

IBM z/OS Connect (OpenAPI 2.0)

Developing RESTful APIs for Db2 DVM Services



**IBM Z
Wildfire Team –
Washington System Center**

Lab Version Date: April 23, 2022

Table of Contents

Overview	3
Use the Data Virtualization Manager Studio to create a virtual table.....	4
Use the Data Virtualization Manager Studio to test SQL commands.....	12
Use the Data Virtualization Manager Studio to create Web Services.....	15
Use the Data Virtualization Manager Studio to deploy the DVM services	29
Use the Data Virtualization Manager Studio to export service archive files.....	32
Using z/OS Connect APIs to access DVM services	33
<i>Connect to a z/OS Connect Server.....</i>	<i>33</i>
<i>Create the Db2 DVM API Project</i>	<i>36</i>
<i>Import the SAR files generated by the DVM Studio</i>	<i>38</i>
<i>Compose an API for the IMS DVM Rest Services.....</i>	<i>41</i>
<i>Deploy the API to a z/OS Connect Server.....</i>	<i>50</i>
<i>Test the Db2 APIs using Swagger UI</i>	<i>52</i>
<i>Test the DB2 APIs using Postman.....</i>	<i>66</i>

Overview

The objective of these exercise is to gain experience using Data Virtualization Manager (DVM) Studio and the z/OS Connect API Toolkit to create RESTful API to IMS data bases. For information about scheduling this workshop in your area contact your IBM representative.

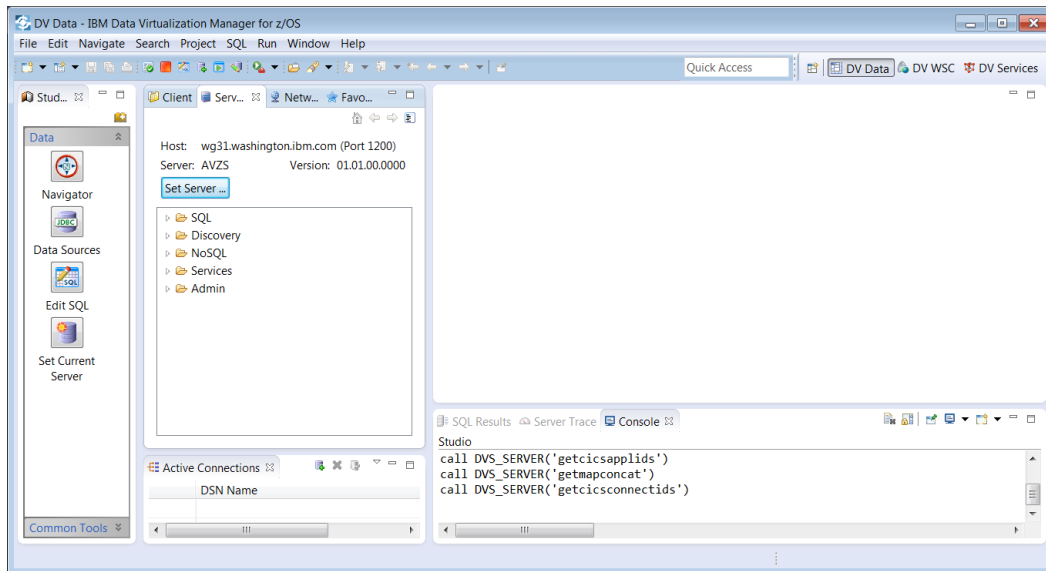
General Exercise Information and Guidelines

- ✓ This exercise requires using z/OS user identity *USER1*. The RACFpassword for this user is USER1.
- ✓ Do not hesitate to request assistance anytime you have any questions about the use of the Data Virtualization Manager Studio, IBM z/OS Explorer, z/OS Connect Toolkit features or other tools
- ✓ The Db2 table used for this exercise is the sample table provided by the Db2 installation verification program (*DSN8120.EMP*). For details of this table, see URL <https://www.ibm.com/docs/en/db2-for-zos/12?topic=tables-employee-table-dsn8c10emp>
- ✓ Please note that there may be minor differences between the screen shots in this exercise versus what you see on your desktop. These differences should not impact the completion of this exercise.
- ✓ Text in **bold** and highlighted in **yellow** in this document should be available for copying and pasting in a file named *Development APIs for DVM CopyPaste* file on the desktop.

Create a DVM virtual table

Access to a Db2 table using DVM SQL commands requires the creation of a DVM virtual table. The virtual table represents the columns in the table. In this section a virtual table for Db2 sample EMP table will be created.

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.



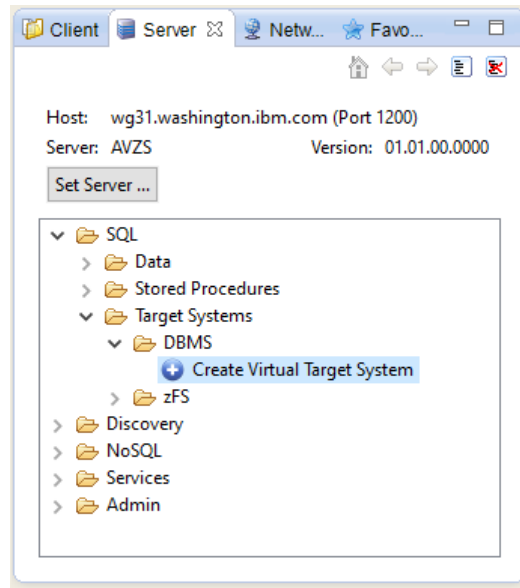
Tech-Tip: Eclipse based development tools like DVM Studio; provide a graphical interface consisting of multiple views within a single window.

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).

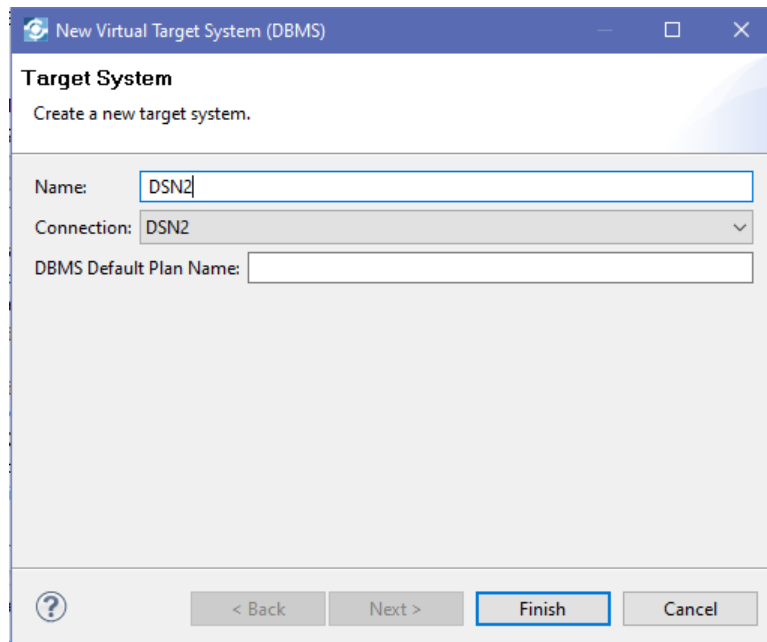
At any time, a specific view can be enlarged to fill the entire window by double clicking in the view's title bar. Double clicking in the view's title bar will be restored the original arrangement. If a DVM Studio view is closed or otherwise disappears, the original arrangement can be restored by selecting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

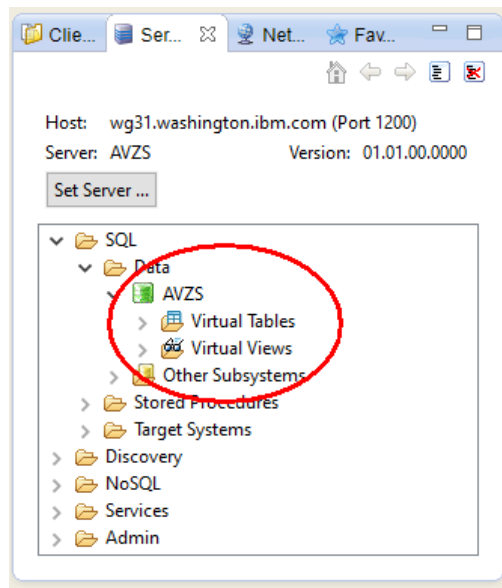
___2. First expand the *SQL* folder then the *Target Systems* folder and then double click *Create Virtual Target System*.



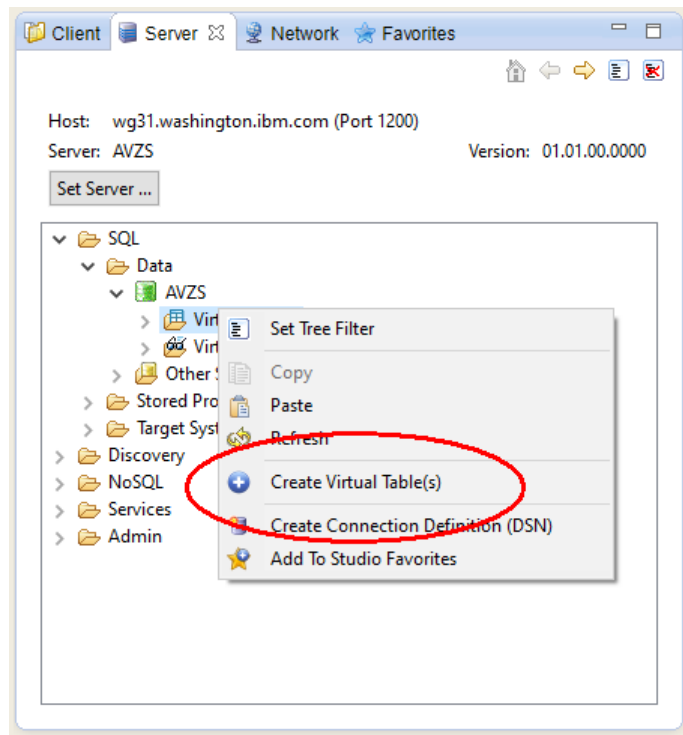
___3. This opens the *New Virtual Target System (DBMS) – Target System* window, click Finish to continue.



4. Next expand the *SQL* folder then the *Data* folder and then the *AVZS* folder to display the *Virtual Tables* and *Virtual Views*, see below.

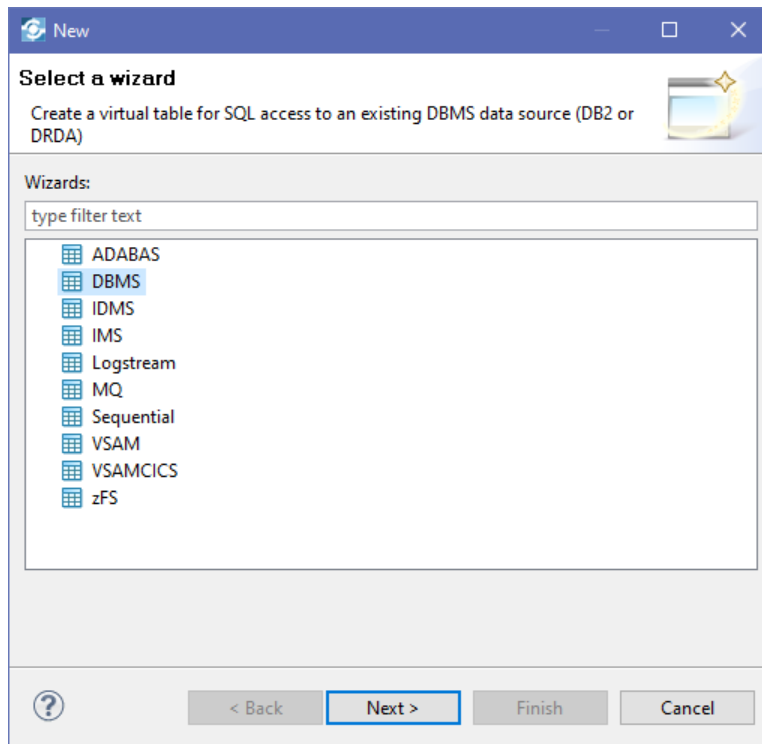


5. Select the *Virtual Tables* folder and right mouse button click and then select *Create Virtual Table(s)* option, see below.

