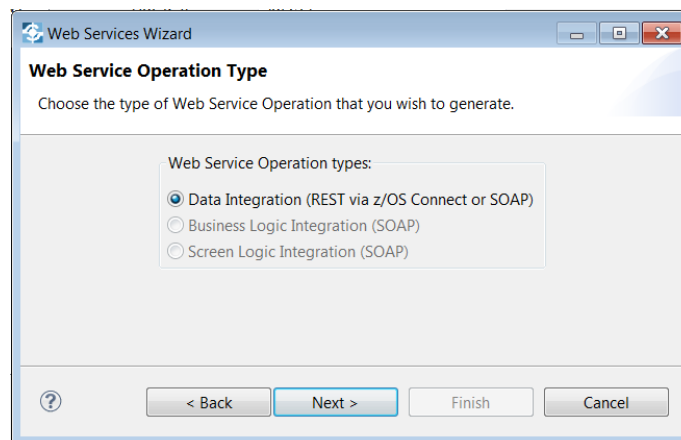
**Tech-Tip:** The High Level Qualifier will be used to create a micro flow partitioned data set with a data set name of USER1.MFL.

\_\_\_ 5. Expand */REST/* and use the *Create Service* wizard to create a new web service.

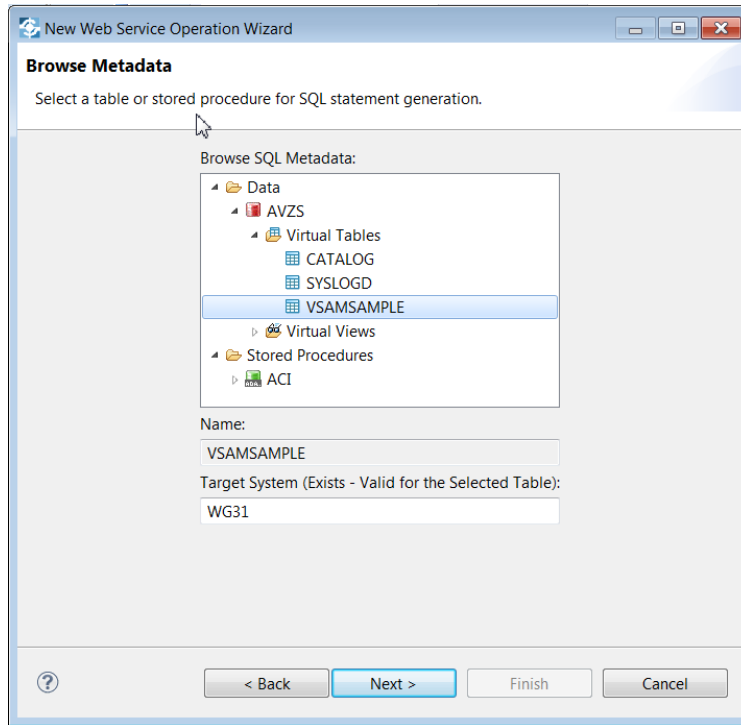


\_\_\_ 6. On the *Web Service – Create a new Web Service* window enter **VSAMSAMPLE** as the *Name* and press **Next** to continue.

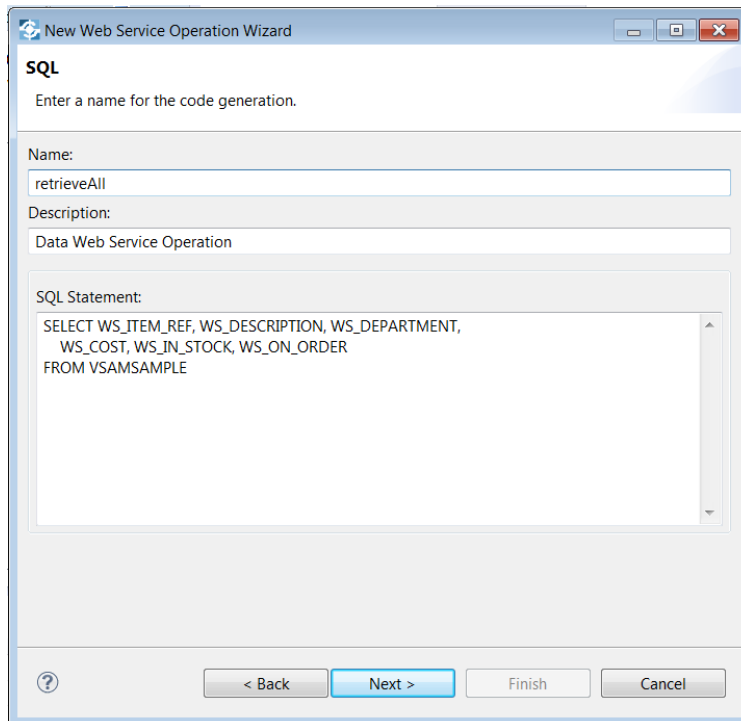
\_\_\_ 7. On the *Web Service Operation Type* window ensure the radio button beside *Data Integration (REST via z/OS Connect or SOAP)* is selected and click **Next** to continue.



8. On the *Browse Metadata* windows expand the *Data* folder then the *AVZS* folder and then the *Virtual Tables* folder to display the virtual table created in the previous section. Select *VSAMSAMPLE* and press **Next** to continue.

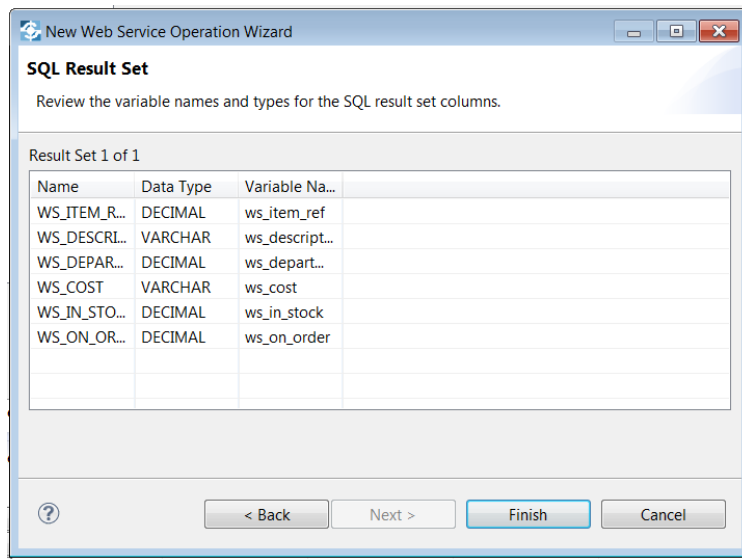


9. The default SQL statement will be displayed on the *SQL* window. Change the name of the operation to ***retrieveAll*** to indicate that this operation will retrieve all records from the VSAM data set. Click **Next** to continue.





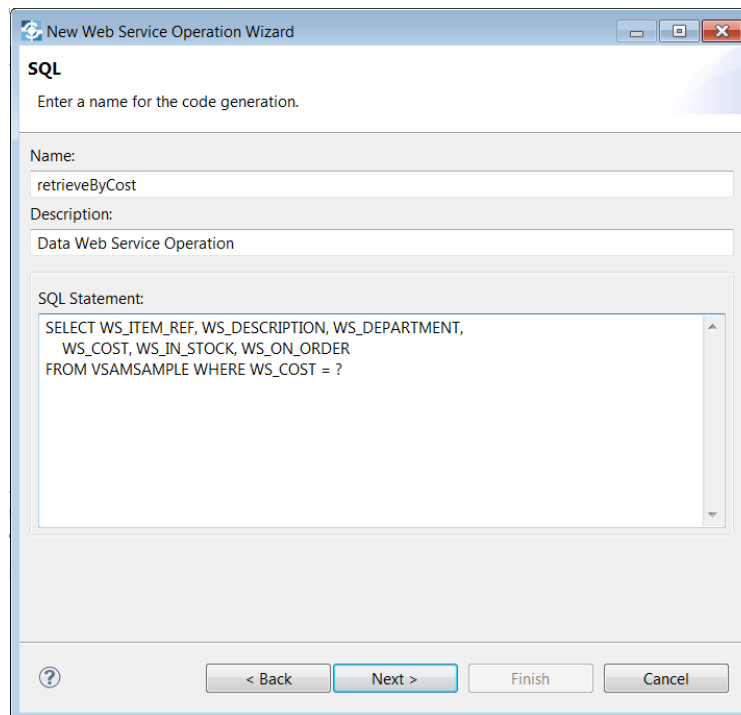
10. The next window to be displayed will show the results that will be returned when the operation is executed.



**Tech-Tip:** This is useful because we could have removed some of the columns on the previous windows. This display simply confirms which columns will be returned.

11. Click **Finish** to continue.

12. Use the *Create Operation* wizard under *VSAMSAMPLE* to create a new operation. This operation should be named **retrieveByCost** and the *SQL Statement* should be changed by adding **WHERE WS\_COST = ?**, see below:



13. Click **Next** to continue. Since a WHERE clause has been added with a variable, providing a value for this variable will be required. The next window to be displayed, *SQL Inputs*, will give us a chance to give meaningful names to this and other variables. On this window click on the value of variable name in the *Name* column and change the contents to **COST**. Click **Next** to continue.

Name	Data Type	Default Val...
COST	VARCHAR	

14. The columns that will be returned are displayed on the SQL Result Set window. Click **Finish** to continue.

15. Use the *Create Operation* wizard under *VSAMSAMPLE* to create a new operation. This operation should be named **retrieveByInventory** and the *SQL Statement* should be changed by adding **WHERE WS\_ON\_ORDER < ? AND WS\_IN\_STOCK < ?**, see below:

SQL Statement:

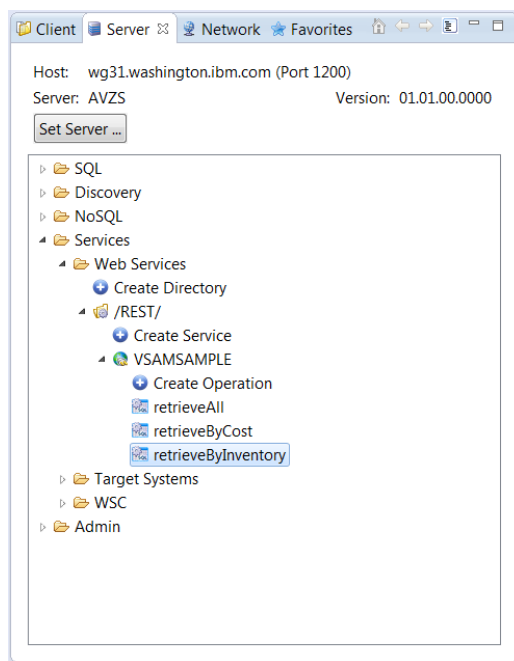
```
SELECT WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTMENT,
       WS_COST, WS_IN_STOCK, WS_ON_ORDER
FROM VSAMSAMPLE WHERE WS_ON_ORDER < ? AND WS_IN_STOCK < ?
```

16. Click **Next** to continue. Since a *WHERE* clause has been added with variables, providing values for these variables will be required. The next window to be displayed, *SQL Inputs*, will give us a chance to give meaningful names to these variables. On this window click on the values of variable name in the *Name* column and change the contents to *onOrder* and *instock*. Click **Next** to continue.

Name	Data Type	Default Val...
onOrder	VARCHAR	
instock	VARCHAR	

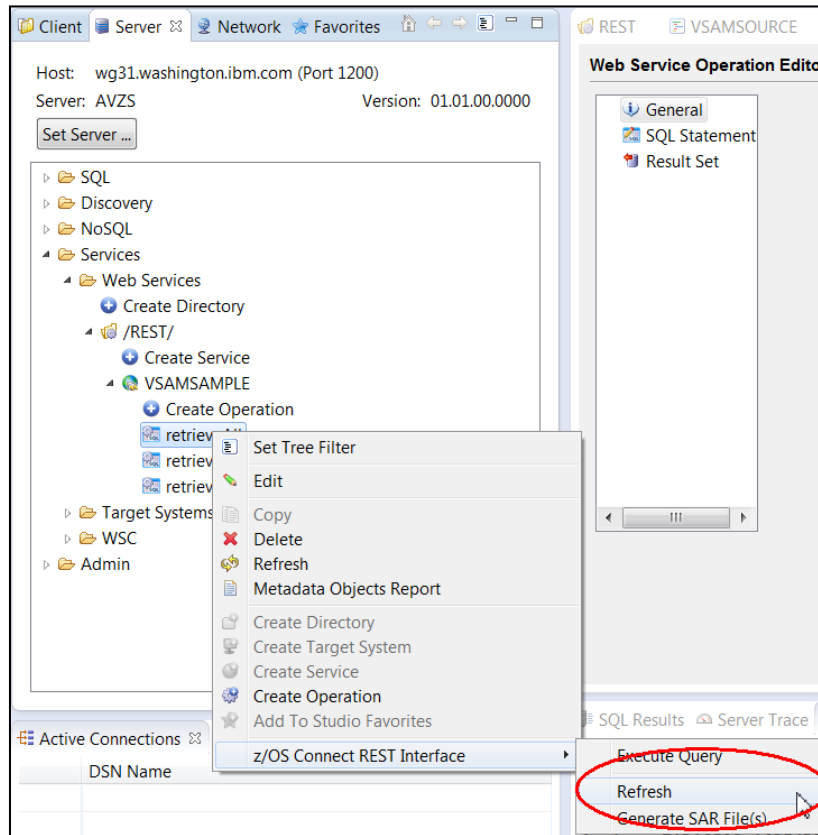
17. The columns that will be returned are displayed on the *SQL Result Set* window. Click **Finish** to continue.

18. All three operations should now be displayed in as services in the *Server* view.



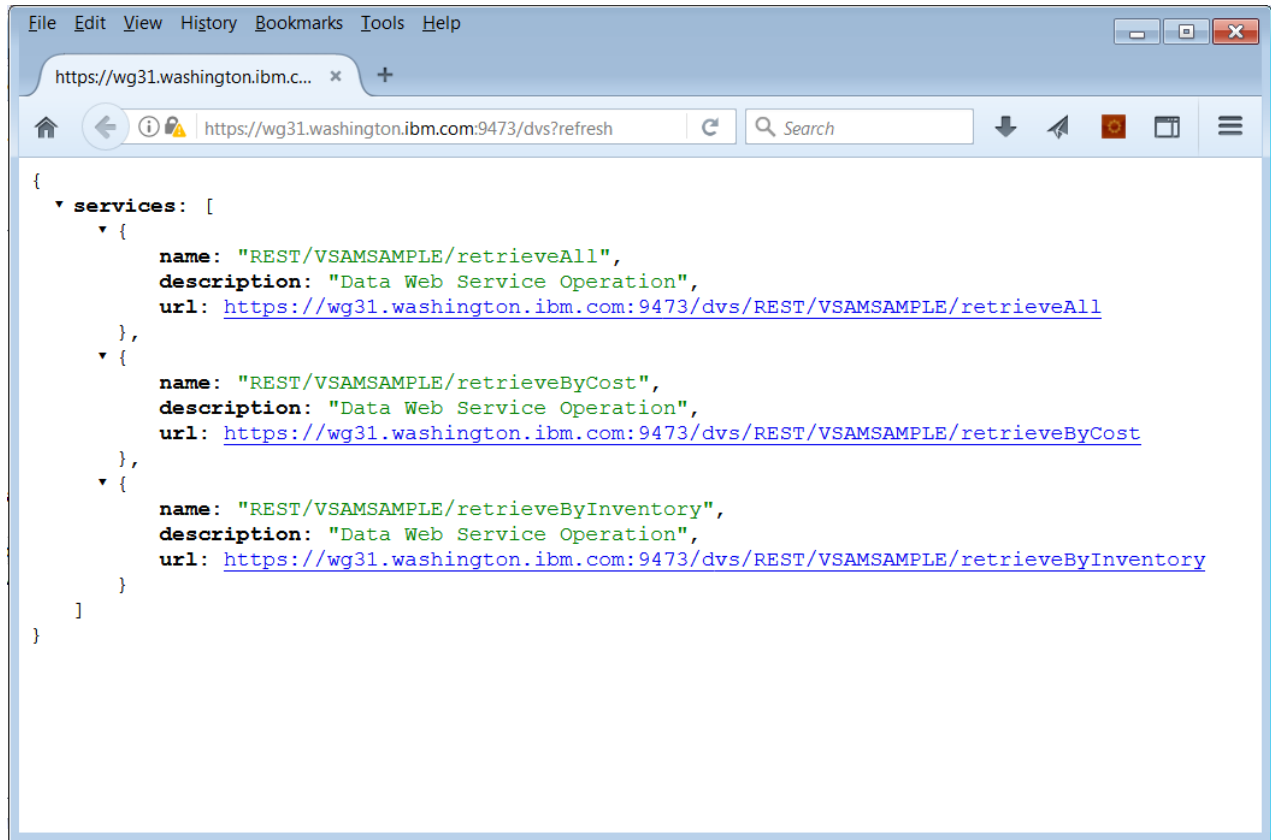
## Use the Data Virtualization Manager Studio to deploy the services

1. These operations need to be deployed to z/OS Connect server. Select each operation in turn and right mouse button click. Select the *z/OS Connect REST Interface* option then the *Refresh* option. This will install the selected operation in to the z/OS Connect EE server as services.