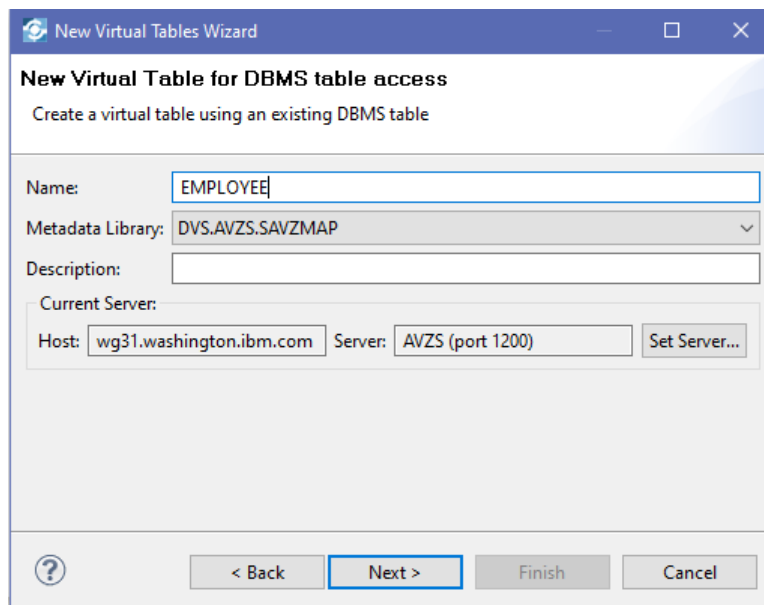


Tech-Tip: You may be presented with a *New Connection Definition (DSN)* pop-up. Just click the **OK** button to proceed.

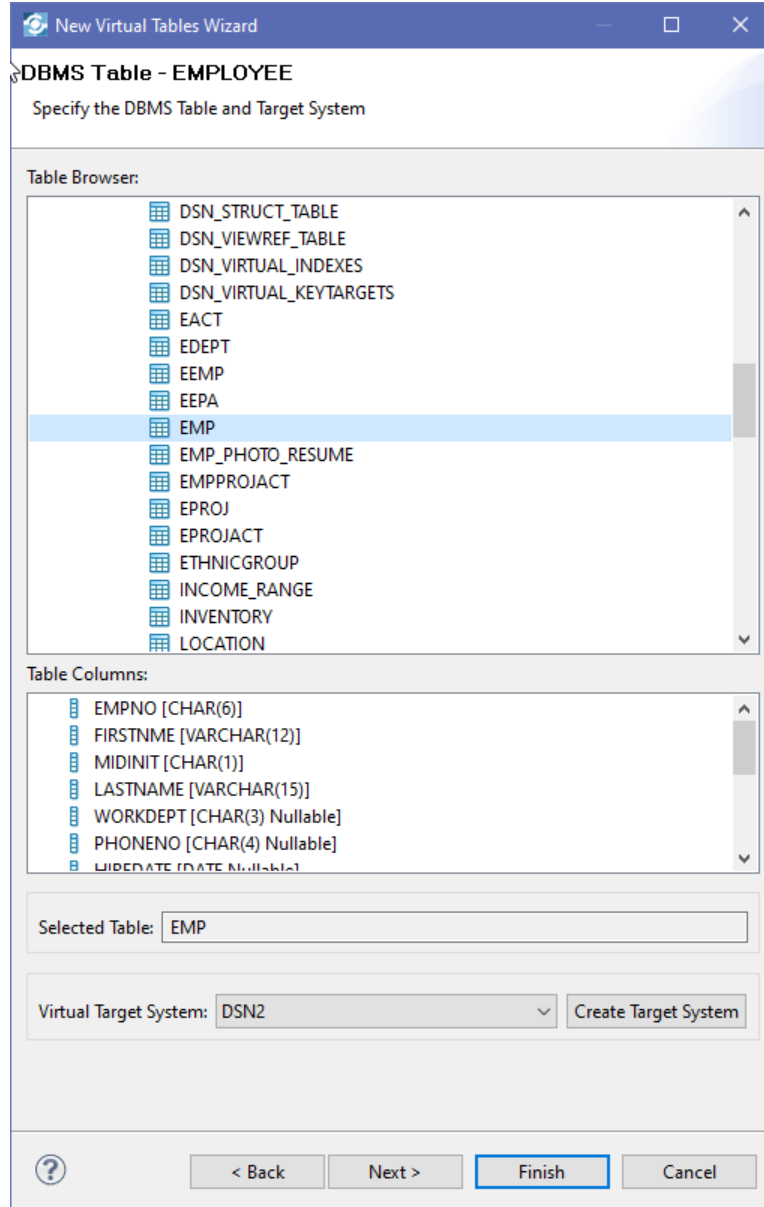
7. On the *Select a wizard* window select, *DBMS* and press **Next** to continue.



8. Next create a virtual table for the Db2 employee sample table.. On the *New Virtual Table Wizard – New Virtual Table for DBMS table access* window enter **EMPLOYEE** as the name of the table. Click **Next** to continue.

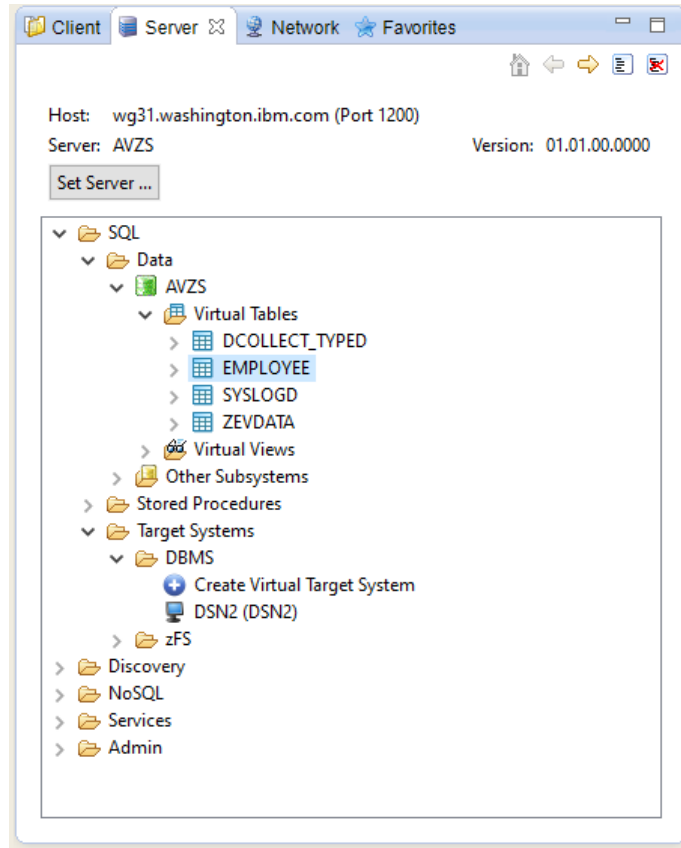


9. On the *DBMS Table - Employee* window, expand *DSN2 (DB2Mbr)*. Then expand schema *DSN81210*, then *Tables* and then scroll down and select table *EMP*, e.g., Db2 sample table *DSN81210.EMP*.

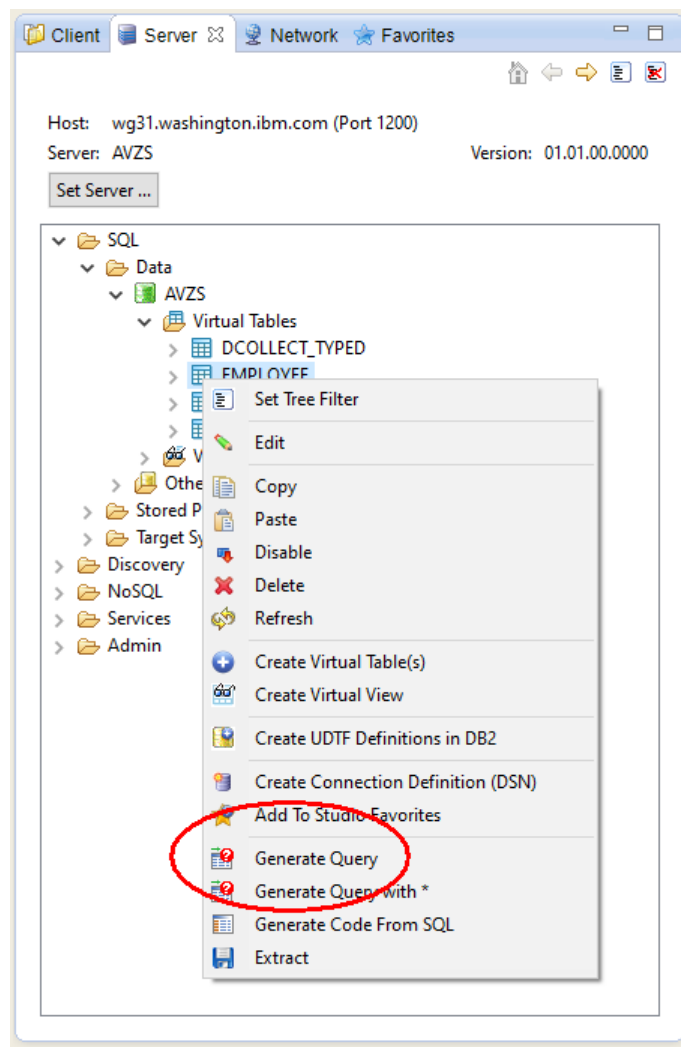


10. Click **Finish** to continue.

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



12. Select option *Generate Query* and click **Yes** on the *Execute Query?* pop up window. If a *New Connection Definition(DSN)* pop up window appears, click **OK** to continue.



13. This will access the Db2 table and display the rows of the table in the view on the lower right-hand side

	EMPNO	FIRSTNAME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM	
0	000010	CHRISTINE	I	HAAS	A00	3978	1965-0...	PRES	18	F	1933-0...	52750.00	1000.00	4220.00	
1	000020	MICHAEL	L	THOMPSON	B01	3476	1973-1...	MANAGER	18	M	1948-0...	41250.00	800.00	3300.00	
2	000030	SALLY	A	KWAN	C01	4738	1975-0...	MANAGER	20	F	1941-0...	38250.00	800.00	3060.00	
3	000050	JOHN	B	GEYER	E01	6789	1949-0...	MANAGER	16	M	1925-0...	40175.00	800.00	3214.00	
4	000060	IRVING	F	STERN	D11	6423	1973-0...	MANAGER	16	M	1945-0...	32250.00	600.00	2580.00	
5	000070	EVA	D	PULASKI	D21	7831	1980-0...	MANAGER	16	F	1953-0...	36170.00	700.00	2893.00	
6	000090	EILEEN	W	HENDERSON	E11	5498	1970-0...	MANAGER	16	F	1941-0...	29750.00	600.00	2380.00	
7	000100	THEODORE	Q	SPENSER	E21	0972	1980-0...	MANAGER	14	M	1956-1...	26150.00	500.00	2092.00	
8	000110	VINCENZO	G	LUCCHESI	A00	3490	1958-0...	SALESREP	19	M	1929-1...	46500.00	900.00	3720.00	
9	000120	SEAN		O'CONNELL	A00	2167	1963-1...	CLERK	14	M	1942-1...	29250.00	600.00	2340.00	
10	000130	DOLORES	M	QUINTANA	C01	4578	1971-0...	ANALYST	16	F	1925-0...	23800.00	500.00	1904.00	
11	000140	HEATHER	A	NICHOLLS	C01	1793	1976-1...	ANALYST	18	F	1946-0...	28420.00	600.00	2274.00	
12	000150	BRUCE		ADAMSON	D11	4510	1972-0...	DESIGNER	16	M	1947-0...	25280.00	500.00	2022.00	
13	000160	ELIZABETH	R	PIANKA	D11	3782	1977-1...	DESIGNER	17	F	1955-0...	22250.00	400.00	1780.00	
14	000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-0...	DESIGNER	16	M	1951-0...	24680.00	500.00	1974.00	
15	000180	MARILYN	S	SCOUTTEN	D11	1682	1973-0...	DESIGNER	17	F	1949-0...	21340.00	500.00	1707.00	
16	000190	JAMES	H	WALKER	D11	2986	1974-0...	DESIGNER	16	M	1952-0...	20450.00	400.00	1636.00	
17	000200	DAVID		BROWN	D11	4501	1966-0...	DESIGNER	16	M	1941-0...	27740.00	600.00	2217.00	
18	000210	WILLIAM	T	JONES	D11	0942	1979-0...	DESIGNER	17	M	1953-0...	18270.00	400.00	1462.00	
19	000220	JENNIFER	K	LUTZ	D11	0672	1968-0...	DESIGNER	18	F	1948-0...	29840.00	600.00	2387.00	
20	000230	JAMES	J	JEFFERSON	D21	4265	1966-1...	CLERK	14	M	1935-0...	22180.00	400.00	1774.00	
21	000240	SALVATORE	M	MARINO	D21	3780	1979-1...	CLERK	17	M	1954-0...	28760.00	600.00	2301.00	
22	000250	DANIEL	S	SMITH	D21	0961	1969-1...	CLERK	15	M	1939-1...	19180.00	400.00	1534.00	
23	000260	SYBIL	V	JOHNSON	D21	8953	1975-0...	CLERK	16	F	1936-1...	17250.00	300.00	1380.00	
24	000270	MARIA	L	PEREZ	D21	9001	1980-0...	CLERK	15	F	1953-0...	27380.00	500.00	2190.00	
25	000280	ETHEL	R	SCHNEIDER	E11	8997	1967-0...	OPERATOR	17	F	1936-0...	26250.00	500.00	2100.00	
26	000290	JOHN	R	PARKER	E11	4502	1980-0...	OPERATOR	12	M	1946-0...	15340.00	300.00	1227.00	
27	000300	PHILIP	X	SMITH	E11	2095	1972-0...	OPERATOR	14	M	1936-1...	17750.00	400.00	1420.00	
28	000310	MAUDE	F	SETRIGHT	E11	3332	1964-0...	OPERATOR	12	F	1931-0...	15900.00	300.00	1272.00	
29	000320	RAMLAL	V	MEHTA	E21	9990	1965-0...	FIELDREP	16	M	1932-0...	19950.00	400.00	1596.00	
30	000330	WING		LEE	E21	2103	1976-0...	FIELDREP	14	M	1941-0...	25370.00	500.00	2030.00	
31	000340	JASON	R	GOUNOT	E21	5698	1947-0...	FIELDREP	16	M	1926-0...	23840.00	500.00	1907.00	
32	200010	DIAN	J	HEMMINGER	A00	3978	1965-0...	SALESREP	18	F	1933-0...	46500.00	1000.00	4220.00	
33	200120	GREG		ORLANDO	A00	2167	1972-0...	CLERK	14	M	1942-1...	29250.00	600.00	2340.00	
34	200140	KIM	N	NATZ	C01	1793	1976-1...	ANALYST	18	F	1946-0...	28420.00	600.00	2274.00	
35	200170	KIYOSHI		YAMAMOTO	D11	2890	1978-0...	DESIGNER	16	M	1951-0...	24680.00	500.00	1974.00	
36	200220	REBA	K	JOHN	D11	0672	1968-0...	DESIGNER	18	F	1948-0...	29840.00	600.00	2387.00	