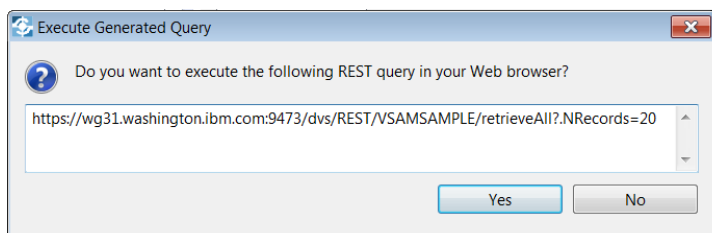
**Tech Tip:** You may be challenged by Firefox because the digital certificate used by the Liberty z/OS server is self-signed. Click the **Add Exception** button to continue. If the **Add Exception** button is not displayed, click the **Advanced** button. Then click on the **Confirm Security Exception** button. Next you may see a prompt for a userid and password. If you do see the prompt, enter the username **USER1** and USER1's password and click **OK**. Remember we are using basic security and this is the user identity and password defined in the server.xml file.

2. When finished all the operations should be displayed as services in the z/OS Connect EE server by entering URL **<https://wg31.washington.ibm.com:9473/dvs?refresh>**.

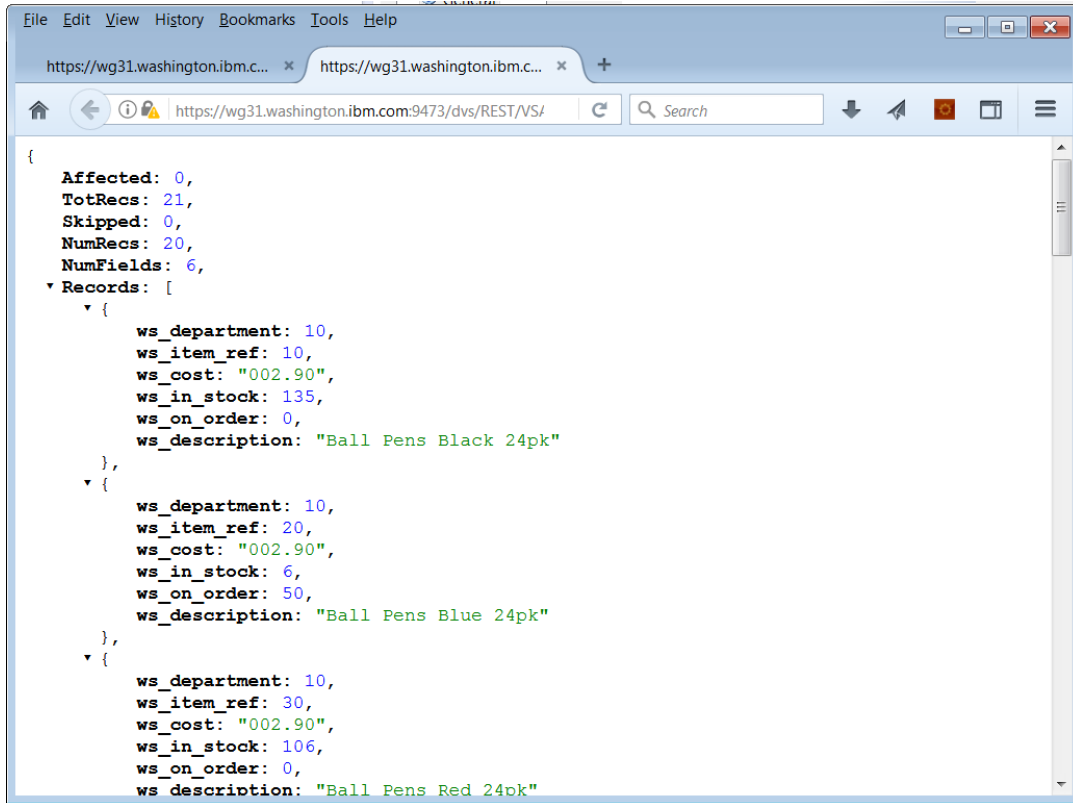


3. These z/OS Connect EE services can now be tested using the DVM Data Manager Studio. Select the *retrieveAll* operation and right mouse button click. Select the *z/OS Connect REST Interface* option then the *Execute Query* option.

4. This pop-up window should be displayed.

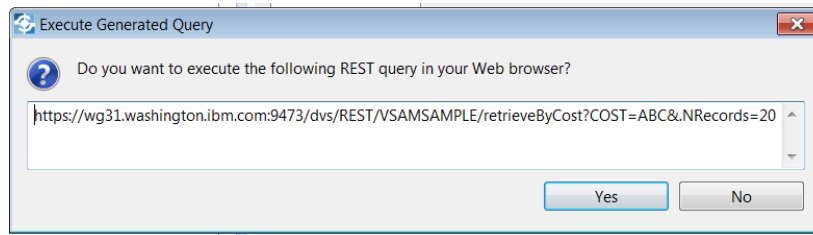


\_\_\_ 5. Click **Yes** and a web browser tab should be opened with results like these.

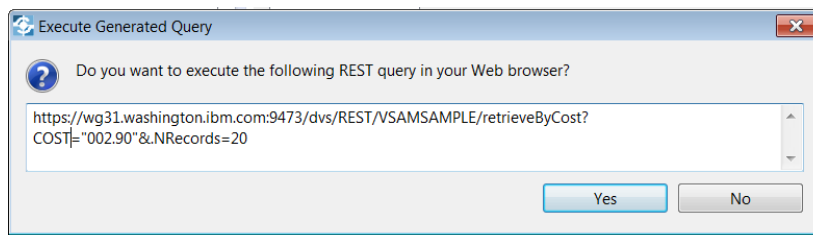


\_\_\_ 6. Select the *retrieveAll* operation and right mouse button click. Select the *z/OS Connect REST Interface* option then the *Execute Query* option.

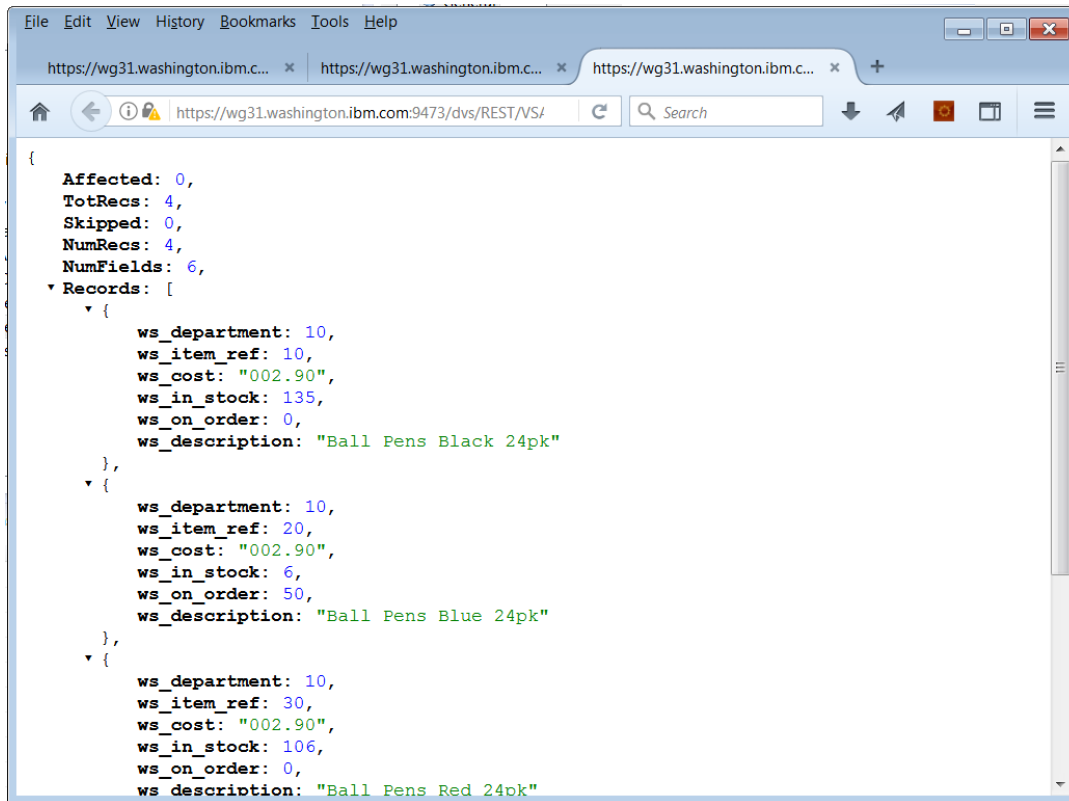
\_\_\_ 7. This pop-up window should be displayed.



\_\_\_ 8. Change the string *COST=ABC* to *COST="002.90"*

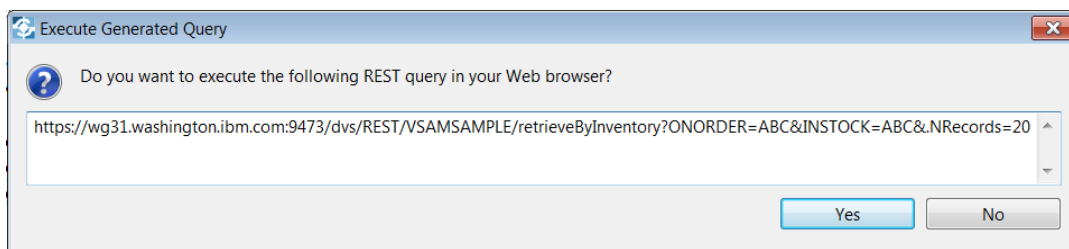


9. Click **Yes** and a web browser tab should be opened with results like these.

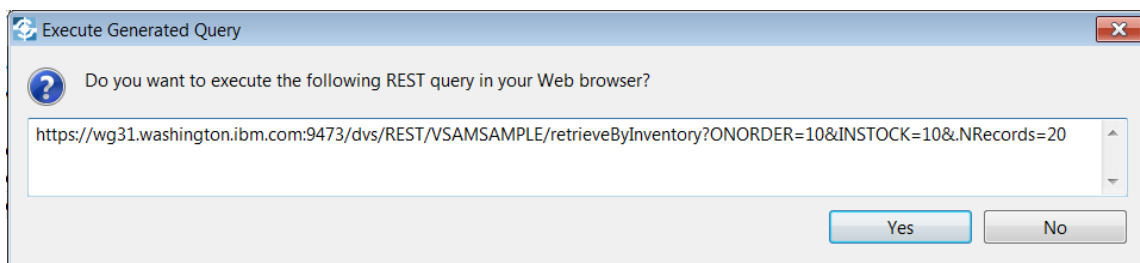


10. Select the *retrieveByInventory* operation and right mouse button click. Select the *z/OS Connect REST Interface* option then the *Execute Query* option.

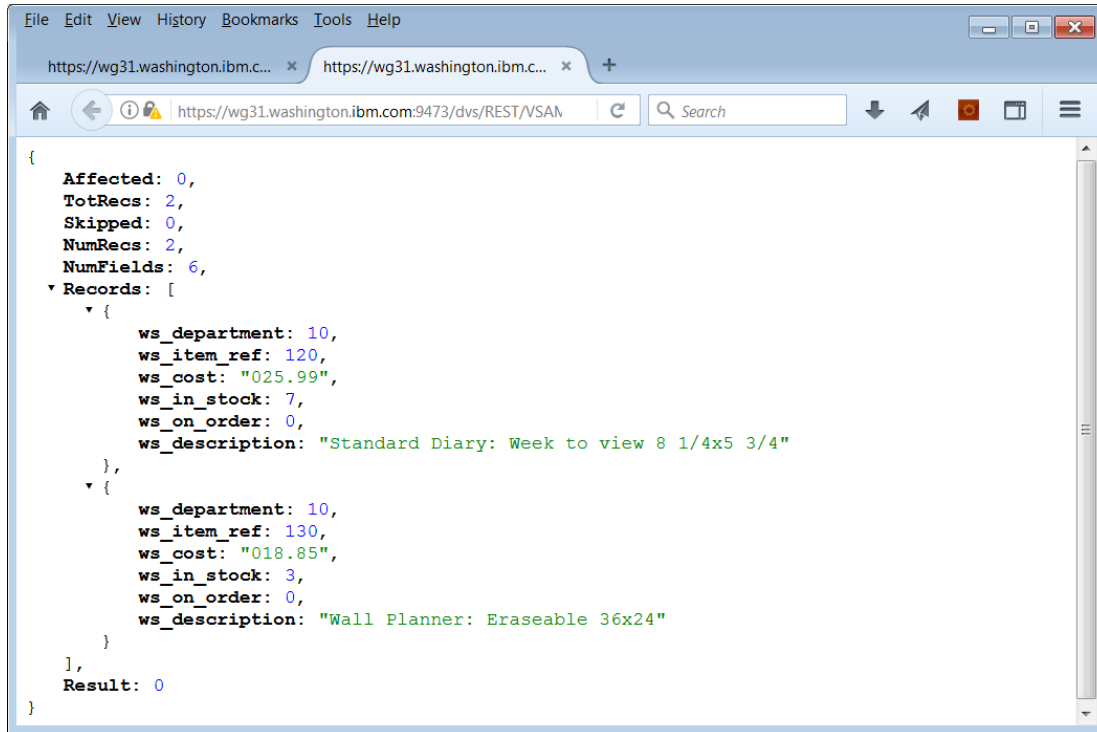
11. This pop-up window should be displayed.



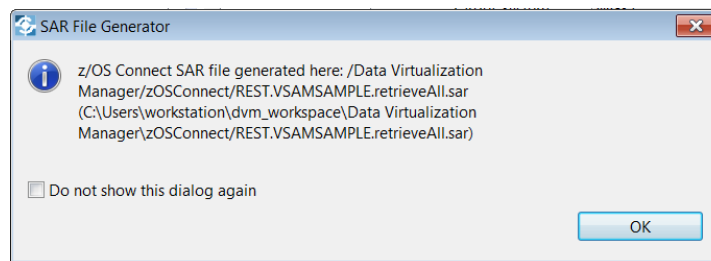
12. Change the string *ONORDER=ABC* to ***ONORDER=10*** and *INSTOCK=ABC* to ***INSTOCK=10***



13. Click **Yes** and a web browser tab should be opened with results like these.



14. Finally, the Service Archive (SAR) files need to be exported from the DVM Studio for use in the z/OS Connect EE API Editor. Select each operation in turn and right mouse button click. Select the *z/OS Connect REST Interface* option then the *Generate SAR File(s)* option. Click **OK** to continue.



This exports the SAR files to a subdirectory in the DVM Toolkit's workspace directory, e.g *C:\Users\workstation\dvm\_workspace\Data Virtualization Manager\zOSConnect*. This directory will be referenced in a latter section of this exercise.



## Create z/OS Connect EE APIs

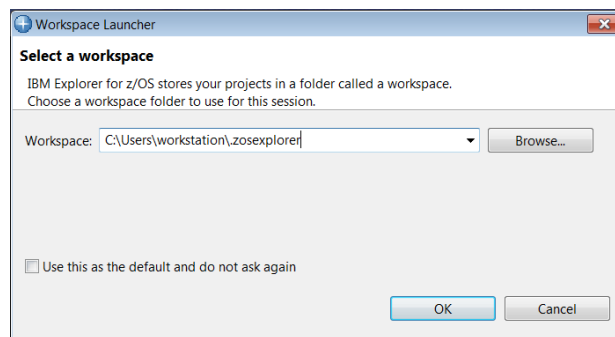
### Connect to a z/OS Connect EE Server

Begin by establishing a connection to DVM z/OS Connect server from IBM z/OS Explorer.

1. On the workstation desktop, locate the *z/OS Explorer* icon and double click on it to open the Explorer.

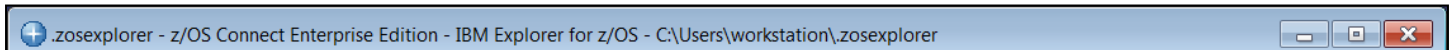
**Tech-Tip:** Windows desktop tools can be opened either by double clicking the icon or by selecting the icon and right mouse button clicking and then selecting the *Open* option.

2. You will be prompted for a workspace:



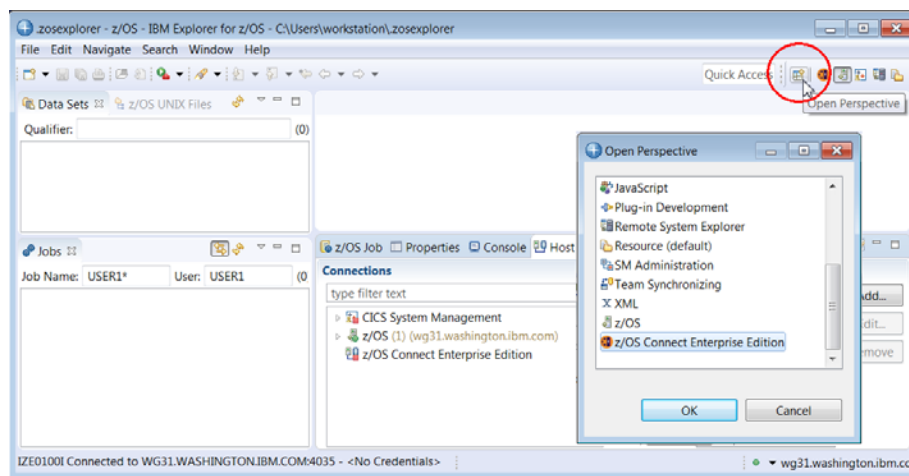
Take the default value by clicking **OK**.

3. The Explorer should open in the *z/OS Connect Enterprise Edition* perspective. Verify this by looking in the upper left corner. You should see:



N.B. If a *Welcome* screen is displayed then click the white X beside *Welcome* to close this view.

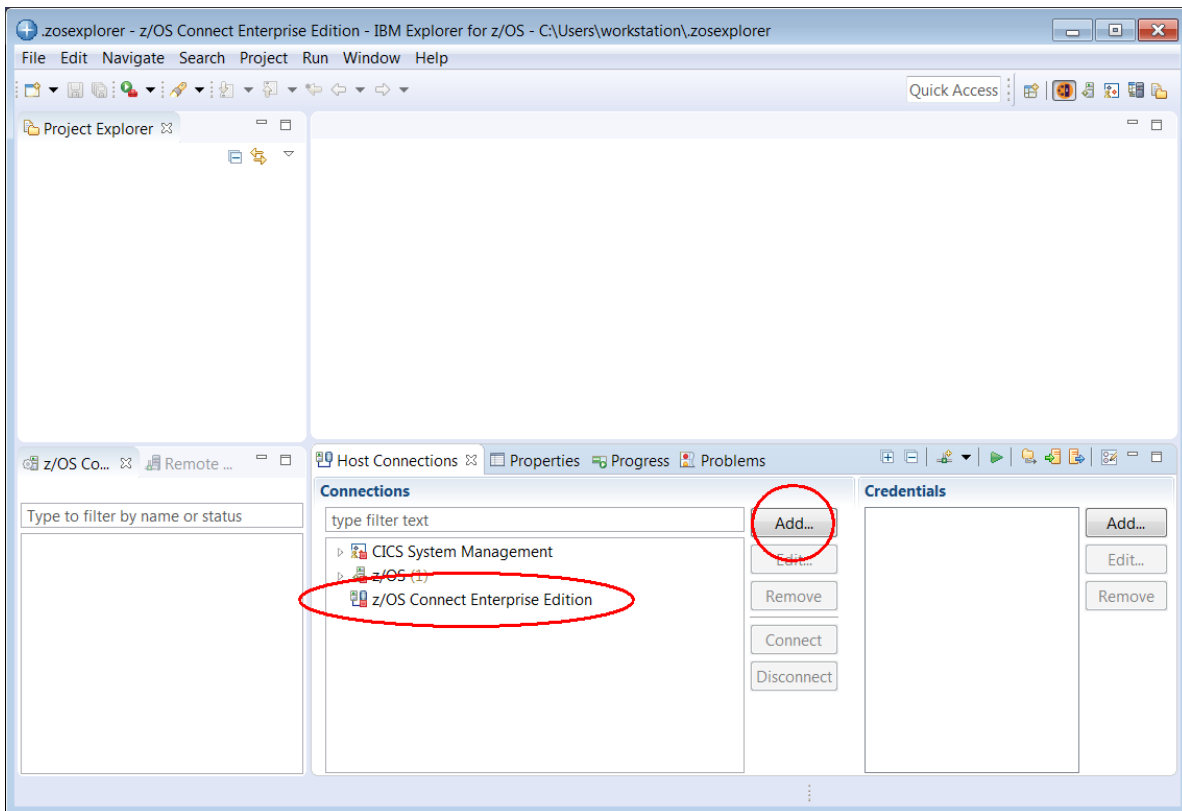
4. If the current perspective is not *z/OS Connect Enterprise Edition*, select the *Open Perspective* icon on the top right side to display the list of available perspectives, see below. Select **z/OS Connect Enterprise Edition** and click the **OK** button to switch to this perspective.



5. To add a

connection to the

z/OS Connect Server select *z/OS Connect Enterprise Edition* connection in the *Host connections* tab in the lower view and then click the **Add** button.



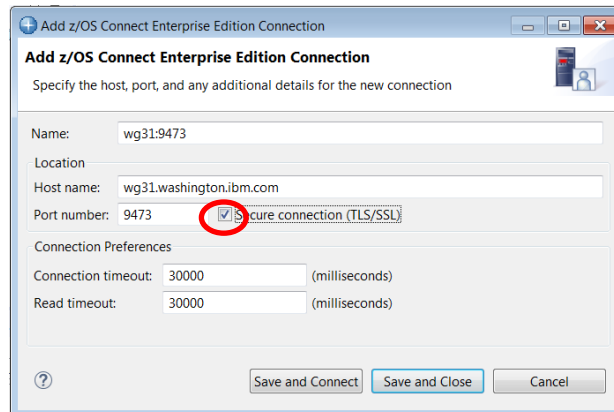
**Tech-Tip:** Eclipse based development tools like z/OS Explorer; provide a graphical interface consisting of multiple views within a single window.

A view is an area in the window dedicated to providing a specific tool or function. For example, in the window above, *Host Connections* and *Project Explorer* are views that use different areas of the window for displaying information. At bottom on the right there is a single area for displaying the contents of four views stacked together (commonly called a *stacked views*), *z/OS Host Connections*, *Properties*, *Progress* and *Problems*. In a stacked view, the contents of each view can be displayed by clicking on the view tab (the name of the view).

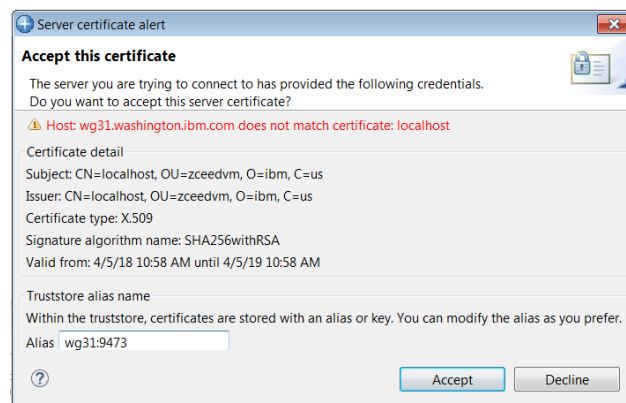
At any time, a specific view can be enlarged to fill the entire window by double clicking in the view's title bar. Double clicking in the view's title bar will be restored the original arrangement. If a z/OS Explorer view is closed or otherwise disappears, the original arrangement can be restored by selecting Windows → Reset Perspective in the window's tool bar.

Eclipse based tools also can display multiple views based on the current role of the user. In this context, a window is known as a perspective. The contents (or views) of a perspective are based on the role the user, i.e., developer or administrator.

6. In the pop-up list displayed select *z/OS Connect Enterprise Edition* and on the *Add z/OS Connect Enterprise Edition Connection* window enter **wg31.washington.ibm.com** for the *Host name*, 9473 for the *Port Number*, check the box for *Secure connection (TLS/SSL)* and then click the **Save and Connect** button.



7. On the *z/OS Connect Enterprise Edition – User ID* required screen create new credentials for a *User ID* of **USER1** and for *Password or Passphrase* enter USER's password. Click **OK** to continue.
8. Click the **Accept** button on the *Server certificate alert – Accept this certificate* screen. You may be presented with another prompt for a userid and password, enter **USER1** and USER1's password again.



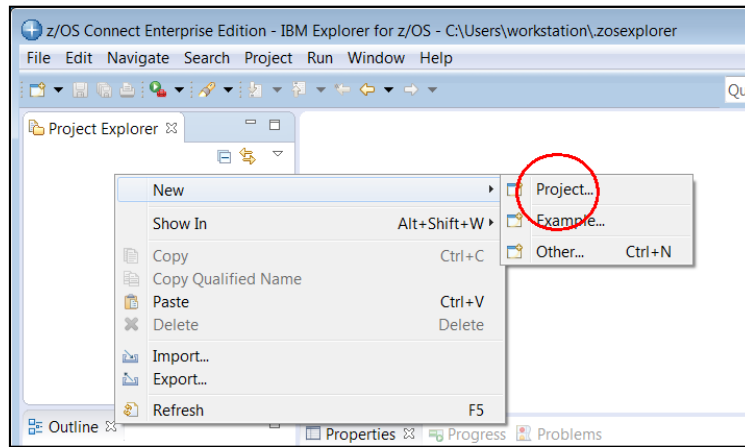
9. The status icon beside **wg31:9473** should now be a green circle with a lock. This shows that a secure connection has been established between the z/OS Explorer and the z/OS Connect server. A red box indicates that no connection exists.
10. A connection to the remote z/OS system was previously added. In the *Host Connection* view expand *z/OS Remote System* under *z/OS* and select **wg31.washington.ibm.com**. If the connection is not active the **Connect** button will be enabled. Click the **Connect** button and this will establish a session to the z/OS system. This step is required when submitting job for execution and viewing the output of these jobs later in this exercise

## Summary

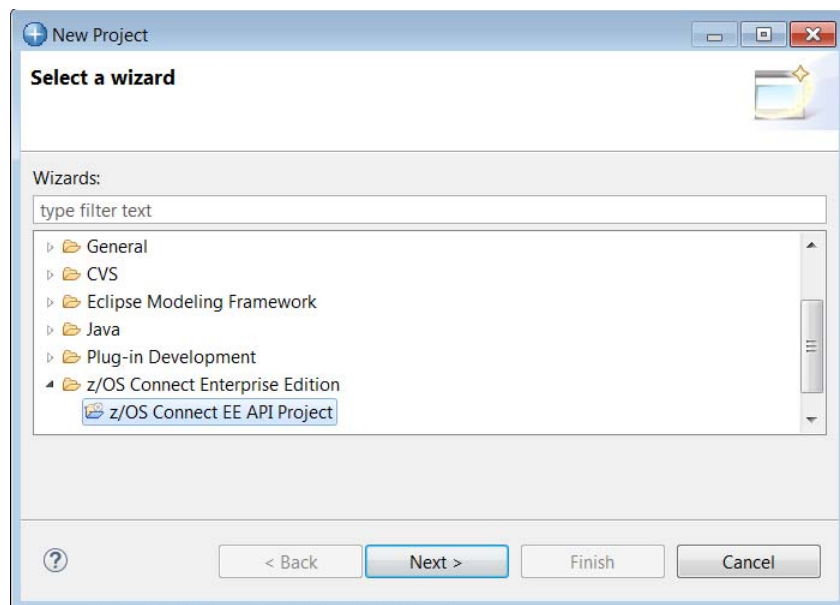
The next step is the creation of the service and the composing and deployment of the API and then the testing of the API functions.

## Create the DVM API Project

1. In the *z/OS Connect Enterprise Edition* perspective of the z/OS Explorer create a new API project by clicking the right mouse button and selecting **New** → **Project**:



2. In the *New Project* window, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE API Project* and then click the **Next** button.



3. Enter **DVMAPI** for the *Project name*. Be sure the *API name* is set to **dvm** and the *Base path* is set to **/dvm**. Click **Finish** to continue.

The screenshot shows a 'New Project' dialog box. The title bar says 'New Project'. The main title is 'z/OS Connect EE API Project'. Below it, it says 'Create a new z/OS Connect EE API project.' There are four input fields: 'Project name' with 'DVMAPI', 'API name' with 'dvm', 'Base path' with '/dvm', and 'Description' which is empty. At the bottom right, there are 'Finish' and 'Cancel' buttons.

**Important:** The values are somewhat arbitrary, but they do relate to later tasks. If you use the values and cases as supplied then the subsequent commands and the use of subsequent URLs will work seamlessly.

4. You should now see something like the view below. The view may need to be adjusted by dragging the view boundary lines.

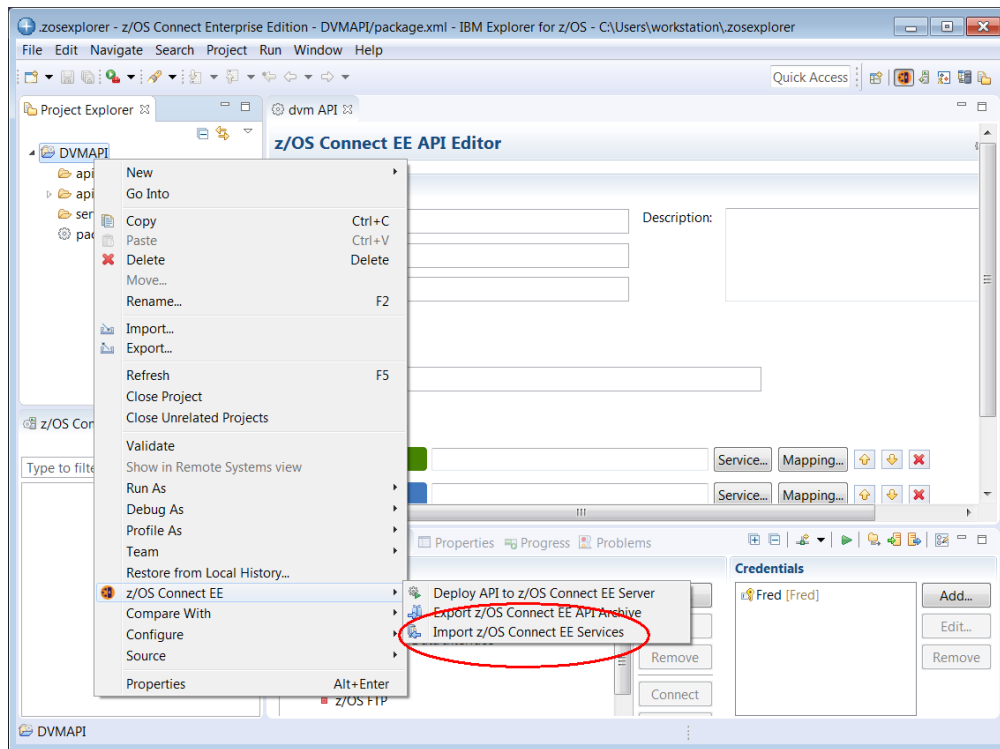
The screenshot shows the 'z/OS Connect EE API Editor' window. The title bar says 'dvm API'. The main title is 'z/OS Connect EE API Editor'. There's a section 'Describe your API' with fields for 'Name' (dvm), 'Base path' (/dvm), and 'Version' (1.0.0). Below this is a 'Path' section with a field showing '/newPath1'. At the bottom, there's a 'Methods' section with a list of HTTP methods: POST, GET, PUT, and DELETE. Each method has a 'Service...' button and a 'Mapping...' button, along with some icons.