Columns Group: 1 of 1 Columns per group: 25

42 rows SQL Messages

Verification of the virtual table completes the creation of the virtual table for the sample EMP table.

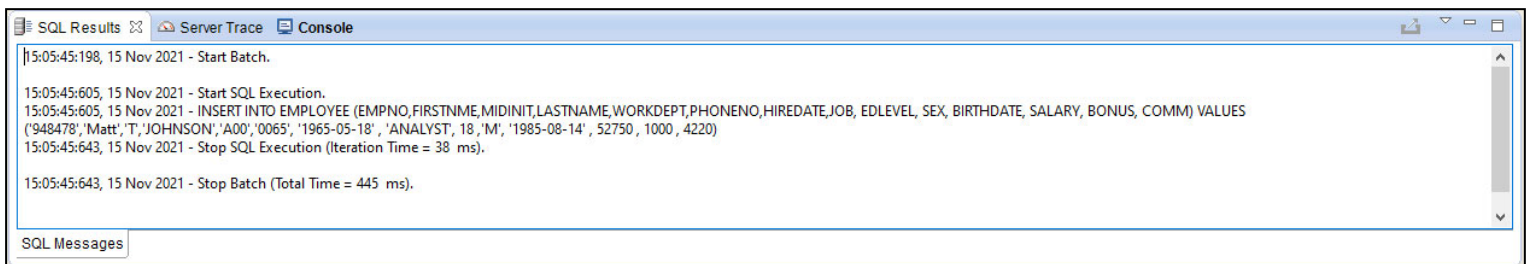
Explore accessing a virtual table using SQL commands

Now that the virtual tables have been created, they can be used to explore and test SQL commands in the *Generated.sql* pane of the DVM studio.

1. In the DVM studio, focus on the *Generated.sql* pane, see below. Enter a SQL INSERT command as shown below:

```
INSERT INTO EMPLOYEE  
(EMPNO,FIRSTNME,MIDINIT,LASTNAME,WORKDEPT,PHONENO,HIREDATE,JOB, EDLEVEL,  
SEX, BIRTHDATE, SALARY, BONUS, COMM)  
VALUES ('948478','Matt','T','JOHNSON','A00','0065', '1965-05-18', 'ANALYST', 18,'M', '1985-08-14',  
52750 , 1000 , 4220)
```

2. Select the entire *INSERT* command and right mouse button click and select the *Execute SQL* option. You should see results like the one below in the *SQL Results* pane.



3. Use a SQL SELECT command to display the inserted row.

SELECT * FROM EMPLOYEE WHERE EMPNO='948478'

	EMPNO	FIRSTNAME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
0	948478	Matt	T	JOHNSON	A00	0065	1965-0...	ANALYST	18	M	1985-0...	52750.00	1000.00	4220.00

Columns Group: 1 of 1 Columns per group: 25

1 rows SQL Messages

4. Use a SQL UPDATE command to update the new row's bonus column.

UPDATE EMPLOYEE SET BONUS=2000 WHERE EMPNO='948478'

15:13:13:893, 15 Nov 2021 - Start Batch.

15:13:14:300, 15 Nov 2021 - Start SQL Execution.

15:13:14:300, 15 Nov 2021 - UPDATE EMPLOYEE SET BONUS=2000 WHERE EMPNO='948478'

15:13:14:342, 15 Nov 2021 - Stop SQL Execution (Iteration Time = 42 ms).

15:13:14:342, 15 Nov 2021 - Stop Batch (Total Time = 449 ms).

SQL Messages

5. Use a SQL SELECT command to display the row's updated information.

SELECT * FROM EMPLOYEE WHERE EMPNO='948478'

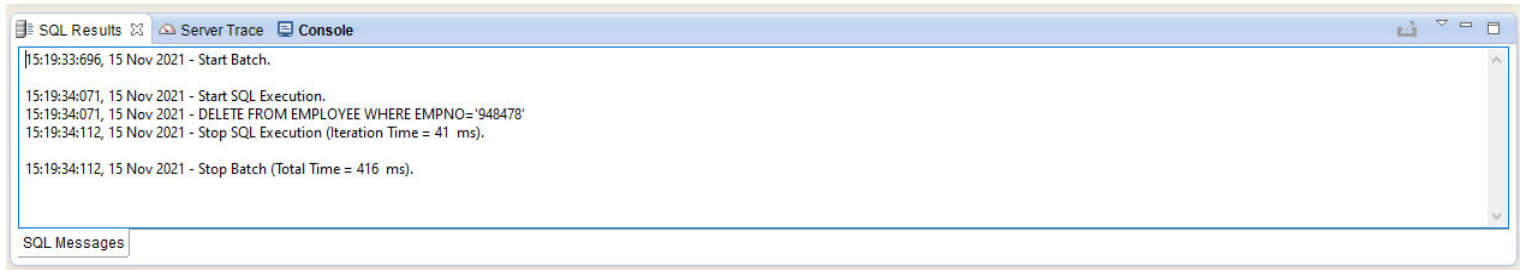
	EMPNO	FIRSTNAME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
0	948478	Matt	T	JOHNSON	A00	0065	1965-0...	ANALYST	18	M	1985-0...	52750.00	2000.00	4220.00

Columns Group: 1 of 1 Columns per group: 25

1 rows SQL Messages

6. Use a SQL DELETE command to delete row.

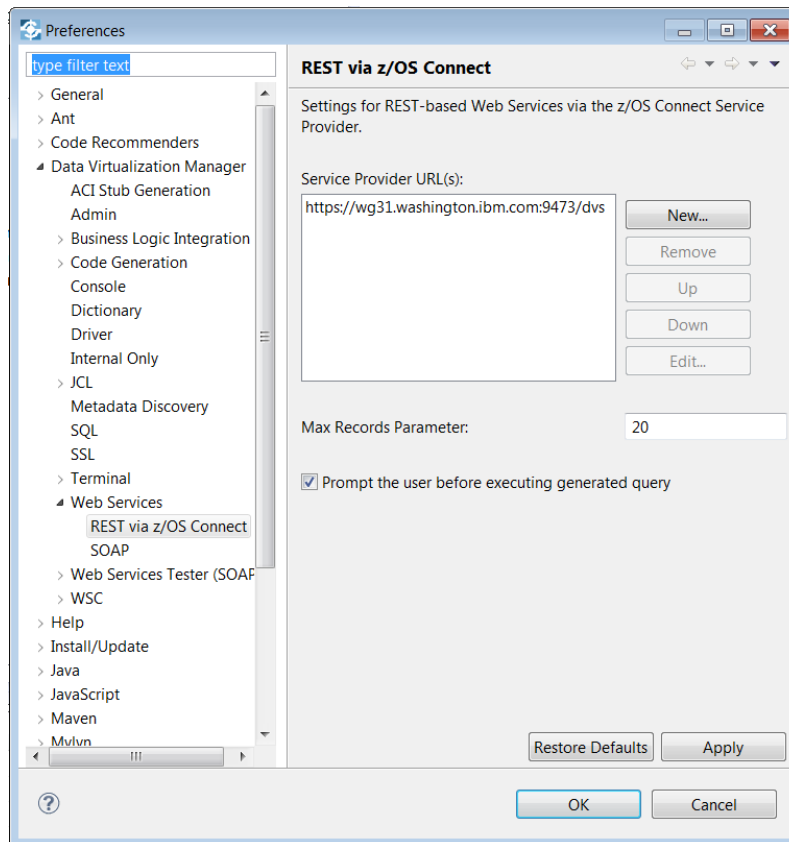
DELETE FROM EMPNO WHERE EMPNO='948478'



The purpose of exploring these SQL command using the studio was to demonstrate how the virtual tables can be accessed to insert, update, retrieve and delete segments in a hierarchical data base. What we learned regarding these SQL commands can be now be used to create DVM services.

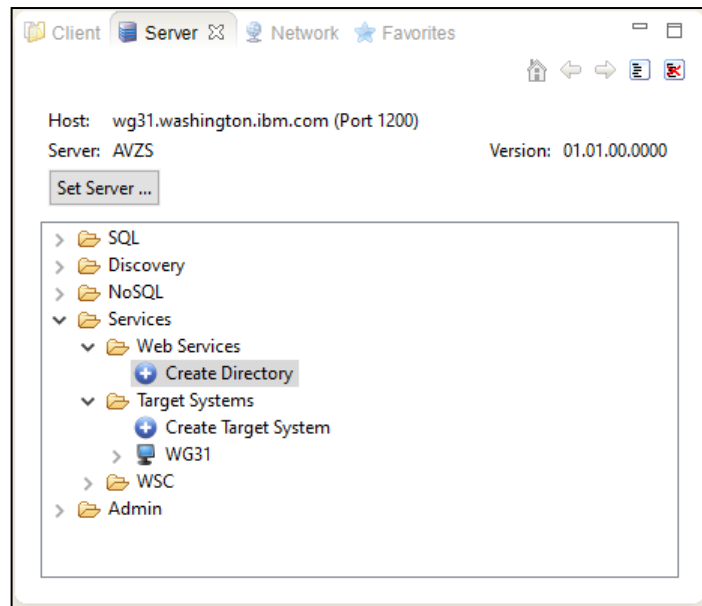
Create a Web Services

1. Confirm that the DVM studio is properly configured to communicate with the z/OS Connect server with 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:

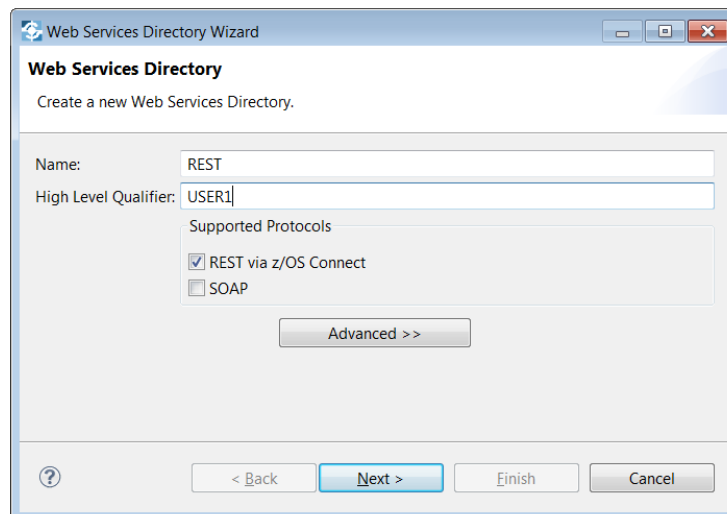


2. Ensure the *Service Provider URL(s)* is set to **<https://wg31.washington.ibm.com:9473/dvs>**. If not, select the provider and use the **Edit** button to set it correctly or if a *Service Provide URL* is not present, add one.