## Summary

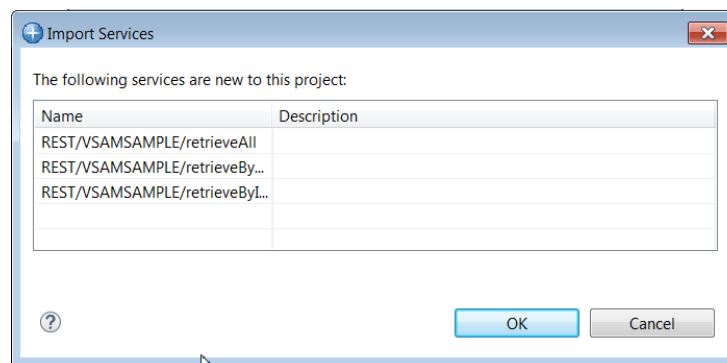
This created the basic framework for the API project in the API editor

## Import the SAR files generated by the DVM Studio

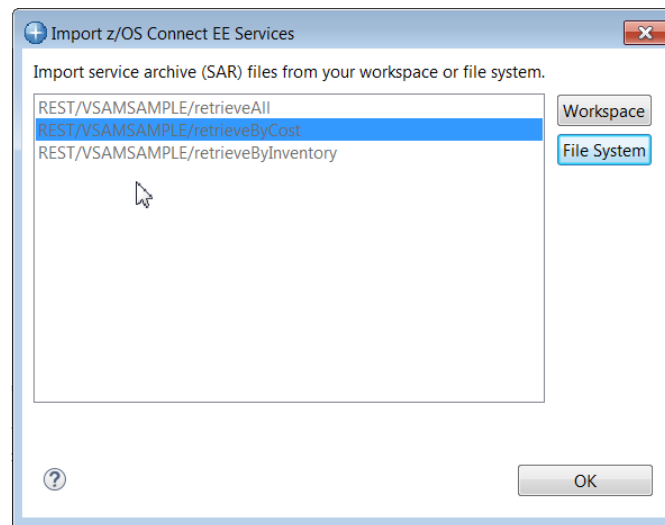
1. In the z/OS Explorer in the z/OS Connect Enterprise Edition perspective in the the *Project Explorer* view (upper left), right-click on the *DVMAPI* project, then select *z/OS Connect EE* and then *Import z/OS Connect EE Services* (see below):



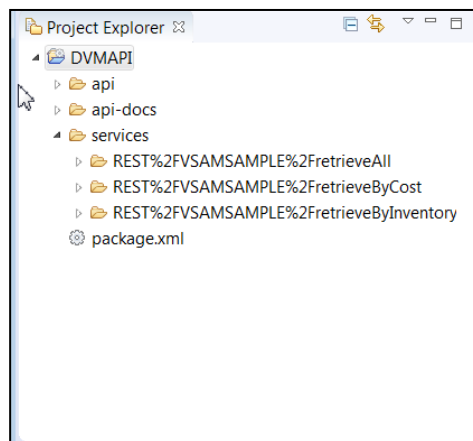
2. In the *Import z/OS Connect EE Services* window click on the **File System** button and navigate to directory *C:\Users\workstation\dvm\_workspace\Data Virtualization Manager\zOSConnect*. Select the three SAR files and click on the **Open** button:



3. The three service archive files should appear in the *Import z/OS Connect EE Services* window. Click the **OK** button to import them into the workspace.



4. In the *Project Explorer* view (upper left), expand the *services* folder to see the imported service:



## Summary

The SAR files created earlier were imported into the editor. That provides the editor with information about the underlying services and the JSON schemas.

## Compose an API for DVM Rest Services

\_\_\_1. Start by entering a *Path* of **/vsamsample** in the *z/OS Connect EE API Editor* view as shown below:

The screenshot shows the 'z/OS Connect EE API Editor' window. The title bar indicates the current API is '\*dvm API'. The main area is titled 'Describe your API' and contains the following fields:

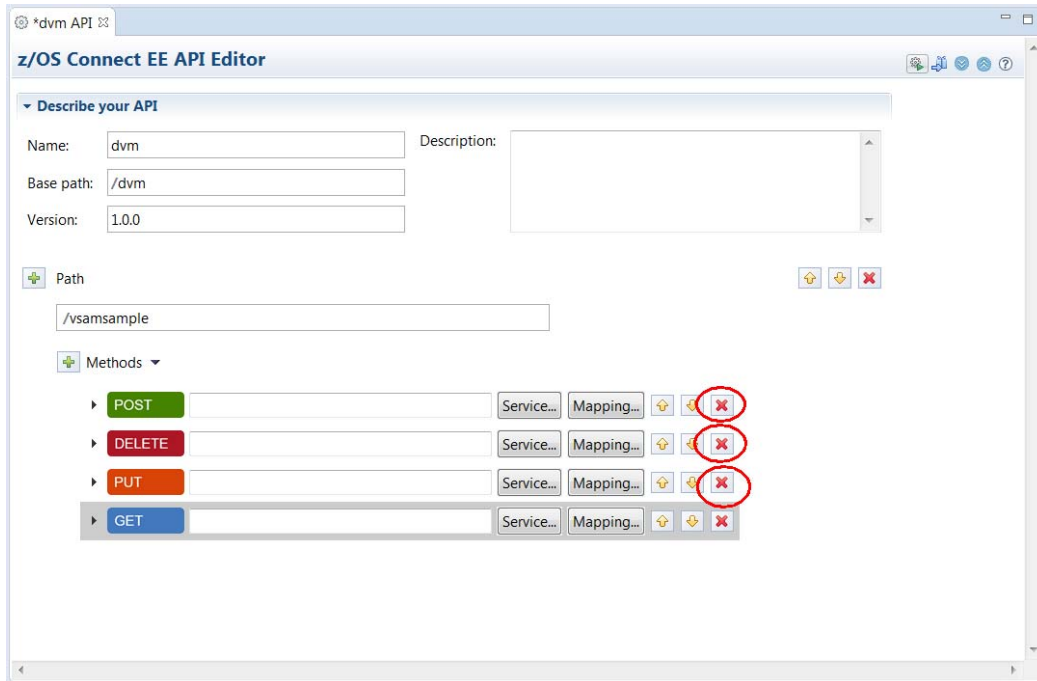
- Name:** dvm
- Description:** (empty text area)
- Base path:** /dvm
- Version:** 1.0.0

Below these fields, there are two expandable sections:

- Path:** Contains a text input field with the value '/vsamsample'.
- Methods:** A list of HTTP methods with corresponding 'Service...' and 'Mapping...' buttons and icons for each:
  - POST:** Green button, followed by 'Service...' and 'Mapping...' buttons and up/down/delete icons.
  - DELETE:** Red button, followed by 'Service...' and 'Mapping...' buttons and up/down/delete icons.
  - PUT:** Orange button, followed by 'Service...' and 'Mapping...' buttons and up/down/delete icons.
  - GET:** Blue button, followed by 'Service...' and 'Mapping...' buttons and up/down/delete icons.



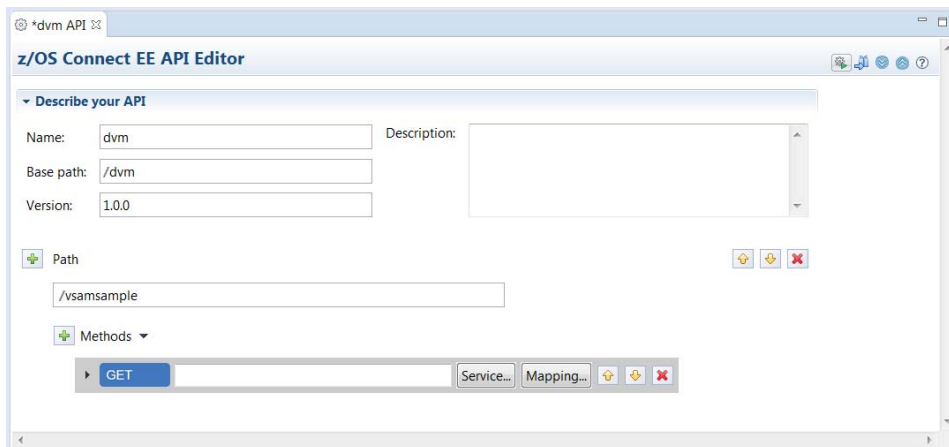
2. For the initial *DVM API* when no path or query parameter is present the supported HTTP methods will only be **GET**. Remove the **DELETE**, **POST** and **PUT** methods by clicking the X icon to the right of each method.



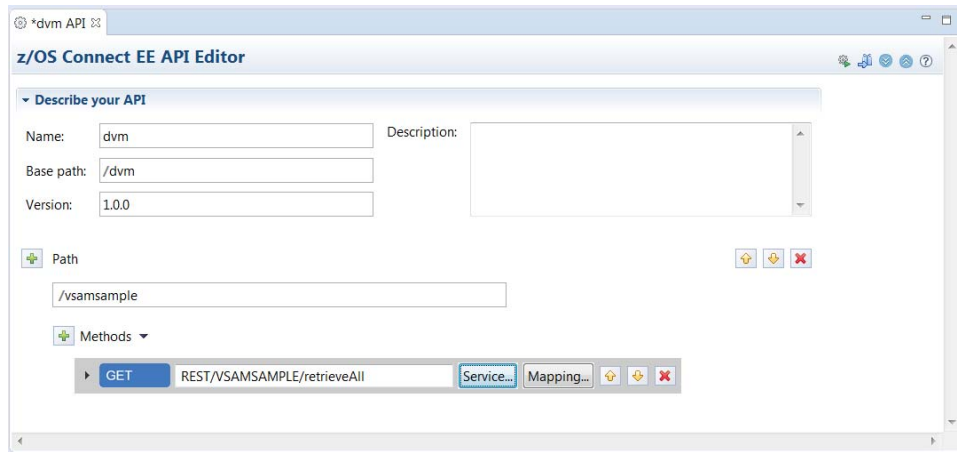
**Note:** The */vsamsample* path element again is somewhat arbitrary, but is used to distinguish this request from other requests that may be configured in the same API.

The full URL to invoke the methods for this path of the API will be <https://hostname:port/dvm/vsamsample>

That should leave you with the **GET** method.



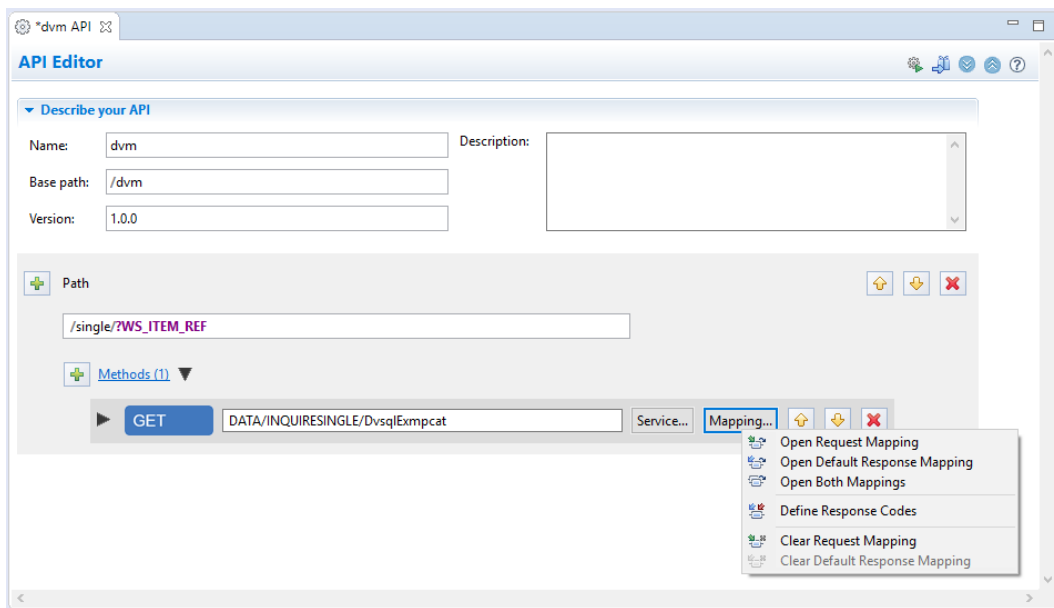
3. Click on the **Service** button to the right of the **GET** method. Then select the *REST/VSAMSAMPLE/retrieveAll* service from the list of service archive files and click **OK**. This will populate the field to the right of the method.



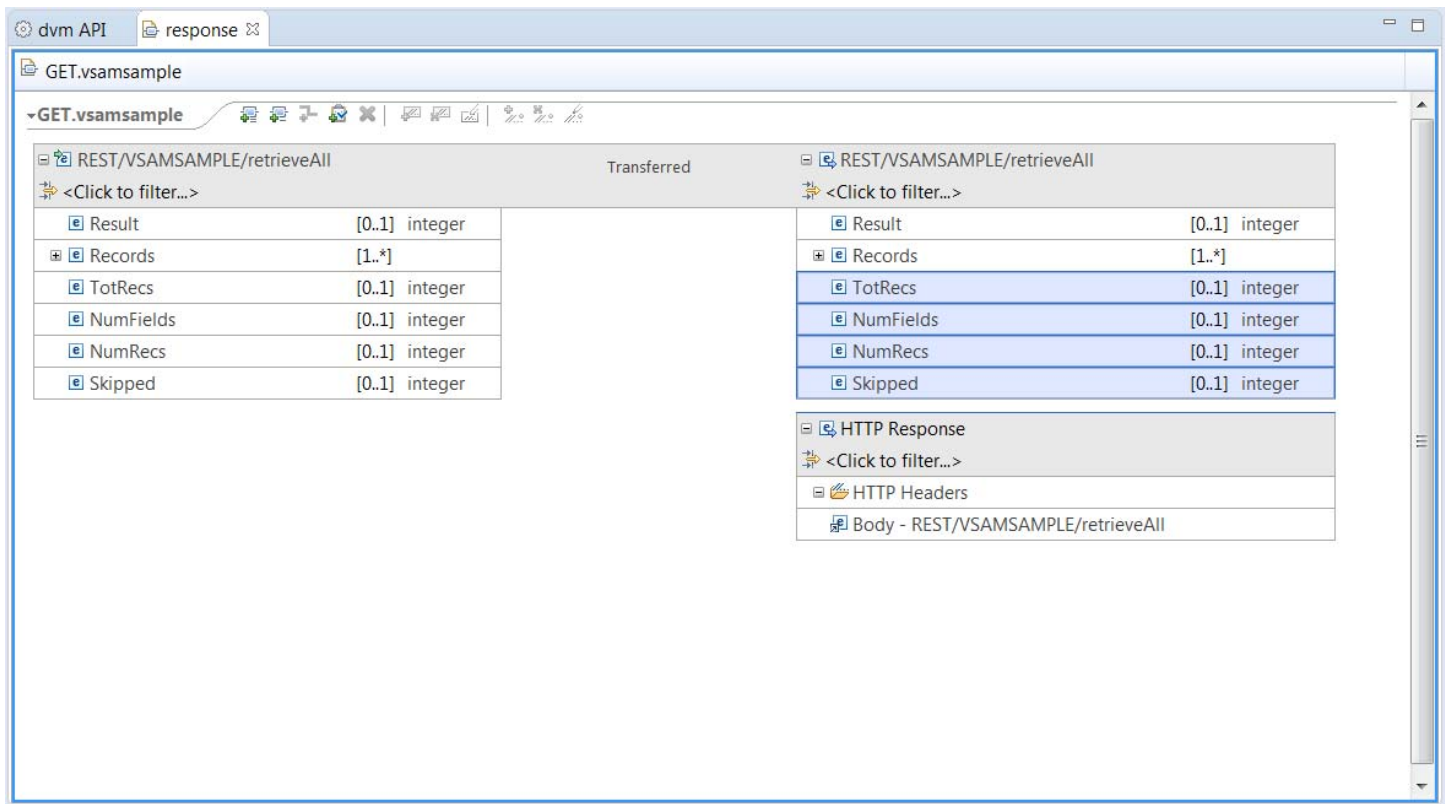
4. Save the changes so far by using the key sequence **Ctrl-S**.

**Tech-Tip:** If any change is made in any edit view an asterisk (\*) will appear before the name of the artifact in the view tab, e.g. *\*package.xml*. Changes can be saved at any time by using the **Ctrl-S** key sequence.

5. Next, click on the **Mapping** button beside the **GET** method and then select *Open Default Response Mapping*:

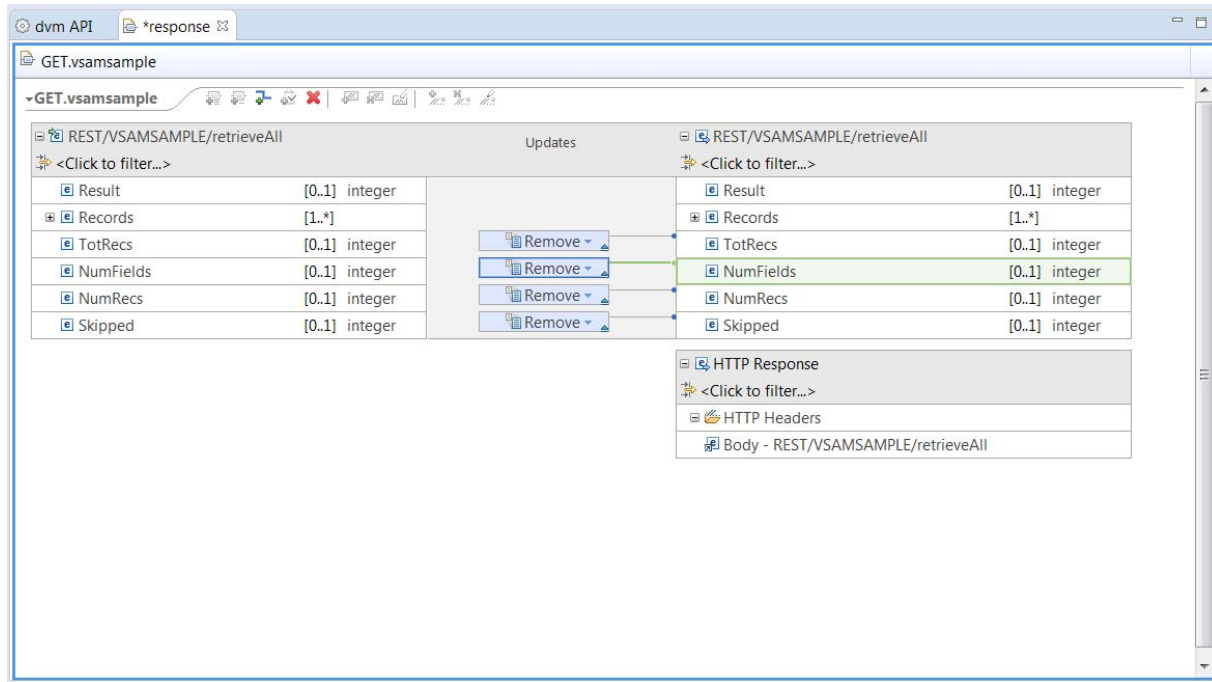


- \_\_\_ 6. Use the left mouse button and draw a dotted line box that **fully** includes the *TotRecs*, *NumFields*, *NumRecs* and *Skipped* fields. When you release the button, these fields should be selected (the background should be blue).

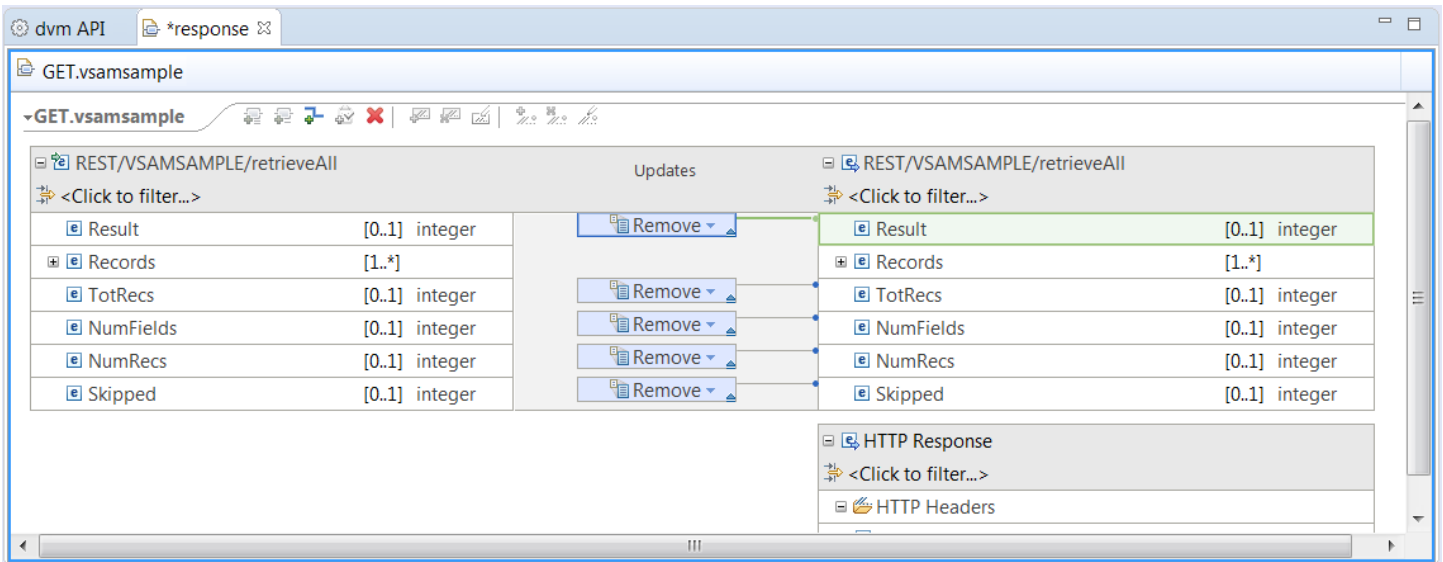


- \_\_\_ 7. Right mouse button click on any of the selected fields and select the *Add Remove transform* from the list of options.

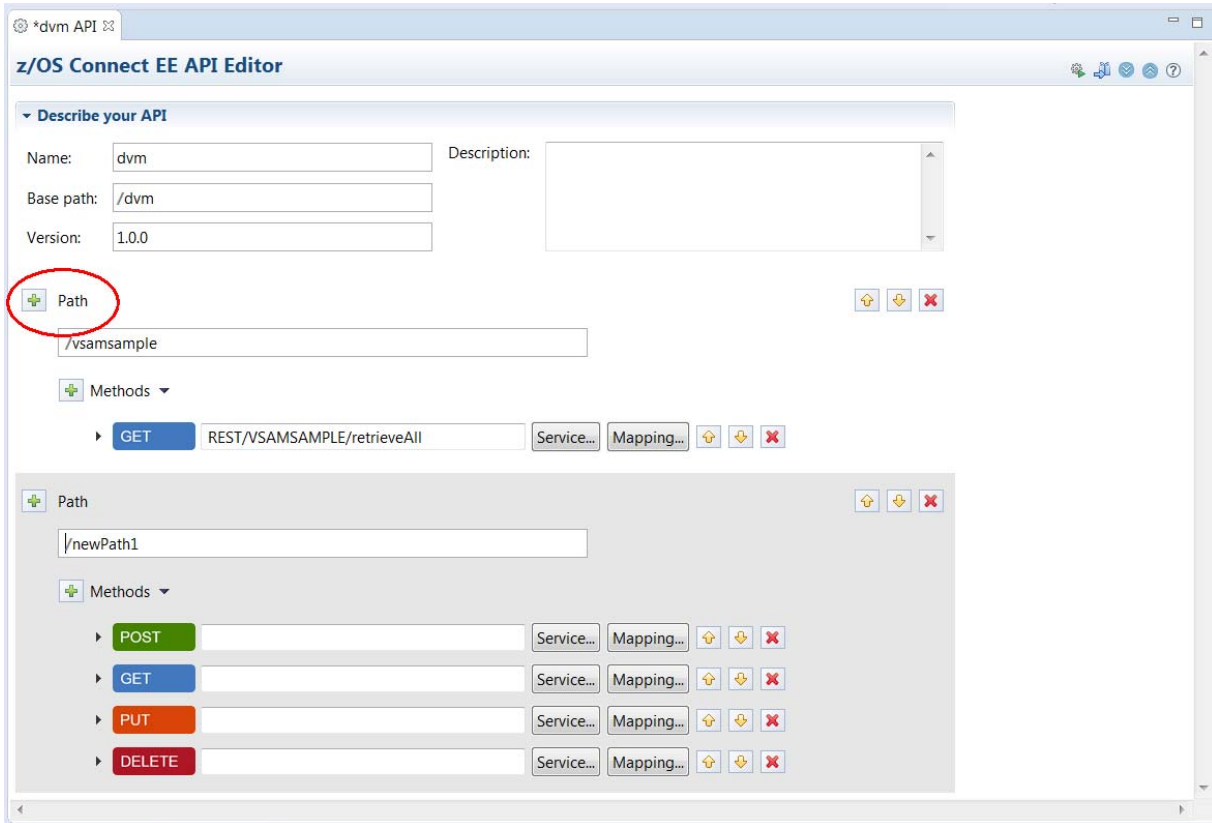
7. This action generates multiple “remove” requests (see below) for the selected fields. These fields are not required so they will be removed from the response.



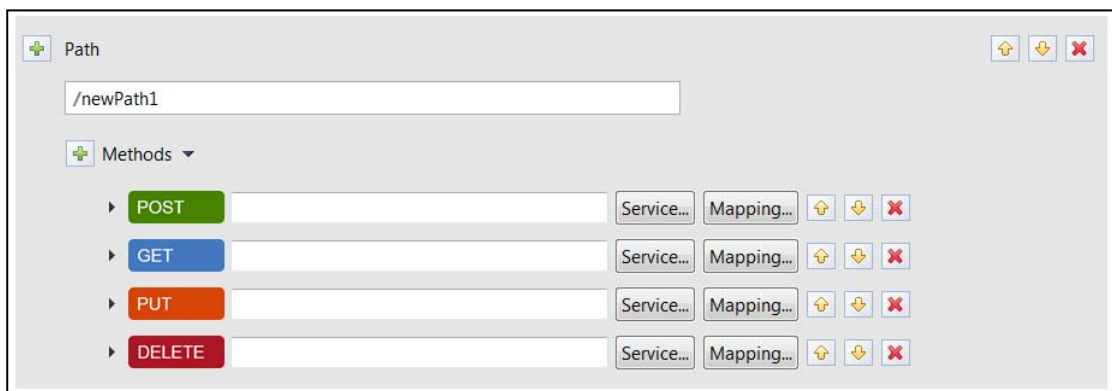
8. Select the *Result* field and remove it from the response. If not expanded already, expand the *Records* structure and you should see the ‘columns’ that will be displayed in the response.



- \_\_\_ 9. Use the **Ctrl-S** key sequence to save all changes and close the *GET.vsamsample* view.
- \_\_\_ 10. Next, we want to add a Path for a **GET** method for the *retrieveByCost* service. Click the plus icon beside Path on the z/OS Connect EE API Editor view to add another path to the API.



The result is another full set of methods for the new *PATH*.



- \_\_\_ 11. Enter a path value of **/vsamsample/items?cost** and remove the *POST*, *PUT* and *DELETE* methods.

**Note:** The */vsamsample/items* path element again is somewhat arbitrary, but is used to distinguish this request from other requests that may be configured in the same API.

The *{cost}* element is a path parameter in the URL that will be used to provide the key of the record for get requests.

The full URL to invoke the methods for this path of the API will be

<https://hostname:port/dvm/vsamsample/items/#####>

where ##### is the cost to be used to select a record in the VSAM data set

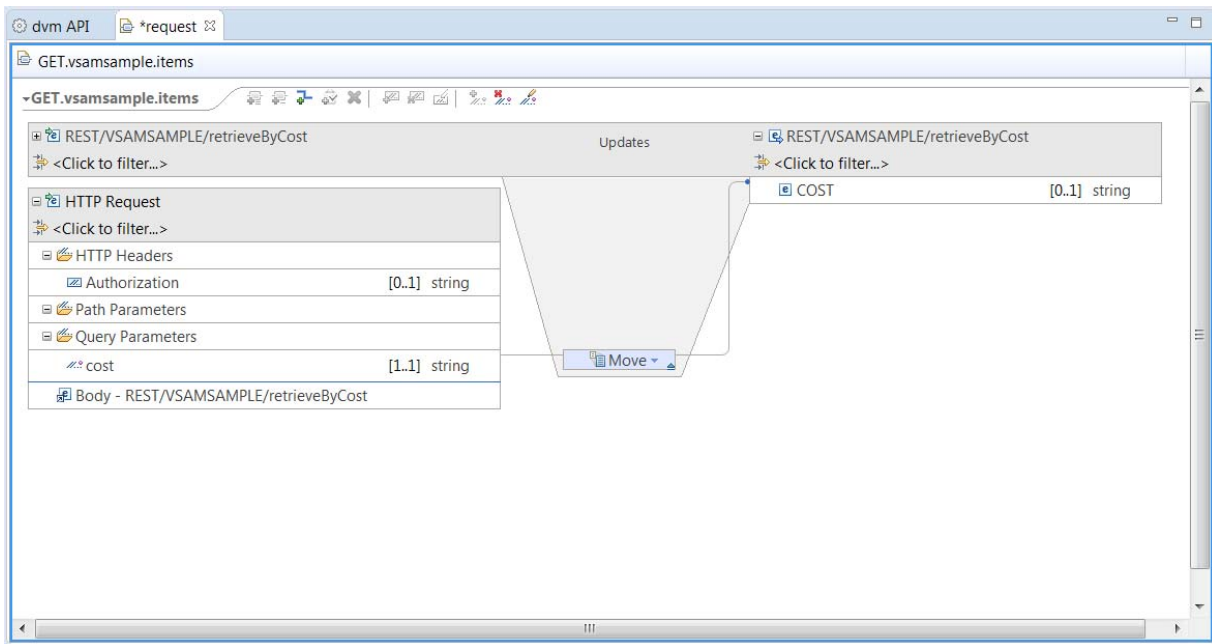
**Tech-Tip:** Additional *Paths* can be added by clicking the + icon beside *Path* and additional *Methods* can be added by clicking the + icon beside *Methods*.

- \_\_\_ 12. Click the **Service** button beside **GET** and select the *REST/VSAMSAMPLE/retrieveByCost* service.

- \_\_\_ 13. Save the changes by using the key sequence **Ctrl-S**.

- \_\_\_ 14. Click on *Mapping* → *Open request mapping*.

- \_\_\_ 15. Use the left mouse button to drag the *cost* query parameter from the left-hand side to the *COST* field on the right side.

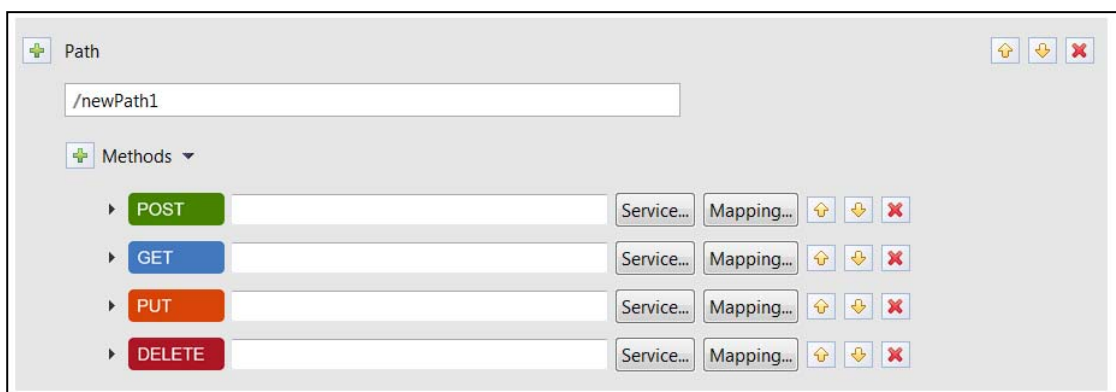


- \_\_\_ 16. Save the the changes using the key sequence **Ctrl-S**.

**Note:** We are not modifying the response message by removing fields. This could be easily done as shown earlier.

- \_\_\_ 17. Next, we want to add a Path for a **GET** method for the *retrieveByInventory*. Click the plus icon beside Path on the z/OS Connect EE API Editor view to add another path to the API.

The result is another full set of methods for the new *PATH*.



- \_\_\_ 18. Enter a path value of **/vsamsample/items/{onOrder}?inStock** and remove the *POST*, *PUT* and *DELETE* methods.

The screenshot shows a configuration window with two main sections: 'Path' and 'Methods'. The 'Path' section has a text input field containing '/vsamsample/items/{onOrder}?inStock'. The 'Methods' section has a dropdown menu with 'GET' selected. There are also buttons for 'Service...' and 'Mapping...' and some navigation icons.

**Note:** The */vsamsample/items* path element again is somewhat arbitrary, but is used to distinguish this request from other requests that may be configured in the same API.