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

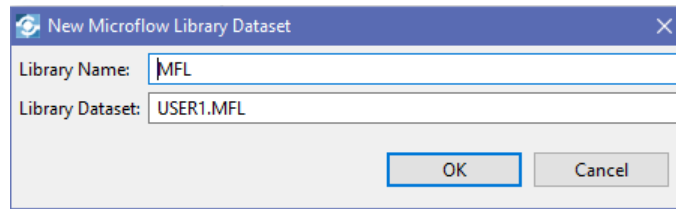


4. If the */REST/* Web Services directory does not exist, double click on *Create Directory* to open the *Web Services Directory Wizard*. Enter **REST** and **USER1** as shown below. Click **Next** to continue. Otherwise continue with Step 7.



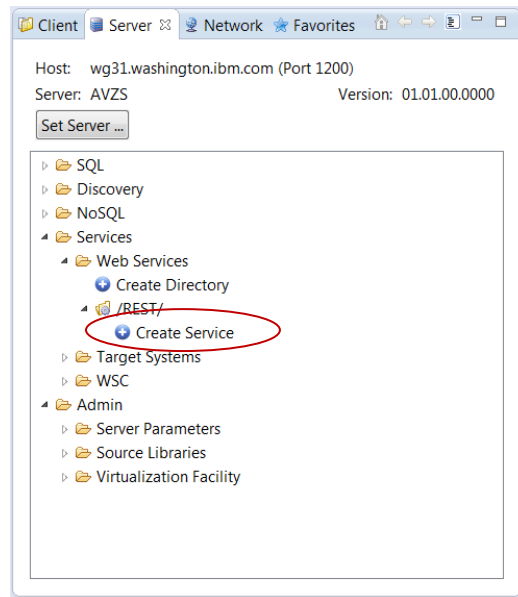
Tech-Tip: The *High-Level Qualifier* will be used to create a micro flow partitioned data set with a data set name of USER1.MFL.

- ___ 5. On the *Web Service Directory Wizard – Microflow Library* window, if there is a current Microflow liberty (e.g., USER1.MFL), select it. Otherwise click the **Create New Microflow Library** button and accept the defaults on the *New Microflow Library Dataset* pop-up window. Click **OK** to continue.



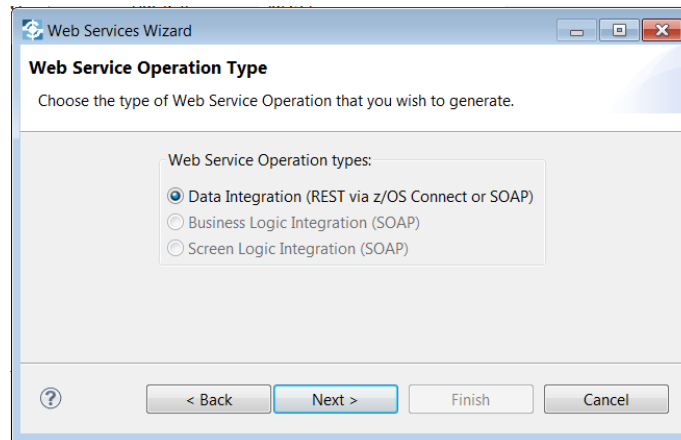
- ___ 6. On the *Microflow Library* window select *MFL* under *Current Microflow Libraries* and click **Finish** to continue.

- ___ 7. Expand */REST/* and use the *Create Service* wizard to create a new web service.

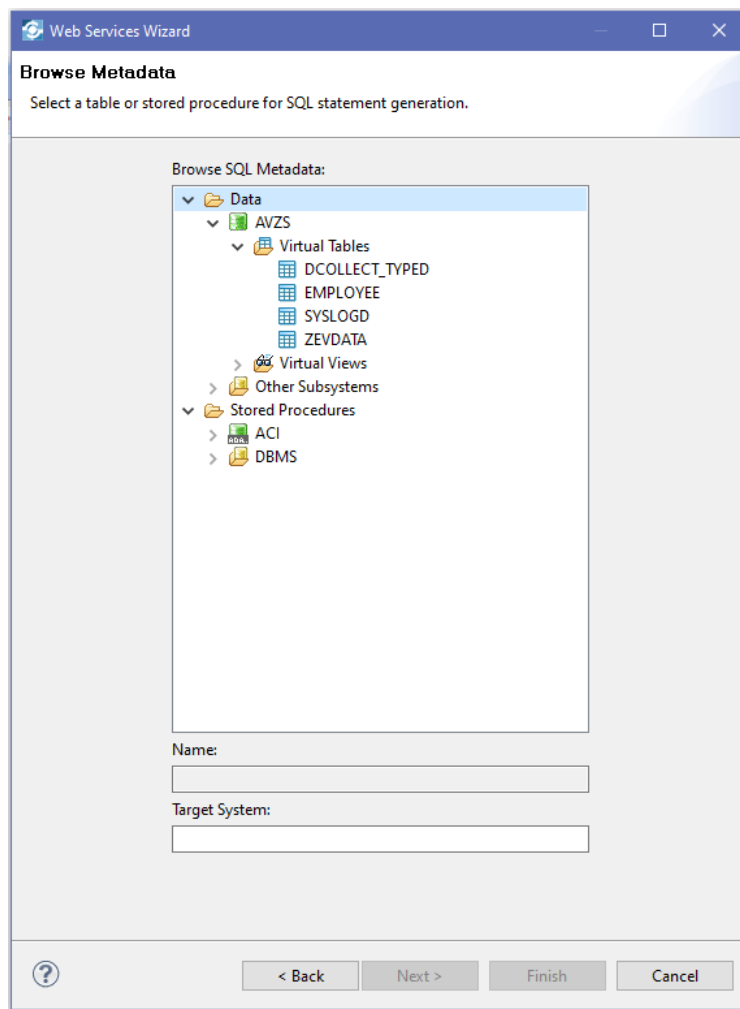


- ___ 8. On the *Web Service – Create a new Web Service* window enter **EMPLOYEE** as the *Name* and press **Next** to continue.

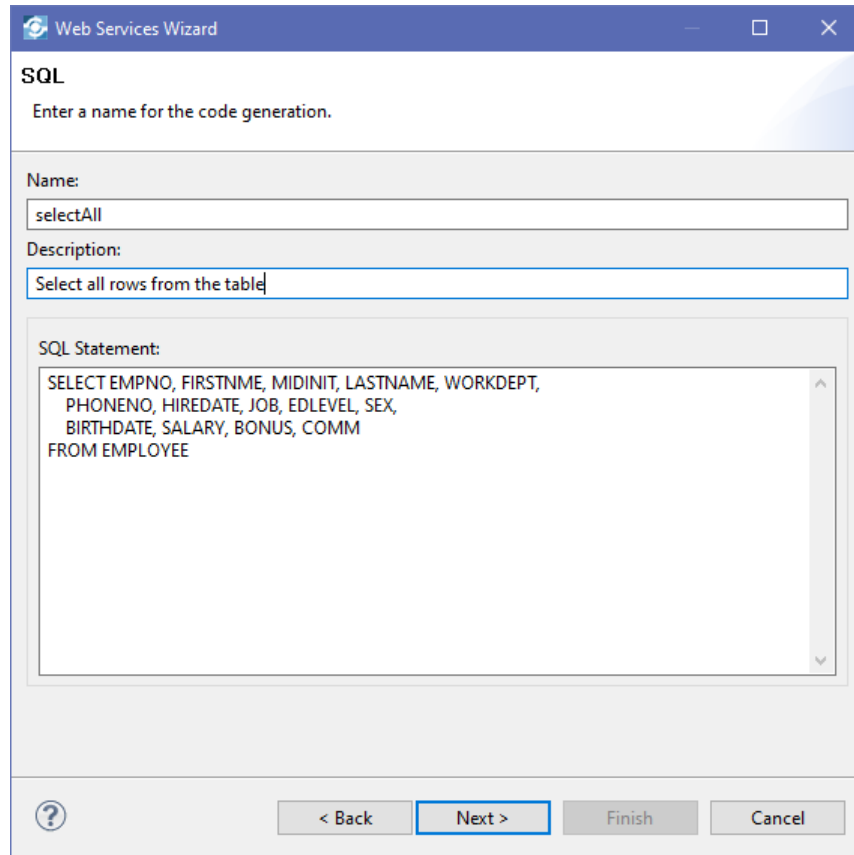
9. On the *Web Service Operation Type* window ensure the radio button beside *Data Integration (REST via z/OS Connect or SOAP)* is selected and click **Next** to continue.



10. 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 virtual table *EMPLOYEE* and press **Next** to continue.



11. The default SQL statement will be displayed on the *SQL* window. Change the name of the operation to ***selectAll*** to indicate that this operation will retrieve all rows from the table. Click **Next** to continue.



The image shows a screenshot of the 'Web Services Wizard' window, specifically the 'SQL' step. The window has a title bar with the text 'Web Services Wizard' and standard window controls. The main content area is titled 'SQL' and contains the instruction 'Enter a name for the code generation.' Below this, there are three input fields: 'Name:' with the value 'selectAll', 'Description:' with the value 'Select all rows from the table', and 'SQL Statement:' with a text area containing the following SQL query:

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,  
       PHONENO, HIREDATE, JOB, EDLEVEL, SEX,  
       BIRTHDATE, SALARY, BONUS, COMM  
FROM EMPLOYEE
```

 At the bottom of the window, there are four buttons: a help button (question mark icon), '< Back', 'Next >' (which is highlighted with a blue border), 'Finish', and 'Cancel'.

Web Services Wizard

SQL

Enter a name for the code generation.

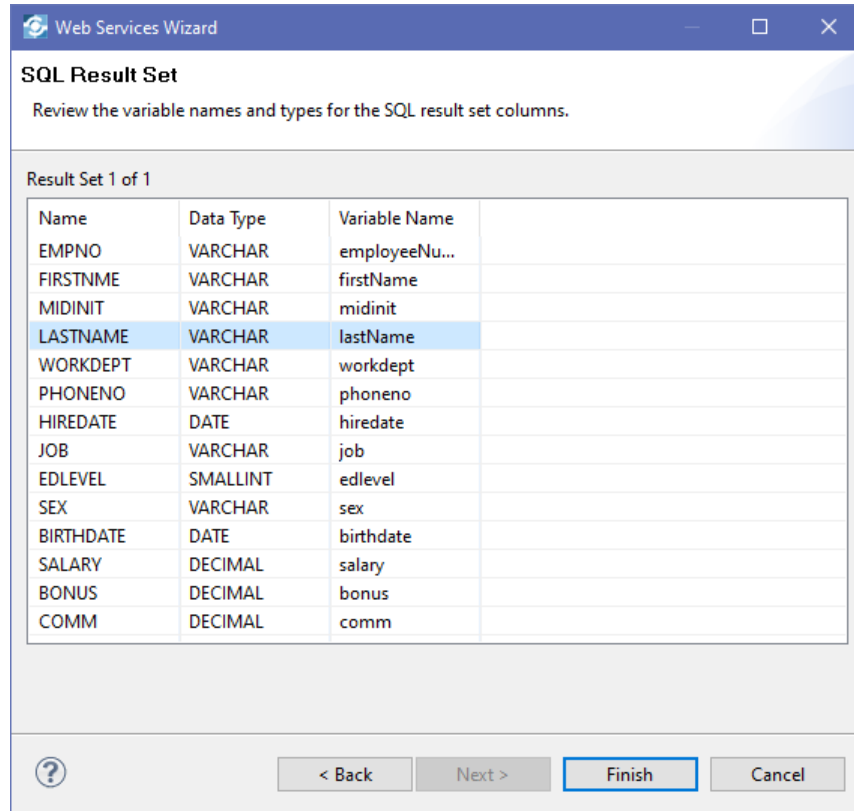
Name:
selectAll

Description:
Select all rows from the table

SQL Statement:
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
 PHONENO, HIREDATE, JOB, EDLEVEL, SEX,
 BIRTHDATE, SALARY, BONUS, COMM
FROM EMPLOYEE

? < Back Next > Finish Cancel

12. The next window to be displayed will show the results that will be returned when the operation is executed. Change the *Variable names* for *EMPNO*, *FIRSTNME* and *LASTNAME* to **employeeNumber**, **firstName** and **lastName**.



The screenshot shows the 'Web Services Wizard' window, specifically the 'SQL Result Set' step. The window title is 'Web Services Wizard'. Below the title bar, it says 'SQL Result Set' and 'Review the variable names and types for the SQL result set columns.' The main area is titled 'Result Set 1 of 1' and contains a table with columns: Name, Data Type, and Variable Name. The table lists various database columns and their corresponding variable names. The 'LASTNAME' row is highlighted in blue. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), and 'Cancel'. A help icon (?) is also present on the left.

Name	Data Type	Variable Name
EMPNO	VARCHAR	employeeNu...
FIRSTNME	VARCHAR	firstName
MIDINIT	VARCHAR	midinit
LASTNAME	VARCHAR	lastName
WORKDEPT	VARCHAR	workdept
PHONENO	VARCHAR	phoneno
HIREDATE	DATE	hiredate
JOB	VARCHAR	job
EDLEVEL	SMALLINT	edlevel
SEX	VARCHAR	sex
BIRTHDATE	DATE	birthdate
SALARY	DECIMAL	salary
BONUS	DECIMAL	bonus
COMM	DECIMAL	comm

Tech-Tip: This window is useful for providing more JSON compatible property names in the RESTful request and response messages.

13. Click **Finish** to continue.

14. Use the *Create Operation* wizard under *EMPLOYEE* to create a new operation. Select the *EMPLOYEE* virtual table and click **Next**. This operation should be named *insertEmployee* and the *SQL Statement* should be changed to

```
INSERT INTO EMPLOYEE
(EMPNO,FIRSTNME,MIDINIT,LASTNAME,WORKDEPT,PHONENO,HIREDATE,JOB,
EDLEVEL, SEX, BIRTHDATE, SALARY, BONUS, COMM) VALUES (?,?,?,?,?, ?,?,?,?,?,?,?)
```

New Web Service Operation Wizard

SQL

Enter a name for the code generation.

Name:
insertEmployee

Description:
Insert a new employee

SQL Statement:
INSERT INTO EMPLOYEE
(EMPNO,FIRSTNME,MIDINIT,LASTNAME,WORKDEPT,PHONENO,HIREDATE,JOB, EDLEVEL, SEX,
BIRTHDATE, SALARY, BONUS, COMM)
VALUES (?,?,?,?,?, ?,?,?,?,?,?,?)

< Back **Next >** Finish Cancel

15. 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 value of variable name in the *Name* column and change the contents of the *Name* column as shown below. The names entered on this screen will be used for the JSON property names in the request message. So, they do not have to match exactly what is shown. Click **Next** to continue.

SQL Inputs
Define the inputs to the SQL statement here.

Inputs:

Name	Data Type	Default Value
employeeNumber	VARCHAR	
firstName	VARCHAR	
middleInit	VARCHAR	
lastName	VARCHAR	
department	VARCHAR	
phoneNumber	VARCHAR	
hireDate	VARCHAR	
job	VARCHAR	
educationLevel	VARCHAR	
sex	VARCHAR	
birthDate	VARCHAR	
salary	VARCHAR	
bonus	VARCHAR	
commission	VARCHAR	

Buttons: Add, Delete, Move Up, Move Down

Navigation: ? < Back Next > Finish Cancel

16. No results are returned on an insert, so click **Finish** on the *SQL Result Set* window.

17. Use the *Create Operation* wizard under *EMPLOYEE* to create a new operation. Select the *EMPLOYEE* virtual table and click **Next**. This operation should be named *selectEmployee* and the *SQL Statement* should be changed to:

SELECT * FROM EMPLOYEE WHERE EMPNO=?

The screenshot shows a 'New Web Service Operation Wizard' window. The title bar says 'New Web Service Operation Wizard'. The main area is titled 'SQL' and contains the instruction 'Enter a name for the code generation.' Below this are three input fields: 'Name:' with the value 'selectEmployee', 'Description:' with the value 'Select an employee by employee number', and 'SQL Statement:' with the value 'SELECT * FROM EMPLOYEE WHERE EMPNO=?'. At the bottom, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

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

Name	Data Type	Default Value
employeeNum...	VARCHAR	

19. The columns that will be returned are displayed on the *SQL Result Set* window. Change the Variable Names as shown below (these names will be JSON property names in the response message) and then click **Finish** to continue.

Name	Data Type	Variable Name
EMPNO	VARCHAR	employeeNu...
FIRSTNME	VARCHAR	firstName
MIDINIT	VARCHAR	middleInit
LASTNAME	VARCHAR	lastName
WORKDEPT	VARCHAR	department
PHONENO	VARCHAR	phone
HIREDATE	DATE	hireDate
JOB	VARCHAR	job
EDLEVEL	SMALLINT	educationLevel
SEX	VARCHAR	sex
BIRTHDATE	DATE	birthdate
SALARY	DECIMAL	salary
BONUS	DECIMAL	bonus
COMM	DECIMAL	commission

20. Use the *Create Operation* wizard under *EMPLOYEE* create a new operation. Select the *EMPLOYEE* virtual table and click **Next**. This operation should be named ***updateEmployee*** and the *SQL Statement* should be changed to:

UPDATE EMPLOYEE SET BONUS=? WHERE EMPNO=?

The screenshot shows the 'New Web Service Operation Wizard' dialog box, specifically the 'SQL' step. The title bar reads 'New Web Service Operation Wizard'. Below the title bar, the text 'SQL' is displayed, followed by the instruction 'Enter a name for the code generation.' The dialog has three main input fields: 'Name:', 'Description:', and 'SQL Statement:'. The 'Name:' field contains the text 'updateEmployee'. The 'Description:' field contains the text 'Update an employee commision'. The 'SQL Statement:' field contains the text 'UPDATE EMPLOYEE SET BONUS=? WHERE EMPNO=?'. At the bottom of the dialog, there are four buttons: a help button (question mark icon), '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