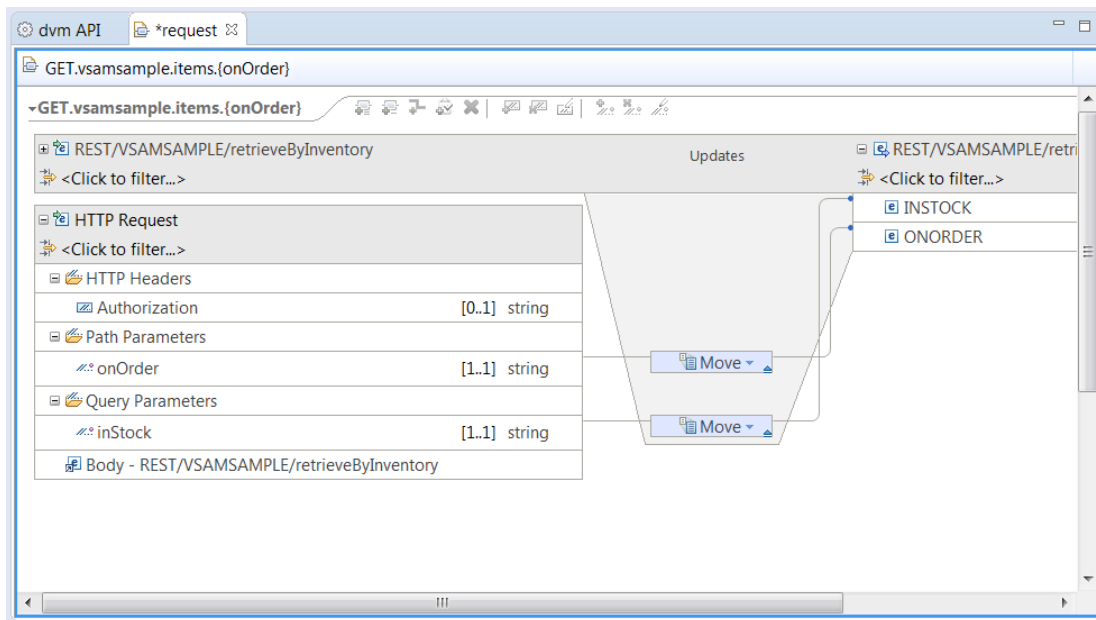
The *{order}* element is a path parameter in the URL that will be used to provide the number of items on order variable while *?inStock* is a query parameter that will be used to provide the number of items in stock for requests.

The full URL to invoke the methods for this path of the API will be  
<https://hostname:port/dvm/vsamsample/items/#####/instock=#####>

- \_\_\_ 19. Click the **Service** button beside **GET** and select the *REST/VSAMSAMPLE/retrieveByInventory* service.
- \_\_\_ 20. Save the changes by using the key sequence **Ctrl-S**.
- \_\_\_ 21. Click on *Mapping* → *Open request mapping*.

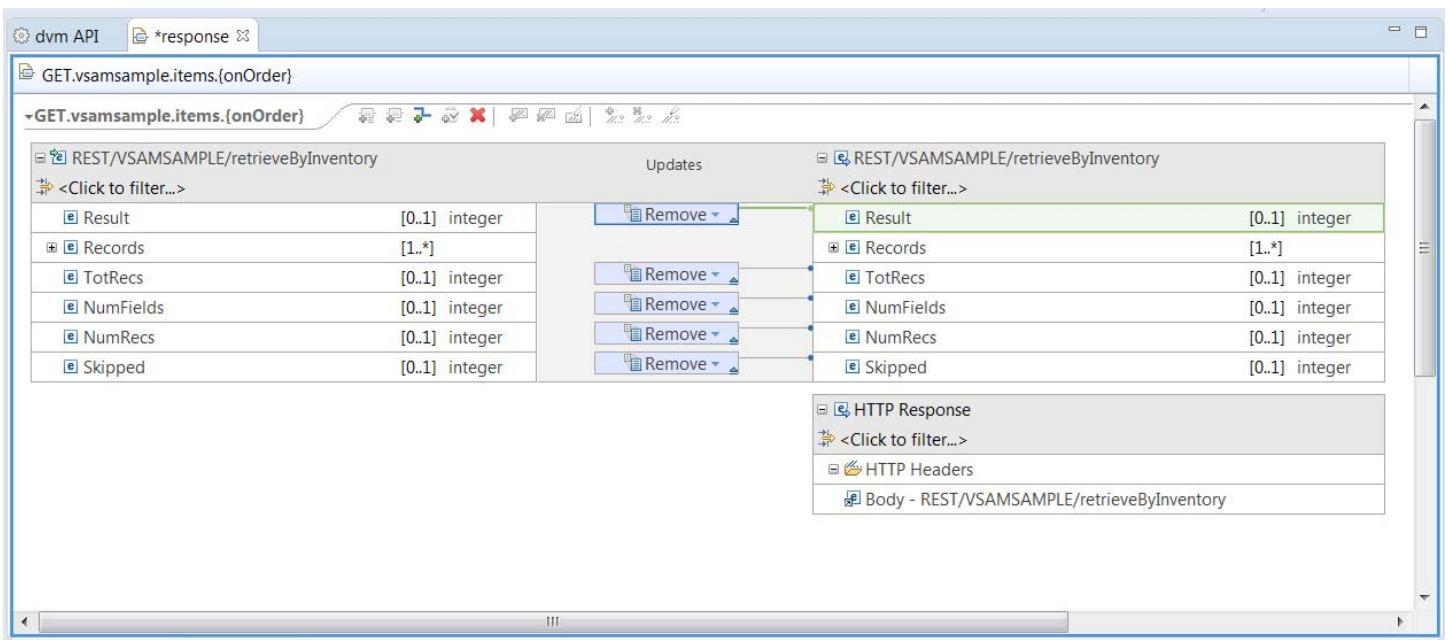


22. Use the left mouse button to drag the *inStock* query parameter from the left-hand side to the *INSTOCK* field on the right side. Repeat this to drag the *onOrder* path parameter from the left-hand side to the *ONORDER* field on the right side.



23. Save the the changes using the key sequence **Ctrl-S**.

24. Use Steps 5 through 8 to remove the fields from the response message for this request.



25. Close any oper *request* or *response* mapping tabs.

26. Next, we want to add another Path for a **GET** method for the *retrieveByInventory*. This DVM service will be reused to provide another API. This API request will return the list of items in the inventory where the number of in stock items is below a certain level. This number of items on order will be set to the maximum value of this field. Click the plus icon beside Path on the z/OS Connect EE API Editor view to add another path to the API.

The result is another full set of methods for the new *PATH*.

The screenshot displays the z/OS Connect EE API Editor interface. At the top, there is a 'Path' section with a plus icon and a text input field containing '/newPath1'. Below this, there is a 'Methods' section with a plus icon and a dropdown arrow. Under the 'Methods' section, there are four rows of method definitions:

Method	Service...	Mapping...	Up	Down	Delete
POST					
GET					
PUT					
DELETE					

- \_\_\_ 27. Enter a path value of **/vsamsample/items/lowStock/?inStock** and remove the *POST*, *PUT* and *DELETE* methods.

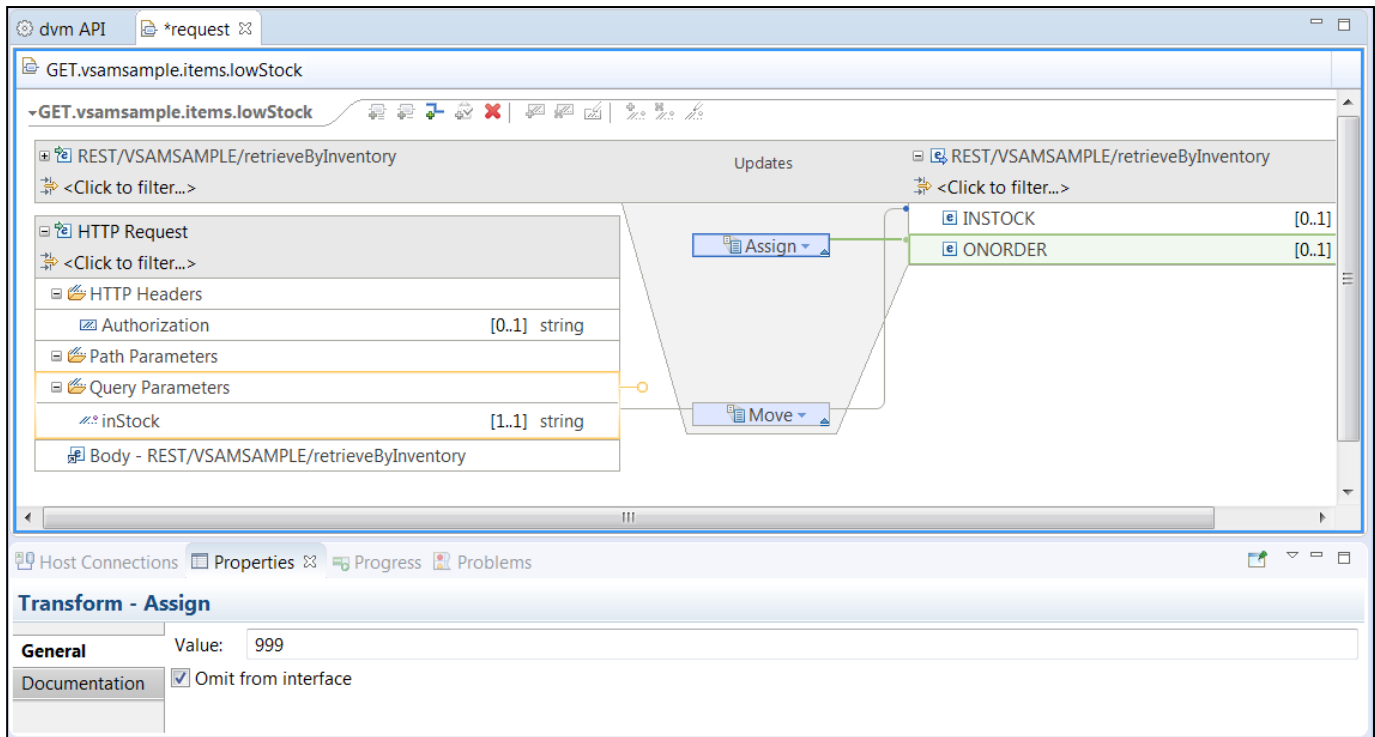
**Note:** The */vsamsample/items/lowStock* path element again is somewhat arbitrary, but is used to distinguish this request from other requests that may be configured in the same API.

?inStock is a query parameter that will be used to provide the number of items in stock for requests. The ONORDER field in the request message will be set to the maximum value by the API developer and omitted from the interface.

The full URL to invoke the methods for this path of the API will be  
<https://hostname:port/dvm/vsamsample/items/lowStock/instock=#####>

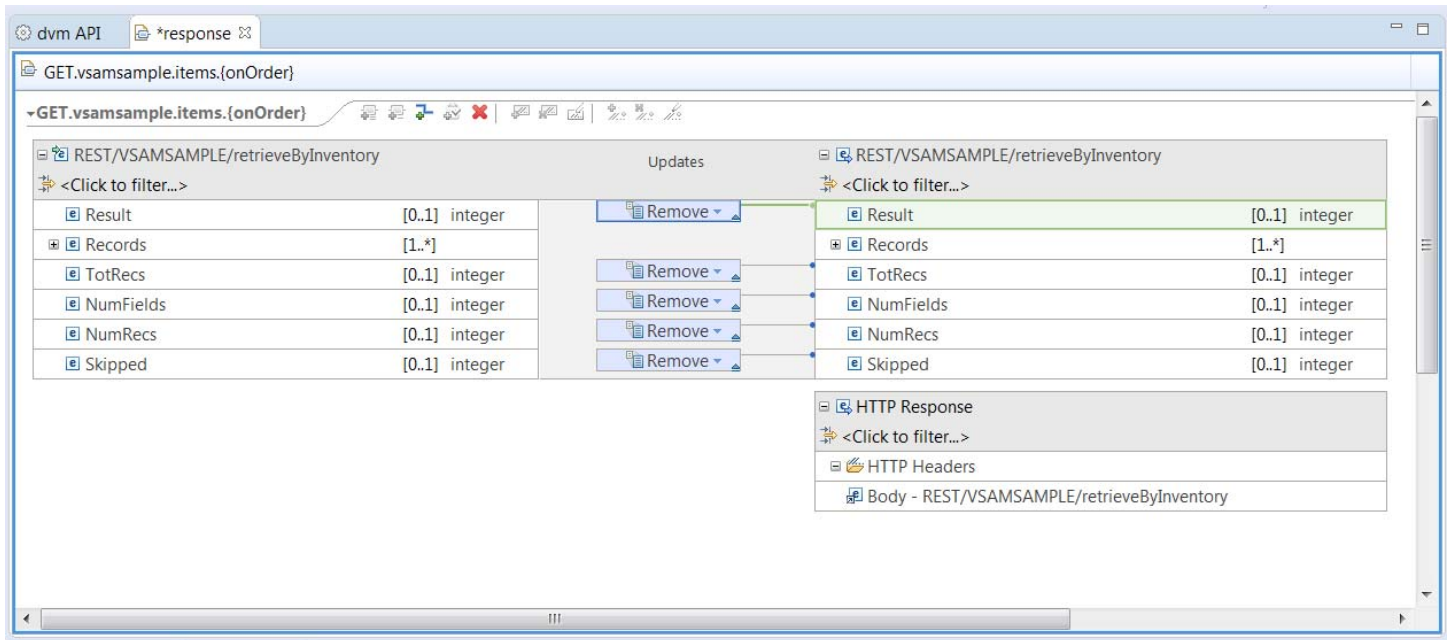
- \_\_\_ 28. Click the **Service** button beside **GET** and select the *REST/VSAMSAMPLE/retrieveByInventory* service.
- \_\_\_ 29. Save the changes by using the key sequence **Ctrl-S**.
- \_\_\_ 30. Click on *Mapping* → *Open request mapping*.

- \_\_\_31. Use the left mouse button to drag the *inStock* query parameter from the left-hand side to the *INSTOCK* field on the right side. Select the *ONORDER* field on the right-hand side and right mouse button click. Select the *Add Assign transform* option. In the lower view select the *Properties* tab and enter **999** in the area beside *Value*.



- \_\_\_32. Save the the changes using the key sequence **Ctrl-S**.

33. Use Steps 5 through 8 to remove the fields from the response message for this request.



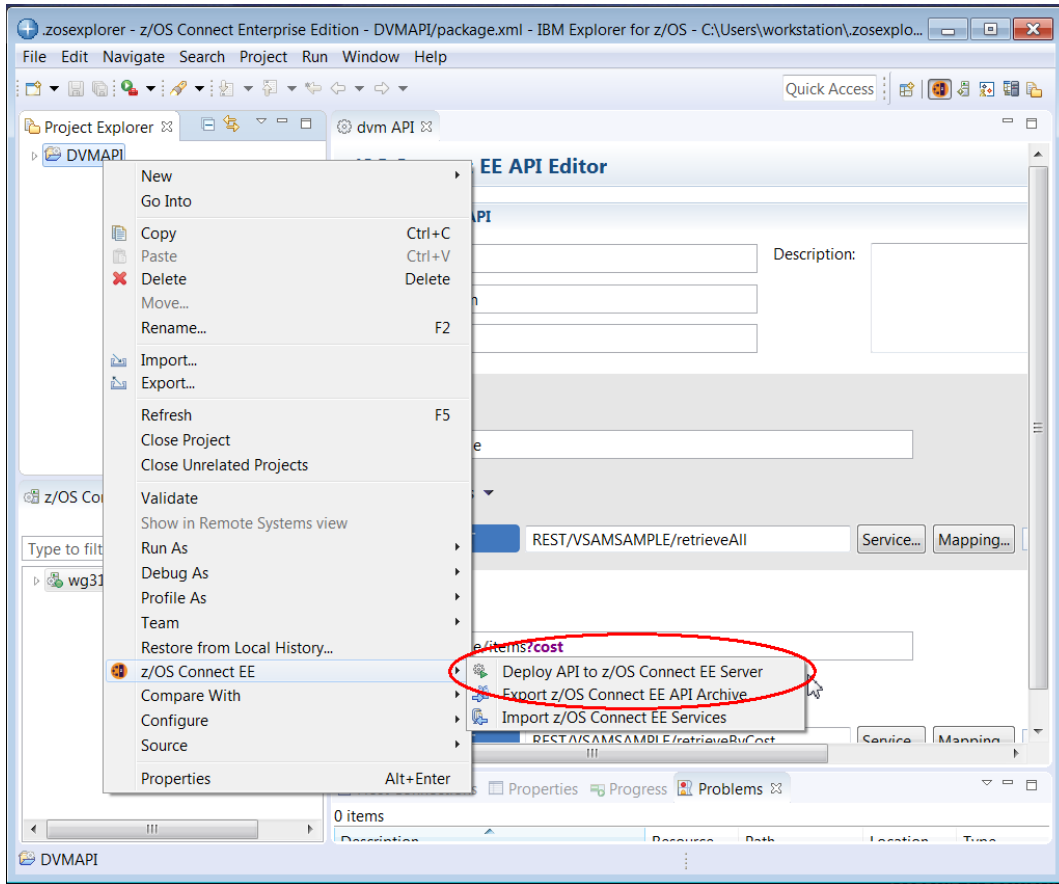
34. Close any open *request* or *response* mapping tabs.