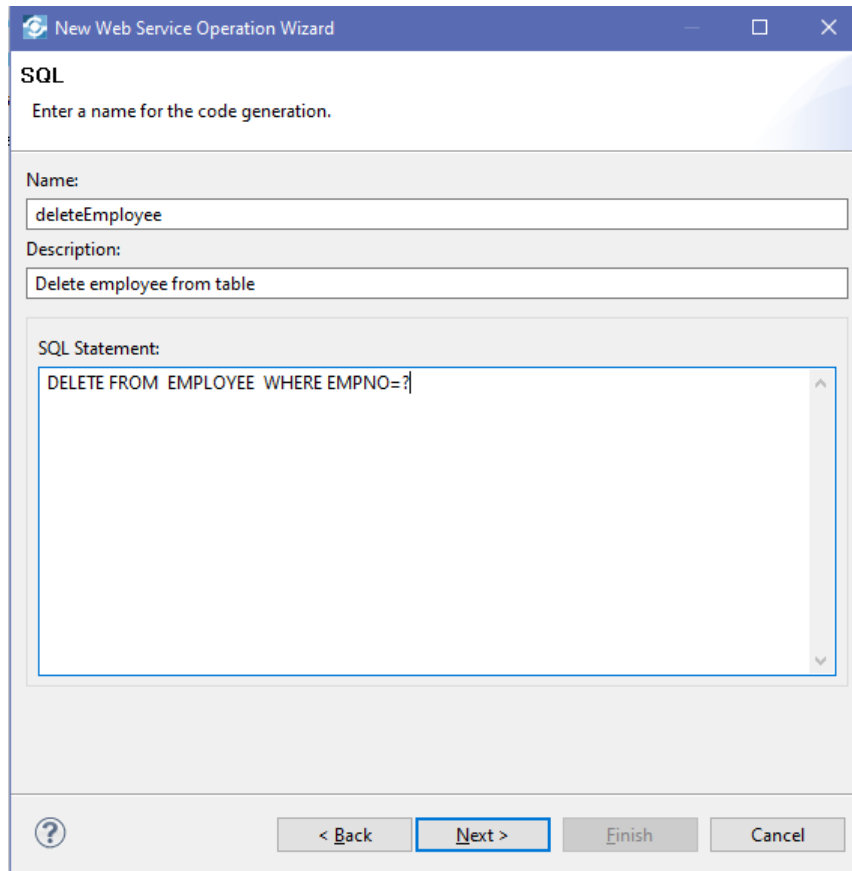
21. Click **Next** to continue. Since a *WHERE* clause has been added with variables, providing meaningful names for these variables will be required. The next window to be displayed, *SQL Inputs*, will give us a chance to give meaningful names. On this window, click on the values of variable name in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

Name	Data Type	Default Value
bonus	VARCHAR	
employeeNumber	VARCHAR	

22. No result set for this operation, so click **Finish** on the *SQL Result Set* window.

23. Use the *Create Operation* wizard under *EMPLOYEE* create a new operation. Select the *EMPLOYEE* virtual table and click **Next**. This operation should be named *deleteEmployee* and the *SQL Statement* should be changed to:

DELETE FROM EMPLOYEE WHERE EMPNO=?



The screenshot shows the 'New Web Service Operation Wizard' dialog box, specifically the 'SQL' tab. The dialog has a title bar with a question mark icon and the text 'New Web Service Operation Wizard'. Below the title bar, the text 'SQL' is displayed, followed by the instruction 'Enter a name for the code generation.' The dialog contains three input fields: 'Name:' with the value 'deleteEmployee', 'Description:' with the value 'Delete employee from table', and 'SQL Statement:' with the value 'DELETE FROM EMPLOYEE WHERE EMPNO=?'. At the bottom of the dialog, there are four buttons: a help button (question mark icon), '< Back', 'Next >', and 'Finish'. The 'Next >' button is highlighted with a blue border.

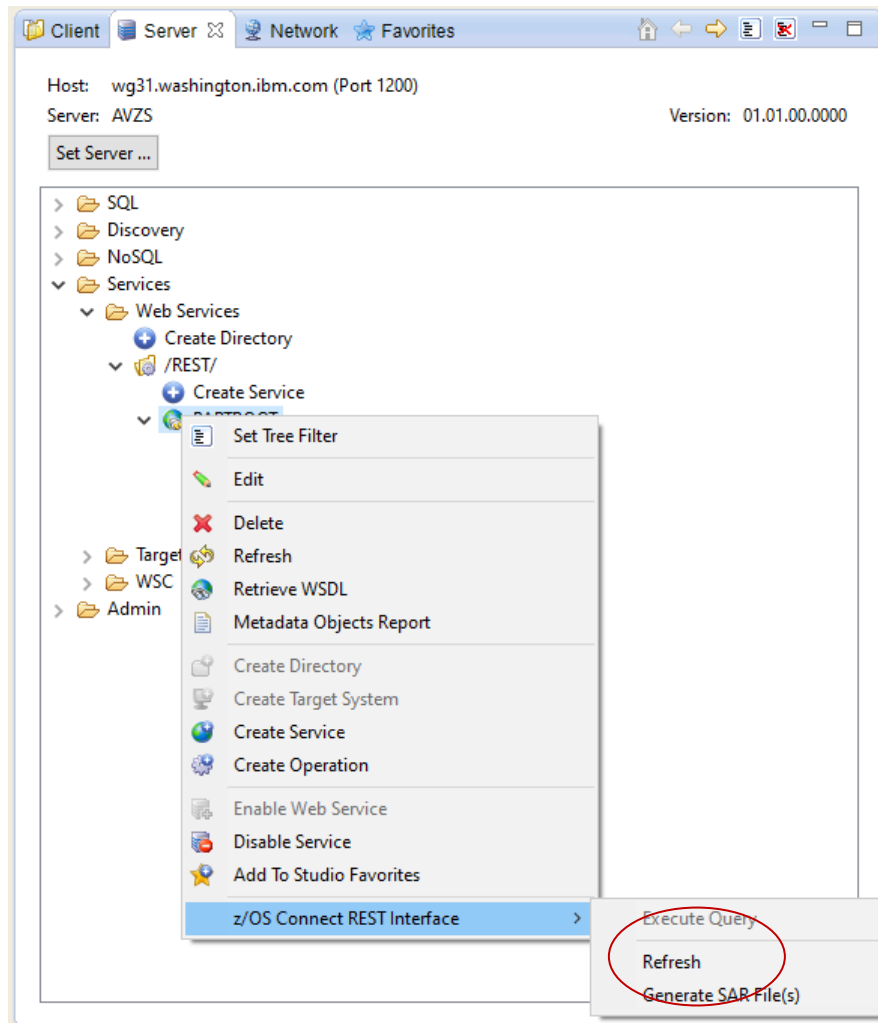
24. Click **Next** to continue. Since a *WHERE* clause has been added with variable, providing a meaningful name for this variable will be required. The next window to be displayed, *SQL Inputs*, will give us a chance to give a meaningful name to the variable. On this window click on the value of variable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

[illegible]

25. No result set for this operation, so click **Finish** on the *SQL Result Set* window.

Deploy the DVM services

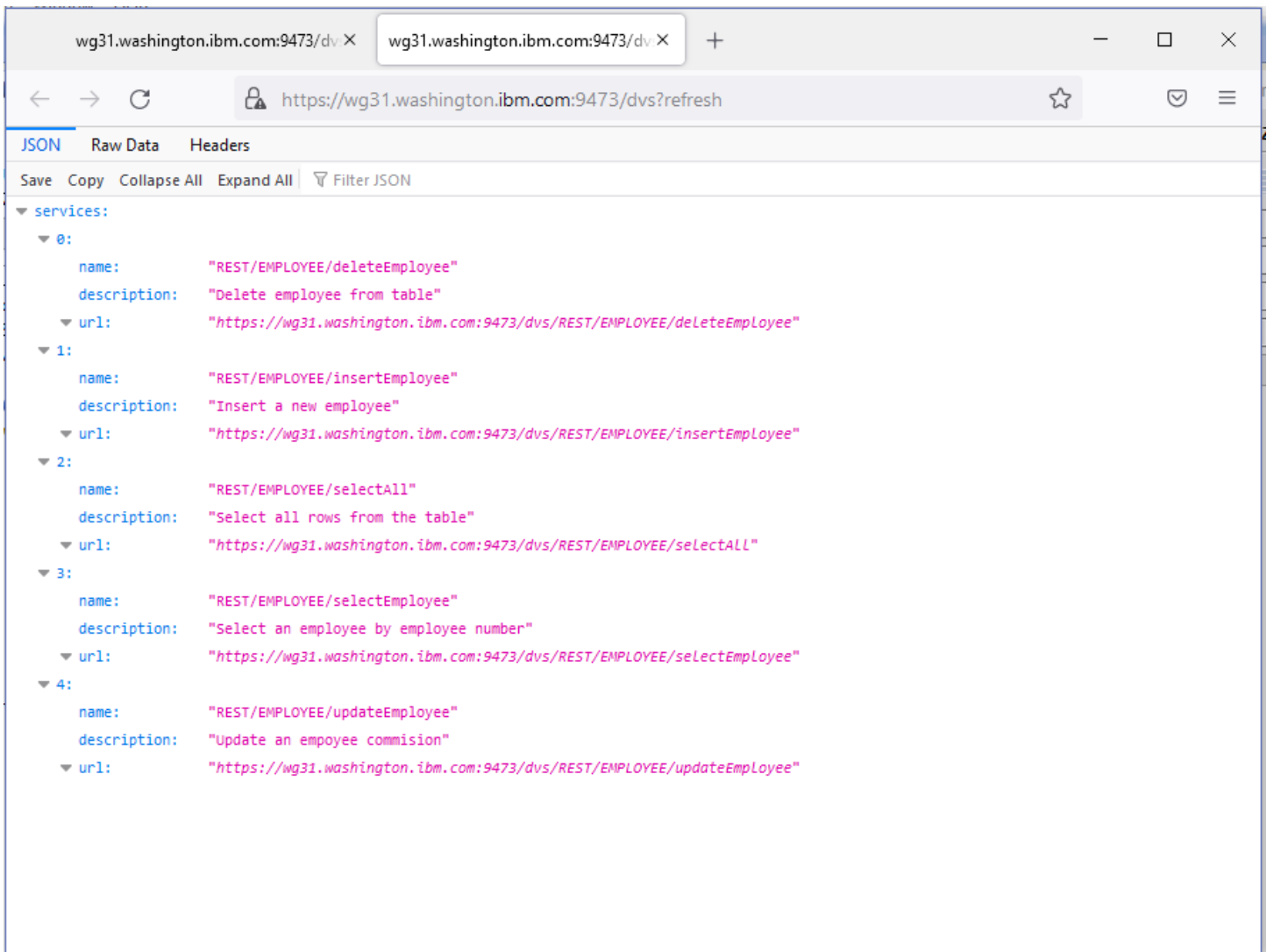
1. These operations need to be deployed to z/OS Connect server. Select *PARTROOT* folder under */REST/* and right mouse button click. Select the *z/OS Connect REST Interface* option then the *Refresh* option. This will install the selected DVM services into the z/OS Connect server as DVM services.



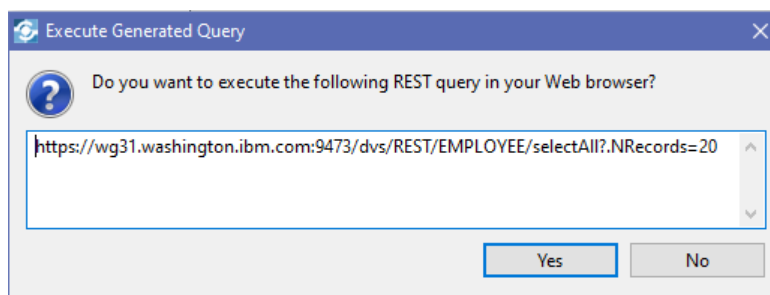
Tech Tip: Operations can be deployed individually by selecting the specific operation and right mouse button clicking and selecting **Refresh**.

Tech Tip: You may be challenged by Firefox because the digital certificate used by the Liberty z/OS server is self-signed Click the **Advanced** button to continue. Scroll down and then click on the **Accept the Risk and Continue** button. Next you may see a prompt you for a userid and password. If you do see the prompt, enter the username **USER1** and password **USER1** and click **OK**.

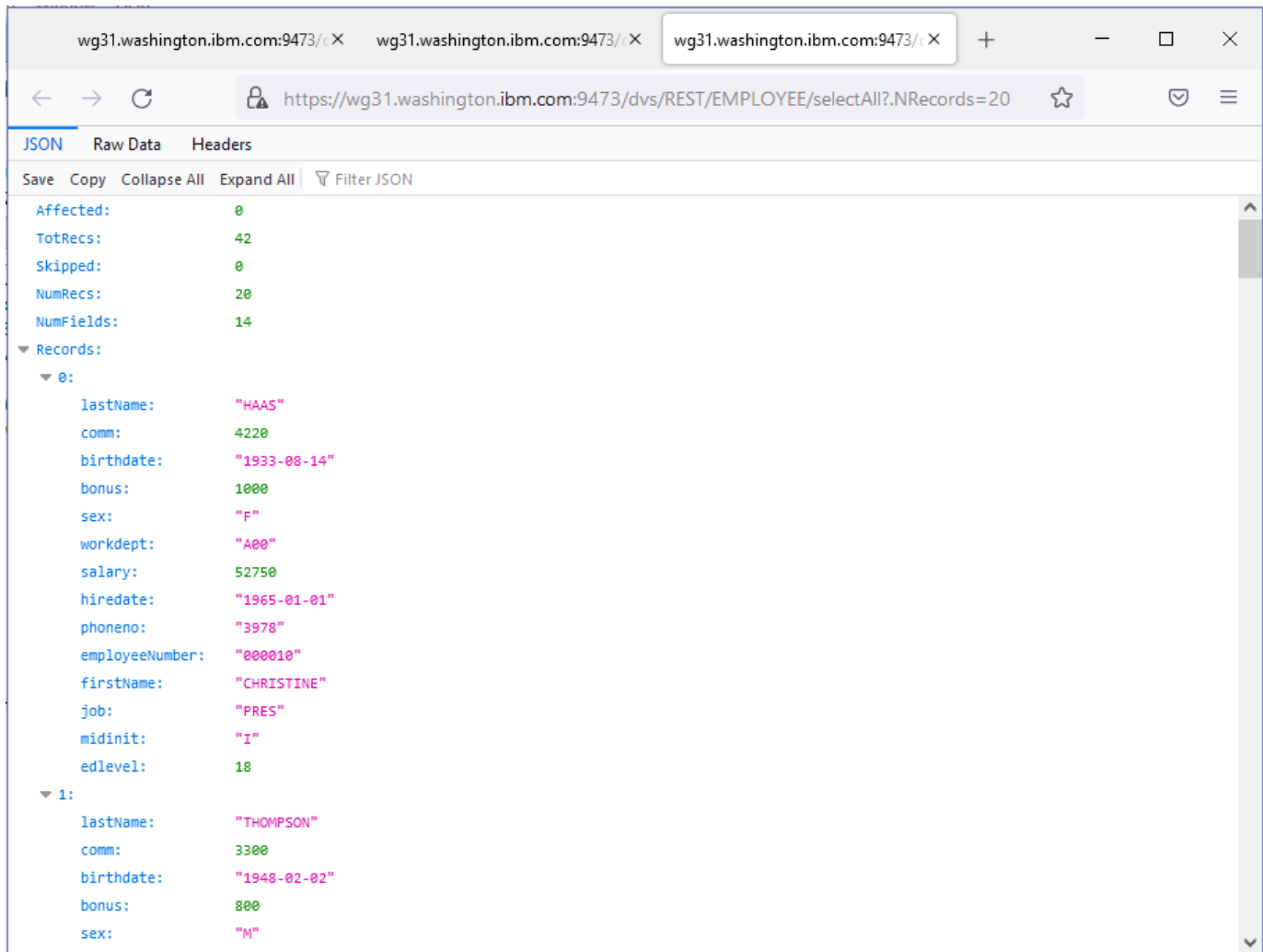
2. When finished all the operations should be displayed as DVM services in the z/OS Connect server by entering URL **<https://wg31.washington.ibm.com:9473/dvs>**.



3. A subset of these z/OS Connect services can now be tested using the DVM Data Manager Studio (only the services that do selects) Select the *selectAll* 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. Click **OK** to continue.



5. A web browser session should open with results like the one below.

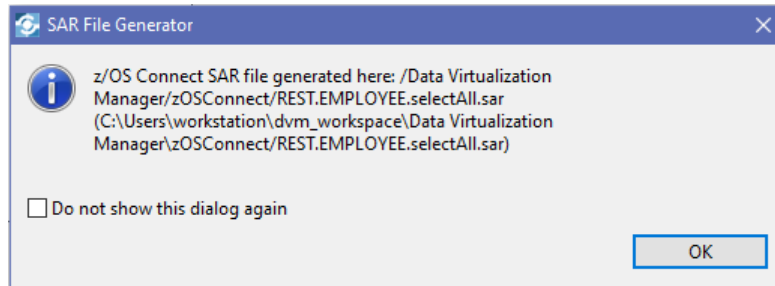


Tech-Tip: The above results show some new fields in the response messages. Their meanings are provided below:

Affected:	The number of records deleted, updated or inserted by this request.
TotRecs:	The number of records found.
Skipped:	The number of records skipped
NumRecs:	The number of records returned.
NumFields:	The number of fields returned for each record.

Export the DVM services as z/OS Connect service archive files

Finally, the Service Archive (SAR) files need to be exported from the DVM Studio for use in the z/OS Connect API Editor. Select EMPLOYEE and right mouse button click. Select the z/OS Connect REST Interface option then the Generate SAR File(s) option. A pop-up window will appear for each SAR file to be exported, click OK on each.



This exports the SAR files to a subdirectory in the DVM Toolkit's workspace directory, e.g.,

Tech-Tip: The directory where the SAR file is exported may be different on your system. Make a note of this directory name so you will know from where to import the SAR file later. On some images, this directory will be *C:\Users\administrator\dvm_workspace\Data Virtualization Manager\zOSConnect*.

C:\Users\workstation\dvm_workspace\Data Virtualization Manager\zOSConnect. This pop-up will be repeated for each operation. This directory will be referenced in a latter section of this exercise.

Using z/OS Connect APIs to access DVM services

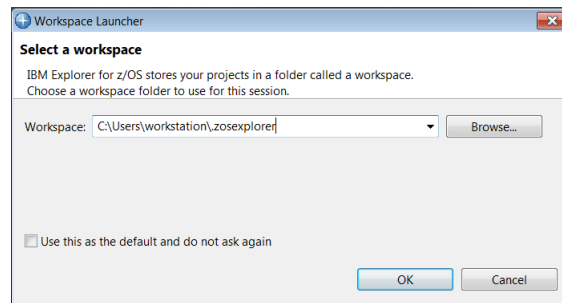
Connect to a z/OS Connect Server

Begin by establishing a connection to the 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.

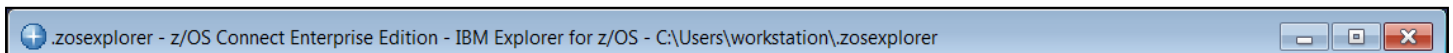
Tech-Tip: Windows desktop tools can be opened either by double clicking the icon or by selecting the icon and right mouse button clicking and then selecting the *Open* option.

2. You will be prompted for a workspace:



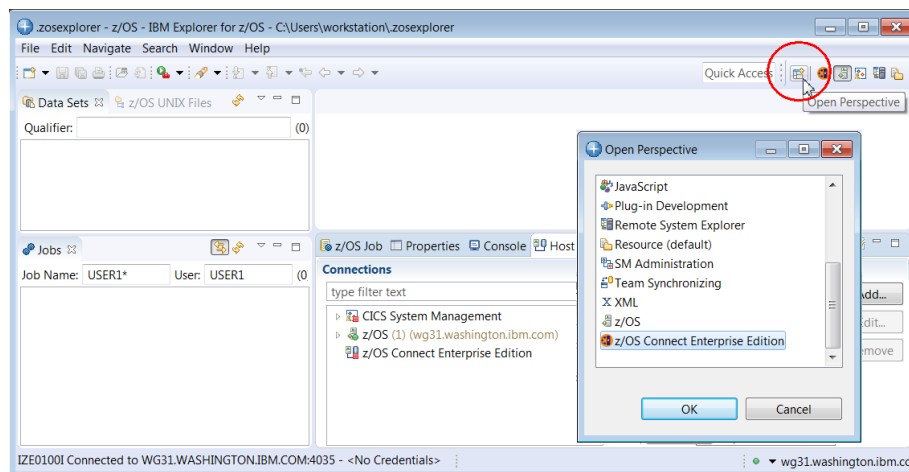
Take the default value by clicking **OK**.

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:

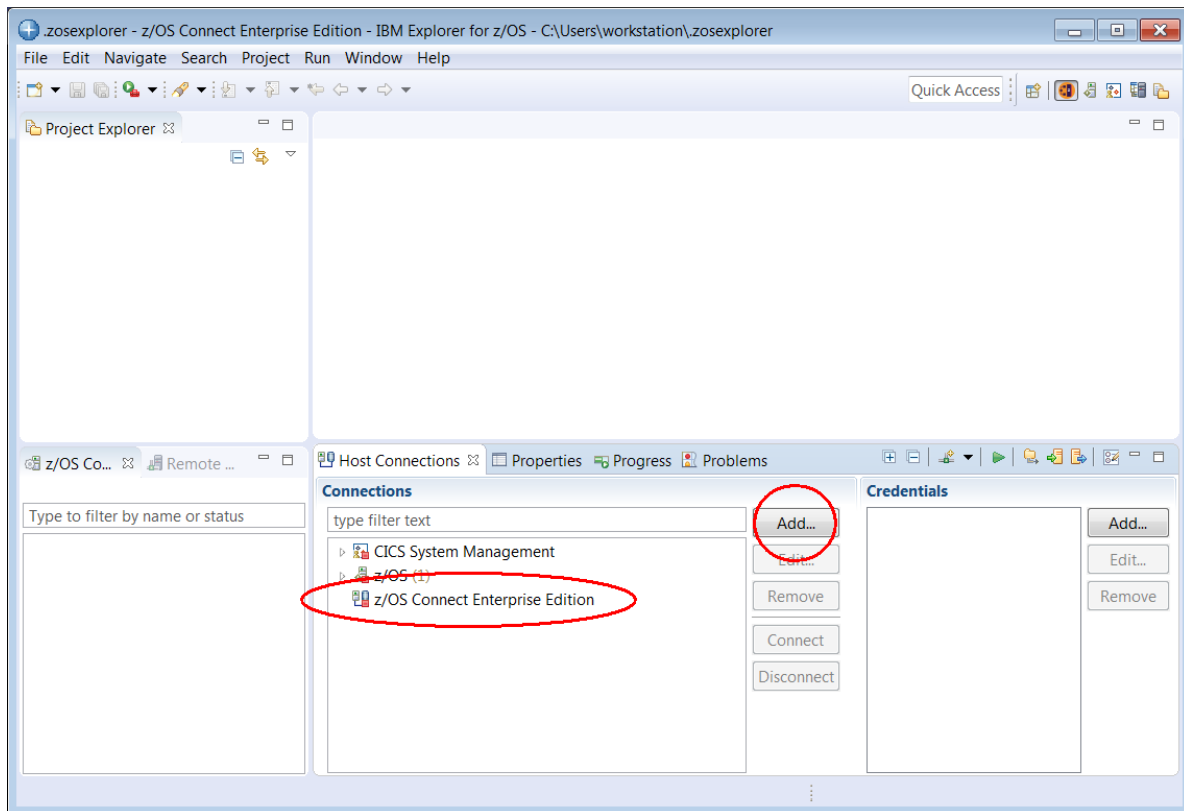


N.B. If a *Welcome* screen is displayed then click the white X beside *Welcome* to close this view.

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.



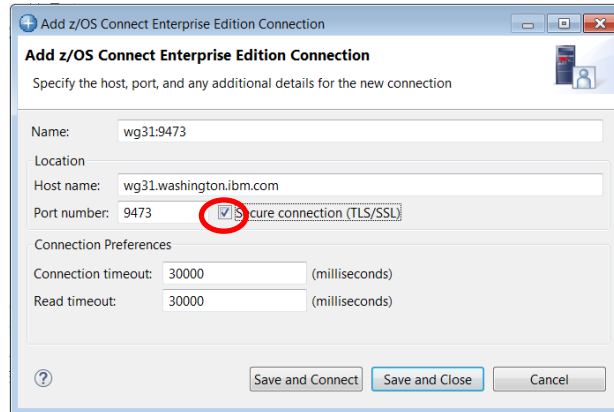
Tech-Tip: Eclipse based development tools like z/OS Explorer; provide a graphical interface consisting of multiple views within a single window.

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).

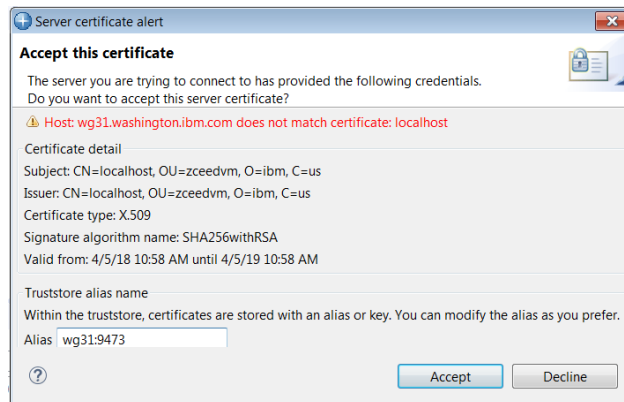
At any time, a specific view can be enlarged to fill the entire window by double clicking in the view's title bar. Double clicking in the view's title bar will be restored the original arrangement. If a z/OS Explorer view is closed or otherwise disappears, the original arrangement can be restored by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



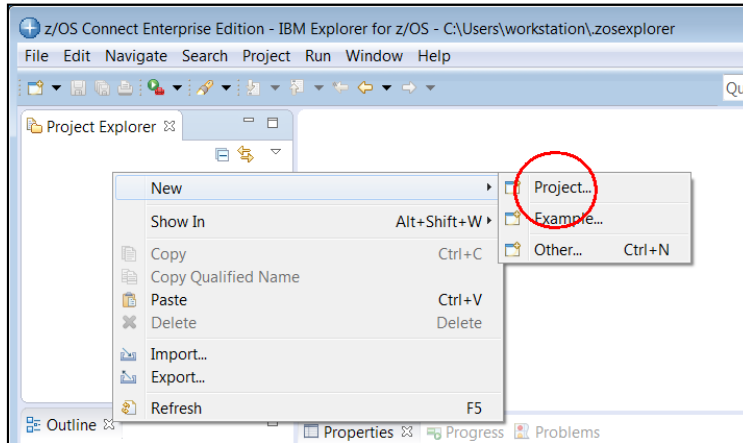
7. On the *z/OS Connect Enterprise Edition – User ID* required screen, create new credentials for a *User ID* of **USER1** and for *Password or Passphrase* enter **USER1**'s password. Click **OK** to continue.
8. Click the **Accept** button on the *Server certificate alert – Accept this certificate*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



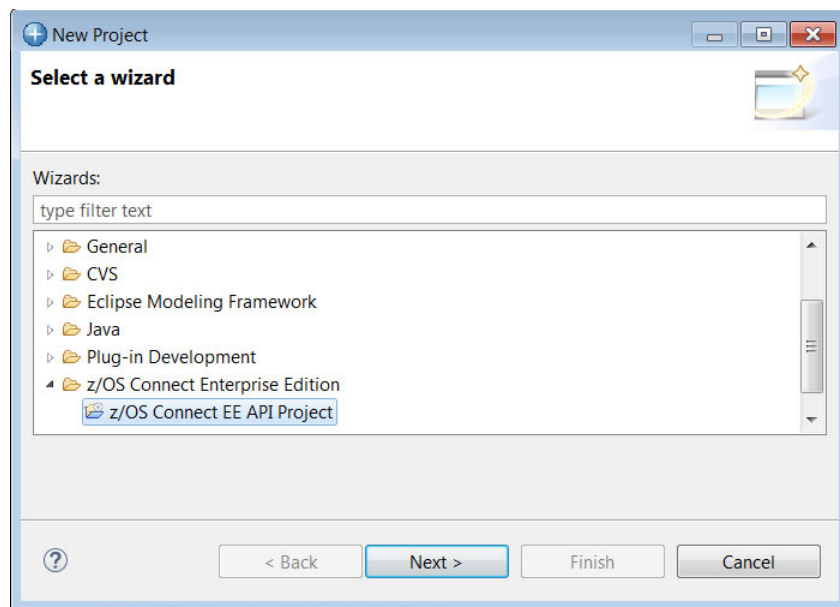
9. The status icon beside **wg31:9473** should now be a green circle with a lock. This shows that a secure connection has been established between the *z/OS Explorer* and the *z/OS Connect* server. A red box indicates that no connection exists.

Create the Db2 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 API Project* and then click the **Next** button.



3. Enter **DB2API** for the *Project name*. Be sure the *API name* is set to **db2** and the *Base path* is set to **/db2**