## Summary

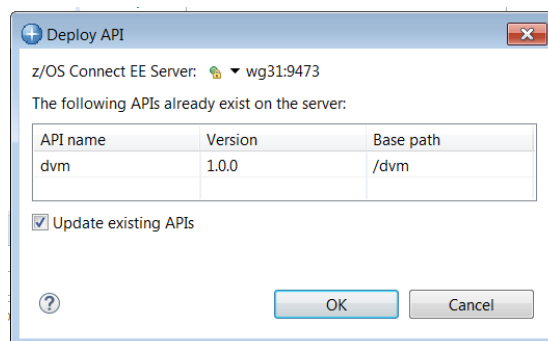
You created the API, which consists of two paths and the request and response mapping associated with each. That API will now be deployed into a z/OS Connect EE server.

## Deploy the API to a z/OS Connect EE Server

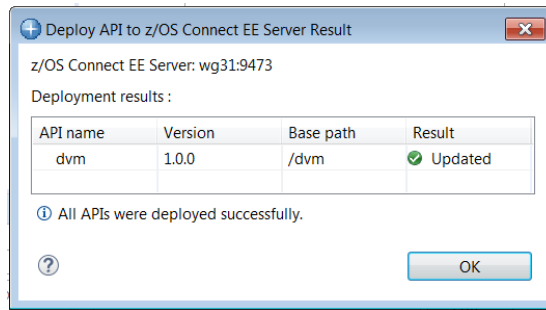
1. In the *Project Explorer* view (upper left), right-mouse click on the *DVMAPI* folder, then select *z/OS Connect EE* → *Deploy API to z/OS Connect EE Server*.



2. If the z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9473*). If z/OS Explorer had multiple connections to z/OS Connect servers then the pull-down arrow would allow a selection to which server to deploy, select *wg31:9453* from the list. Click **OK** on this window to continue.

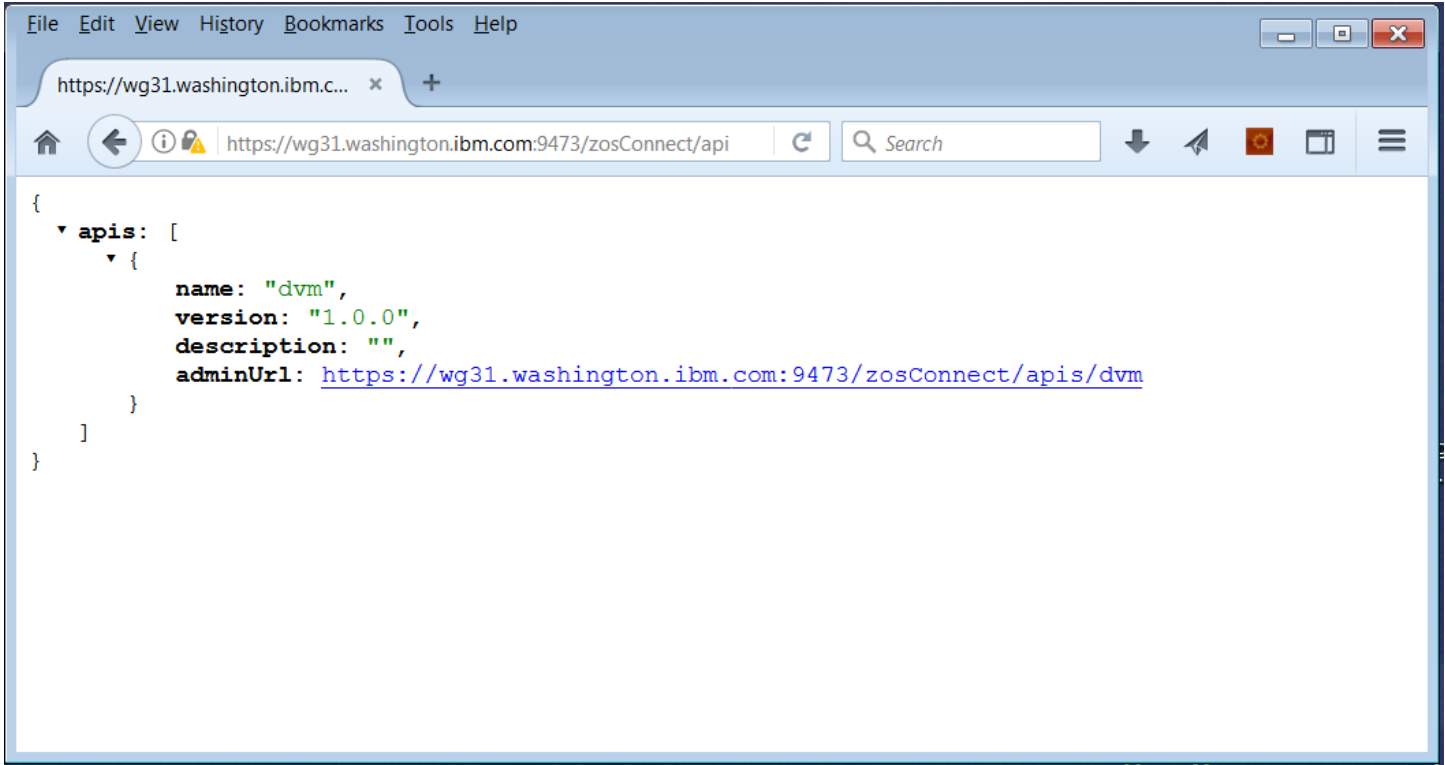


3. The API artifacts will be transferred to z/OS in an API archive (AAR) file and copied into the */var/ats/zosconnect/servers/zceedvm/resources/zosconnect/apis* directory.



## Test the DVM APIs

1. Next enter URL <https://wg31.washington.ibm.com:9473/zosConnect/apis> in the Firefox browser and you should see the window below.

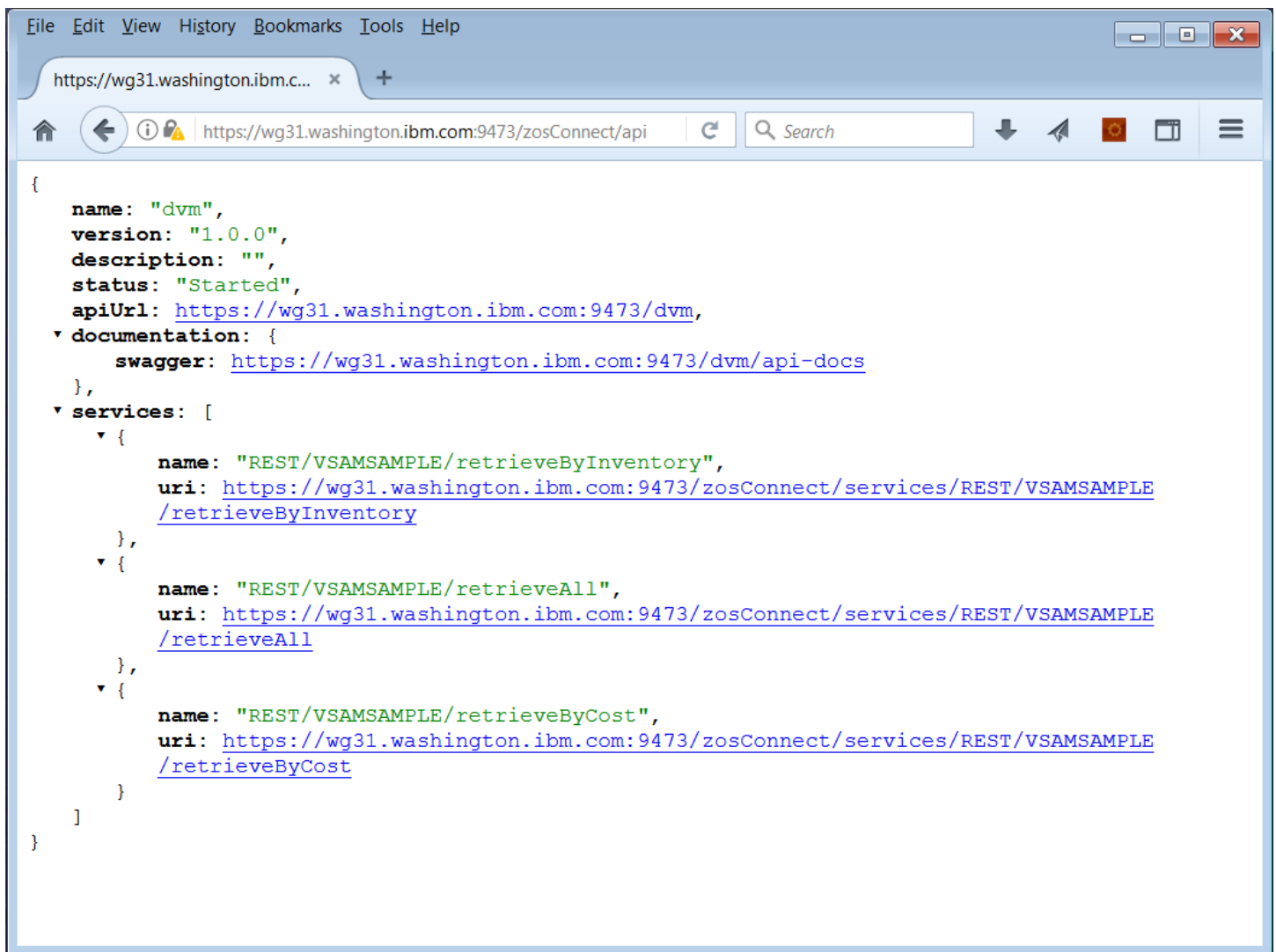


Note that other APIs may also be displayed.

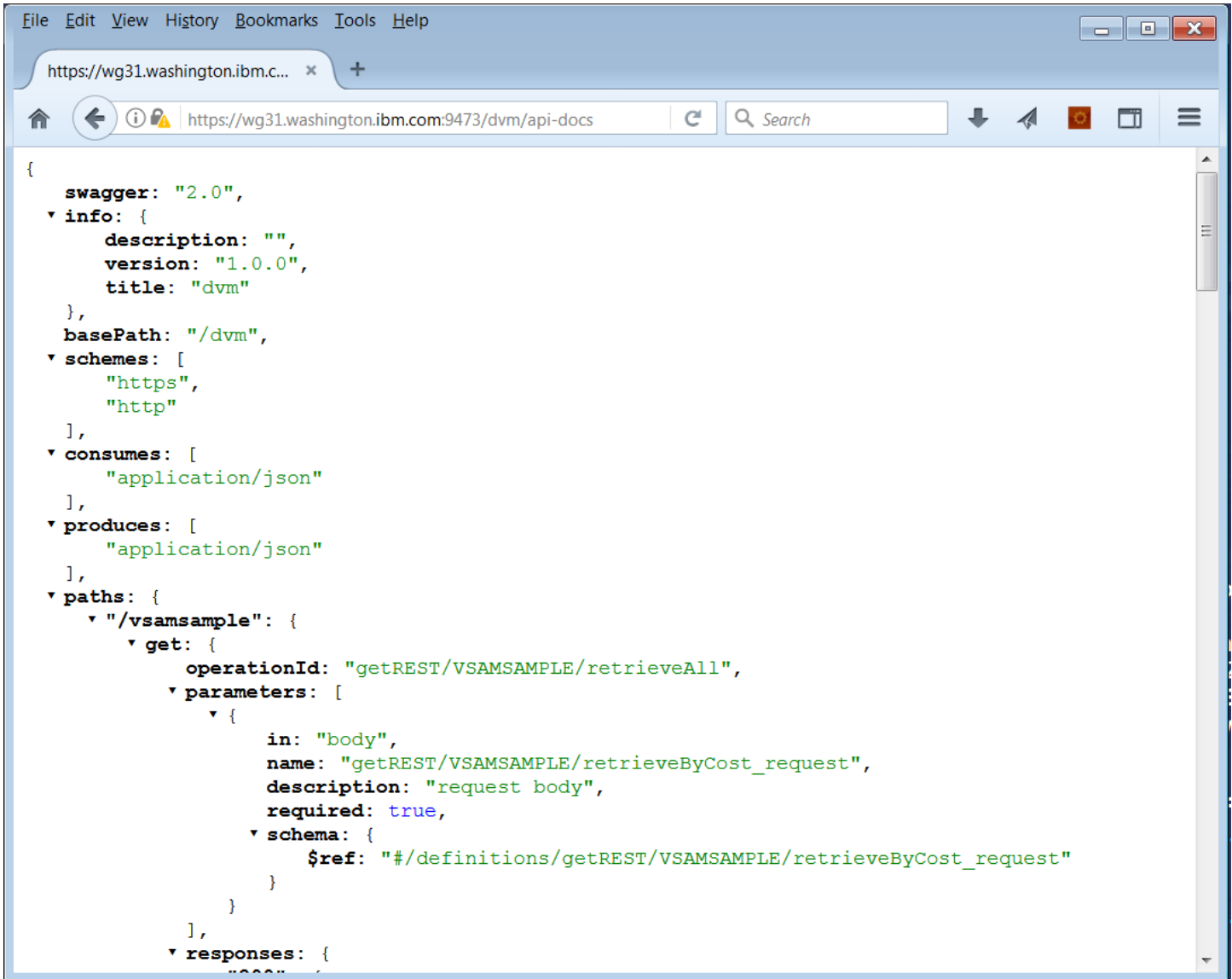
**Tech Tip:** It is very important to access the z/OS Connect server from a browser prior to any testing using the Swagger UI. Accessing a z/OS Connect URL from a browser starts an SSL handshake between the browser and the server. If this handshake has not performed prior to performing any test the test will fail with no message in the browser and no explanation. Ensuring this handshake has been performed is why you may be directed to access a z/OS Connect URL prior to using the Swagger UI.



3. If you click on *adminUrl* URL the window below should be displayed:

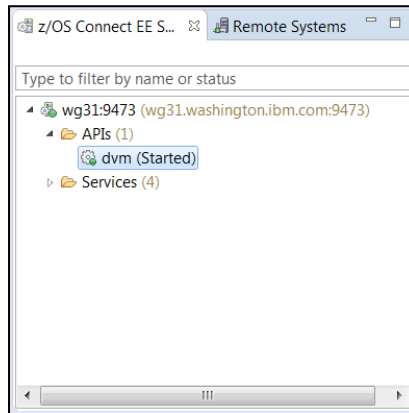


4. Finally click on the *swagger* URL for and you should see the Swagger document associated with this API.

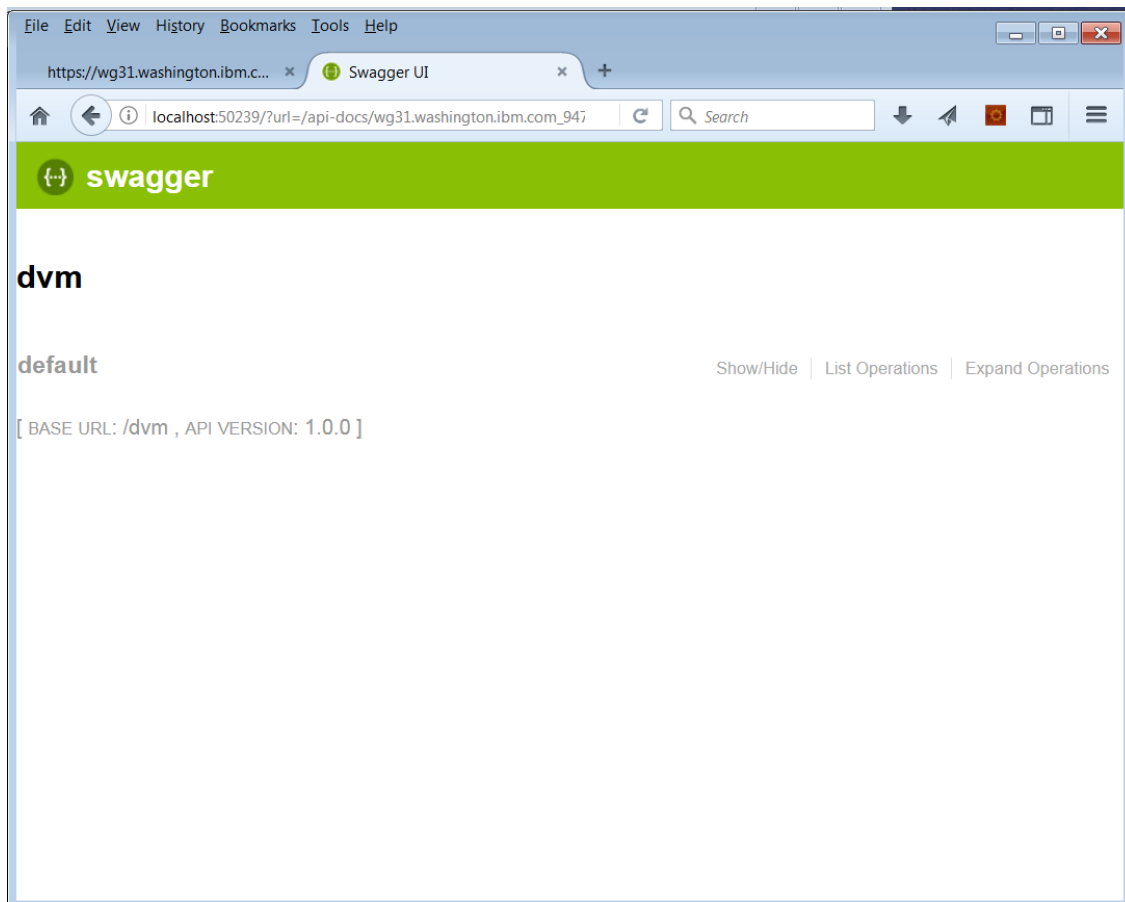


Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This Swagger document can be used by a developer or other tooling to develop REST clients for this specific API.

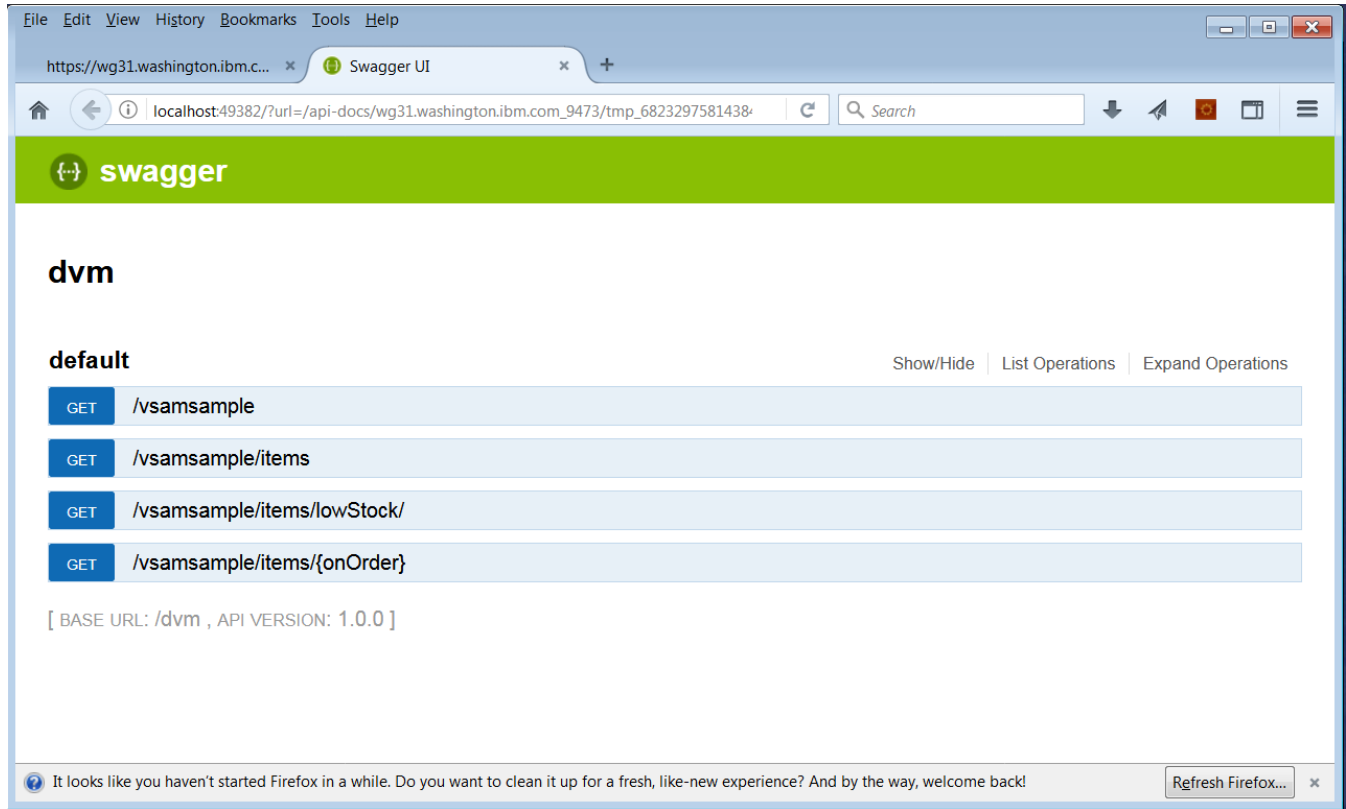
5. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect EE Servers*. Expand *wg31:9453* and then expand the *APIs* folder. You should see a list of the APIs installed in the server.



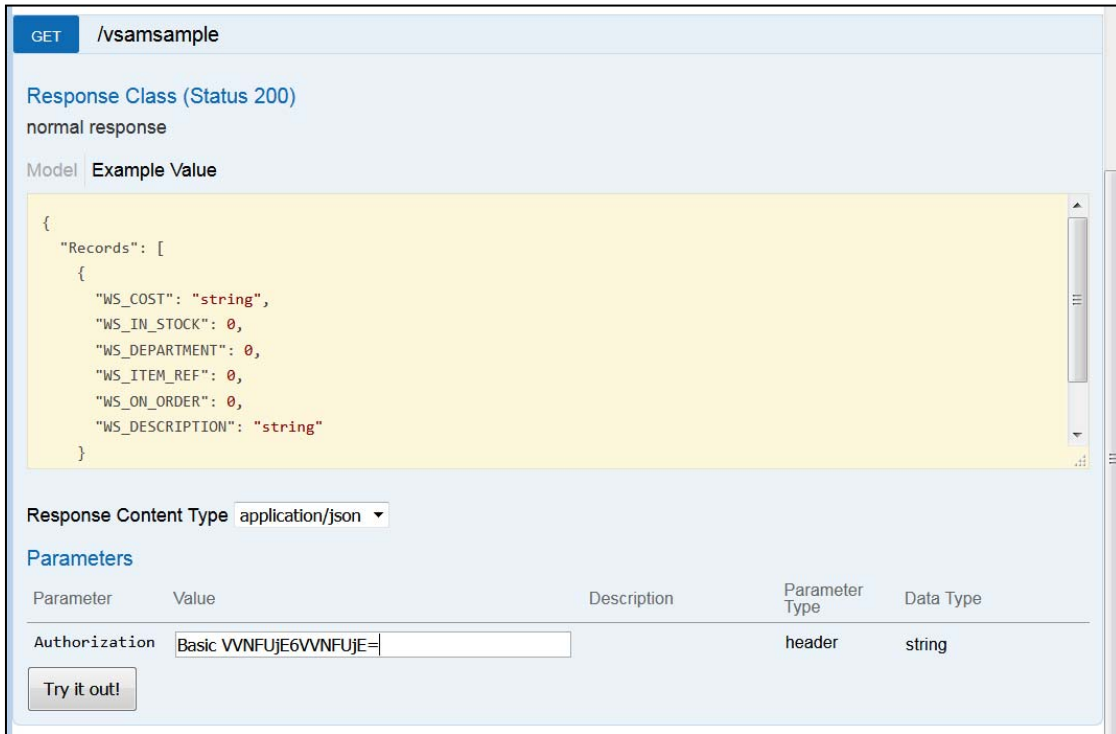
6. Right mouse button click on *dvm* and select *Open in Swagger UI*. Click OK if an informational prompt appears. This will open a new view showing a *Swagger* test client (see below).



7. Click on *List Operations* option in this view and this will display a list of available HTTP methods in this API.



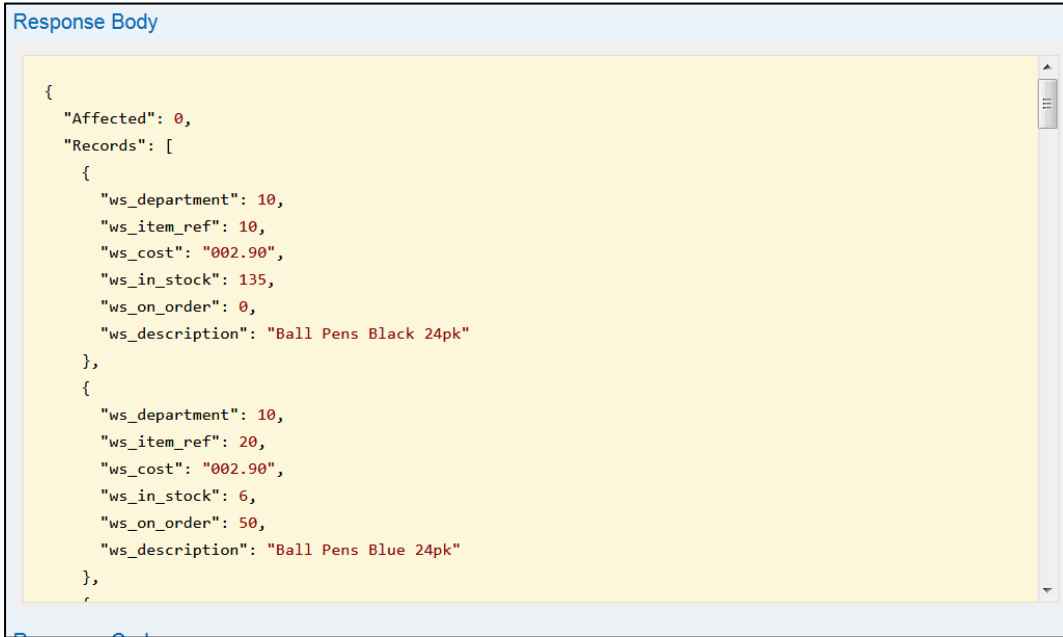
8. Select the *GET* method for selecting all records from the VSAM data set by clicking on the */vsamsample* URI string. Remember this was the *Path* specified for the *GET* method for the *retrieveAll* service when the API was defined. This action will expand this method in this view and provides a Swagger UI test client (you may have to use the slider bar and adjust the perspective to see the entire client).



9. Enter **Basic VVNFUjE6VVNFUjE=** in the box beside *Authorization* and press the **Try it out!** button. You may see a Security Alert pop-up warning about the self-signed certificate being used by the z/OS Connect EE server. Click **Yes** on this pop-up.

**Tech Tip:** The string *VVNFUjE6VVNFUjE=* is the string *USER1:USER1* encoded in base 64. See URL <https://www.base64encode.org/> for information on how this string was generated.

11. Scroll down the view and you should see the *Response Body* which contains the results of the GET method (see below). Note that the columns removed from the interface in an earlier step are not present.



```

{
  "Affected": 0,
  "Records": [
    {
      "ws_department": 10,
      "ws_item_ref": 10,
      "ws_cost": "002.90",
      "ws_in_stock": 135,
      "ws_on_order": 0,
      "ws_description": "Ball Pens Black 24pk"
    },
    {
      "ws_department": 10,
      "ws_item_ref": 20,
      "ws_cost": "002.90",
      "ws_in_stock": 6,
      "ws_on_order": 50,
      "ws_description": "Ball Pens Blue 24pk"
    }
  ]
}

```

Below are the contents of the VSAM data set.

Item#	Description	Dept	Cost	In Stock	On Order
0010	Ball Pens Black 24pk	010	002.90	0135	000
0020	Ball Pens Blue 24pk	010	002.90	0006	050
0030	Ball Pens Red 24pk	010	002.90	0106	000
0040	Ball Pens Green 24pk	010	002.90	0080	000
0050	Pencil with eraser 12pk	010	001.78	0083	000
0060	Highlighters Assorted 5pk	010	003.89	0013	040
0070	Laser Paper 28-lb 108 Bright 500/ream	010	007.44	0102	020
0080	Laser Paper 28-lb 108 Bright 2500/case	010	033.54	0025	000
0090	Blue Laser Paper 20lb 500/ream	010	005.35	0022	000
0100	Green Laser Paper 20lb 500/ream	010	005.35	0003	020
0110	IBM Network Printer 24 - Toner cart	010	169.56	0012	000
0120	Standard Diary: Week to view 8 1/4x5 3/4	010	025.99	0007	000
0130	Wall Planner: Erasable 36x24	010	018.85	0003	000
0140	70 Sheet Hard Back wire bound notepad	010	005.89	0084	000
0150	Sticky Notes 3x3 Assorted Colors 5pk	010	005.35	0036	045
0160	Sticky Notes 3x3 Assorted Colors 10pk	010	009.75	0067	030
0170	Sticky Notes 3x6 Assorted Colors 5pk	010	007.55	0064	030
0180	Highlighters Yellow 5pk	010	003.49	0088	010
0190	Highlighters Blue 5pk	010	003.49	0076	020
0200	12 inch clear rule 5pk	010	002.12	0014	010
0210	Clear sticky tape 5pk	010	004.27	0073	000

12. Click on the URI for the **GET** method for `/vsamsample/items` to open its *Swagger Test* user interface and scroll down to the *Response Content Type* area.

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
<b>cost</b>	<input type="text" value="(required)"/>		query	string
Authorization	<input type="text"/>		header	string

Note that this API requires one a query parameter *cost*. This parameter is present because the path `/vsamsample/items?cost` was specified when the API was developed. Enter **002.90** for the *cost* and **Basic VVNFUjE6VVNFUjE=** for the *Authorization* and press the **Try it Out!** button. Scroll down and you should see the following information in the *Response Body*.

Response Body

```
{
  "Affected": 0,
  "TotRecs": 4,
  "Skipped": 0,
  "NumRecs": 4,
  "NumFields": 6,
  "Records": [
    {
      "ws_department": 10,
      "ws_item_ref": 10,
      "ws_cost": "002.90",
      "ws_in_stock": 135,
      "ws_on_order": 0,
      "ws_description": "Ball Pens Black 24pk"
    },
    {
      "ws_department": 10,
      "ws_item_ref": 20,
      "ws_cost": "002.90",
      "ws_in_stock": 6
    }
  ]
}
```

Note that the contents of columns *TotRecs*, *Skipped*, *NumRecs* and *NumFields* are displayed. This were not removed as they were in the **GET** method for the *retrieveByCost* service.

Try a few other combinations for *cost* and compare the results with the above table.

13. Click on the URI for the **GET** method for `/vsamsample/items/{onOrder}` to open its *Swagger Test* user interface and scroll down to the *Response Content Type* area.

Response Content Type application/json ▼

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
<b>onOrder</b>	<input type="text" value="(required)"/>		path	string
<b>inStock</b>	<input type="text" value="(required)"/>		query	string
Authorization	<input type="text"/>		header	string

Note that this API requires two parameters, *onOrder* and *inStock*. These are present because the path `/vsamsample/items/{onOrder}?inStock` was specified when the API was developed. Enter **10** for *onOrder* and **10** for *inStock* and **Basic VVNFUjE6VVNFUjE=** for *Authorization* and press the **Try it Out!** button. Scroll down and you should see the following information in the *Response Body*.

**Response Body**

```

{
  "Affected": 0,
  "Records": [
    {
      "ws_department": 10,
      "ws_item_ref": 120,
      "ws_cost": "025.99",
      "ws_in_stock": 7,
      "ws_on_order": 0,
      "ws_description": "Standard Diary: Week to view 8 1/4x5 3/4"
    },
    {
      "ws_department": 10,
      "ws_item_ref": 130,
      "ws_cost": "018.85",
      "ws_in_stock": 3,
      "ws_on_order": 0,
      "ws_description": "Wall Planner: Eraseable 36x24"
    }
  ]
}

```



14. Click on the URI for the **GET** method for `/vsamsample/items/lowStock` to open its *Swagger Test* user interface and scroll down to the *Response Content Type* area.

Response Content Type: application/json

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
Authorization	<input type="text"/>		header	string
inStock	(required)		query	string

Note that this API requires only one parameter, *inStock*. The *onOrder* parameter has been set to 999 by the API developer and omitted from the interface. Enter **10** for *inStock* and **Basic VVNFUjE6VVNFUjE=** for *Authorization* and press the **Try it Out!** button. Scroll down and you should see the following information in the *Response Body* showing the items with less than 10 items in stock.

**Response Body**

```
{
  "Affected": 0,
  "Records": [
    {
      "ws_department": 10,
      "ws_item_ref": 20,
      "ws_cost": "002.90",
      "ws_in_stock": 6,
      "ws_on_order": 50,
      "ws_description": "Ball Pens Blue 24pk"
    },
    {
      "ws_department": 10,
      "ws_item_ref": 100,
      "ws_cost": "005.35",
      "ws_in_stock": 3,
      "ws_on_order": 20,
      "ws_description": "Green Laser Paper 20lb 500/ream"
    }
  ]
}
```

## Summary

You use DVM to develop 3 services. The SAR files for the 3 services were imported in the API Editor of z/OS Connect EE. The API Editor was used to develop 4 RESTful APIs. You have verified the API. The API layer provided a further level of abstraction and allows a more flexible use of HTTP verbs, and better mapping of data via the API editor function.