New Project

z/OS Connect EE API Project
Create a new z/OS Connect EE API project.

Project name:

API name:

Base path:

Description:

Note: The Base path name of **/db2** is used to distinguish a request for this API from other APIs in the same server. It can be any value as long as the value is unique within the server. The same is true of any sub path names added to the base path. Sub path names are used to distinguish one method/service from another within an API.

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

db2 API

API Editor

Describe your API

Name: Description:

Base path:

Version:

Contact Information

Path

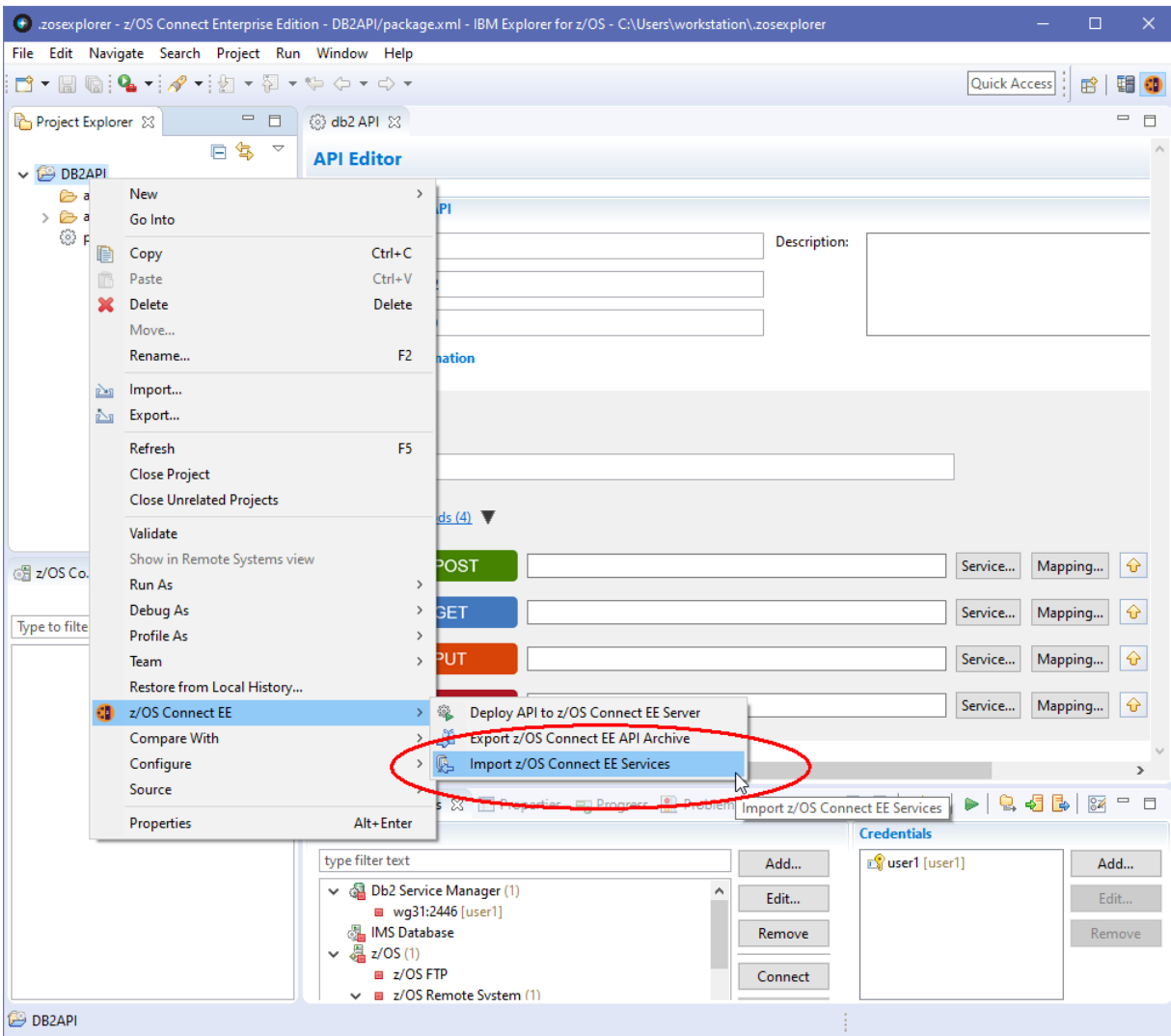
Methods (4)

POST	<input type="text"/>	Service...	Mapping...	<input type="button" value="Up"/>
GET	<input type="text"/>	Service...	Mapping...	<input type="button" value="Up"/>
PUT	<input type="text"/>	Service...	Mapping...	<input type="button" value="Up"/>
DELETE	<input type="text"/>	Service...	Mapping...	<input type="button" value="Up"/>

Tech-Tip: If the API Editor view is closed, it can be reopened by double clicking the *package.xml* file in the API project.

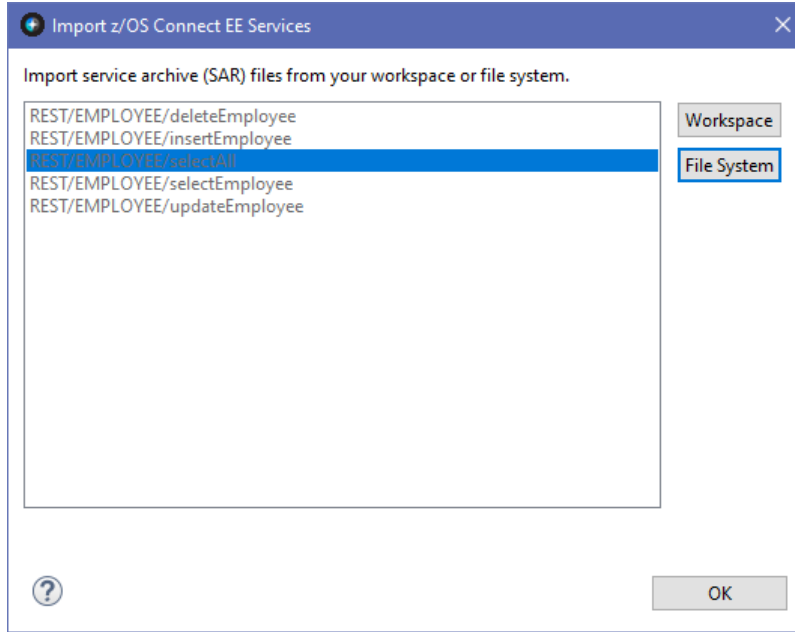
Import the SAR files generated by the DVM Studio

1. In the z/OS Explorer in the *z/OS Connect Enterprise Edition* perspective in the *Project Explorer* view (upper left), right-click on the *DB2API* project, then select *z/OS Connect* and then *Import z/OS Connect Services* (see below):

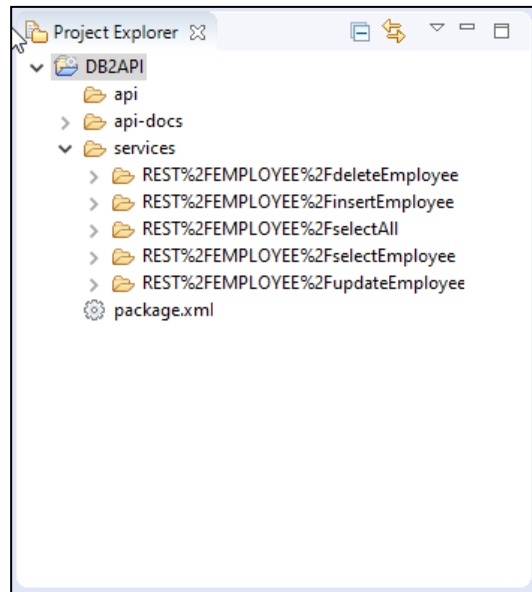


Tech-Tip: Remember from step 6 on page 32, the directory where the SAR file is to be imported from may be different on your system. On some images, this directory will be *C:\Users\administrator\dvm_workspace\Data Virtualization Manager\zOSConnect*.

2. In the *Import z/OS Connect Services* window click on the **File System** button and navigate to directory *C:\Users\workstation\dvm_workspace\Data Virtualization Manager\zOSConnect*. Select all the SAR files and click on the **Open** button. (Hint: use the *Ctrl-A* key sequence to select all the files).
3. The service archive files should appear in the *Import Services* window. Click the **OK** button twice to import them into the workspace.



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



Compose an API for the IMS DVM Rest Services

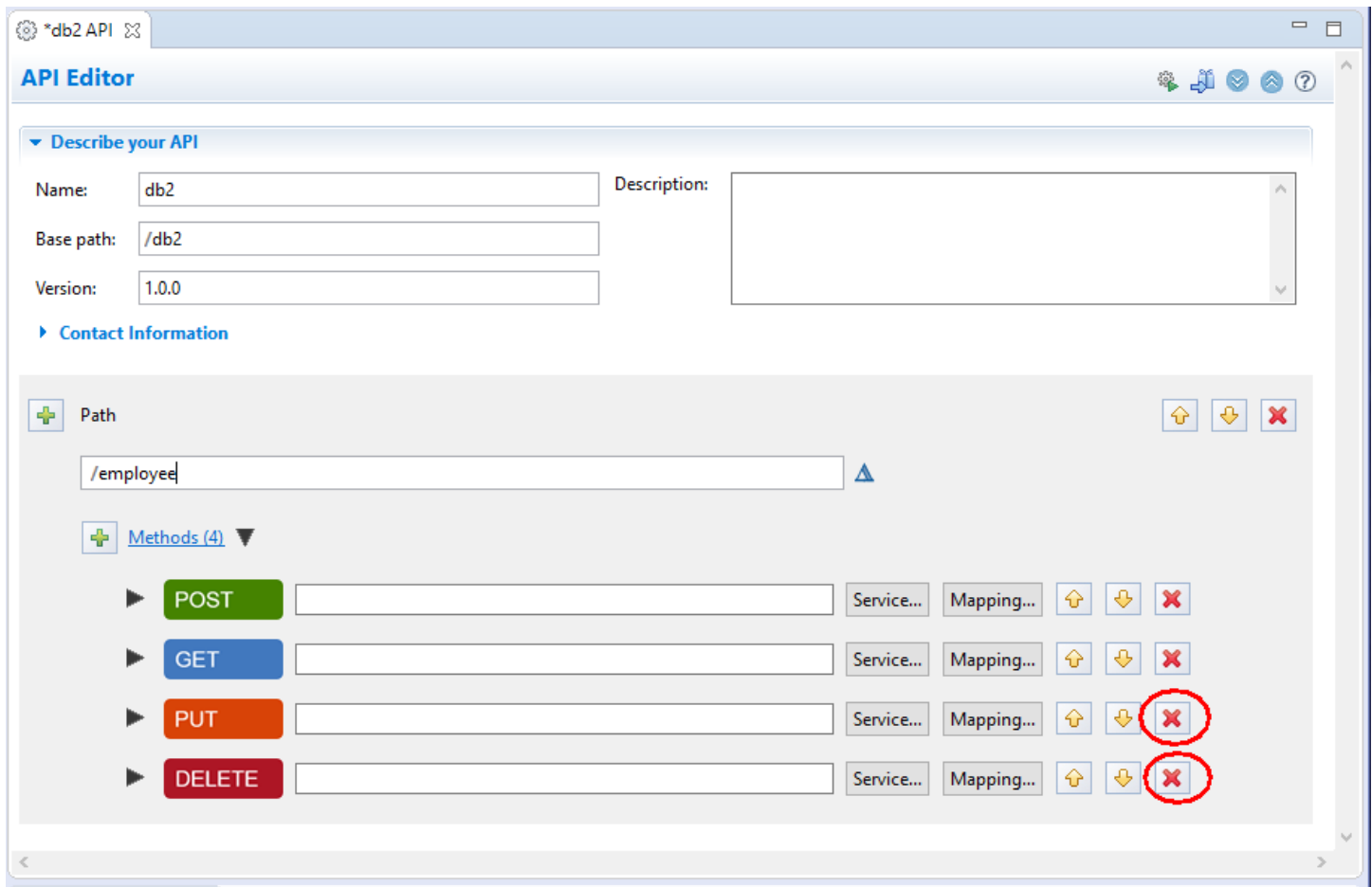
___1. Start by entering a *Path* of */employee* in the *z/OS Connect API Editor* view as shown below:

The screenshot shows the 'API Editor' window for an API named 'db2'. The 'Describe your API' section is expanded, showing the following fields:

- Name: db2
- Base path: /db2
- Version: 1.0.0
- Description: (empty text area)

Below the 'Describe your API' section is the 'Contact Information' section, which is currently collapsed. The main area of the editor shows the 'Path' field set to '/employee'. Below the path field is a section for 'Methods (4)', which lists four methods: POST, GET, PUT, and DELETE. Each method has a corresponding 'Service...' button, a 'Mapping...' button, and three control icons (up, down, and delete).

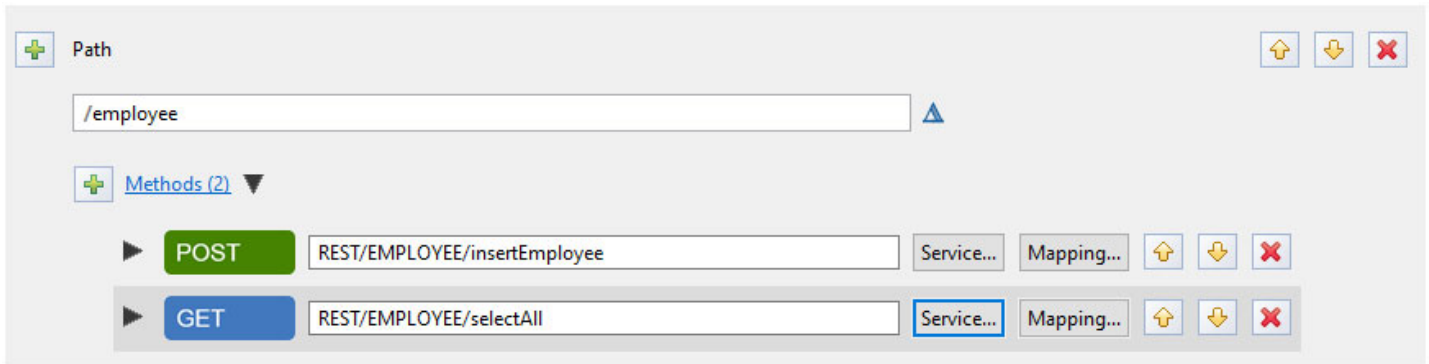
2. The initial API to be added will be when no path or query parameter will be required, the supported HTTP methods will only be the **GET** and **POST** methods. Remove the **PUT** and **DELETE** methods by clicking the red **X** icon to the right of each method.



3. That should leave you with just the **GET** and **POST** methods.

4. Click on the **Service** button to the right of the **POST** method. Then select the *REST/EMPLOYEE/insertEmployee* service from the list of services and click **OK**. This will populate the field to the right of the method.

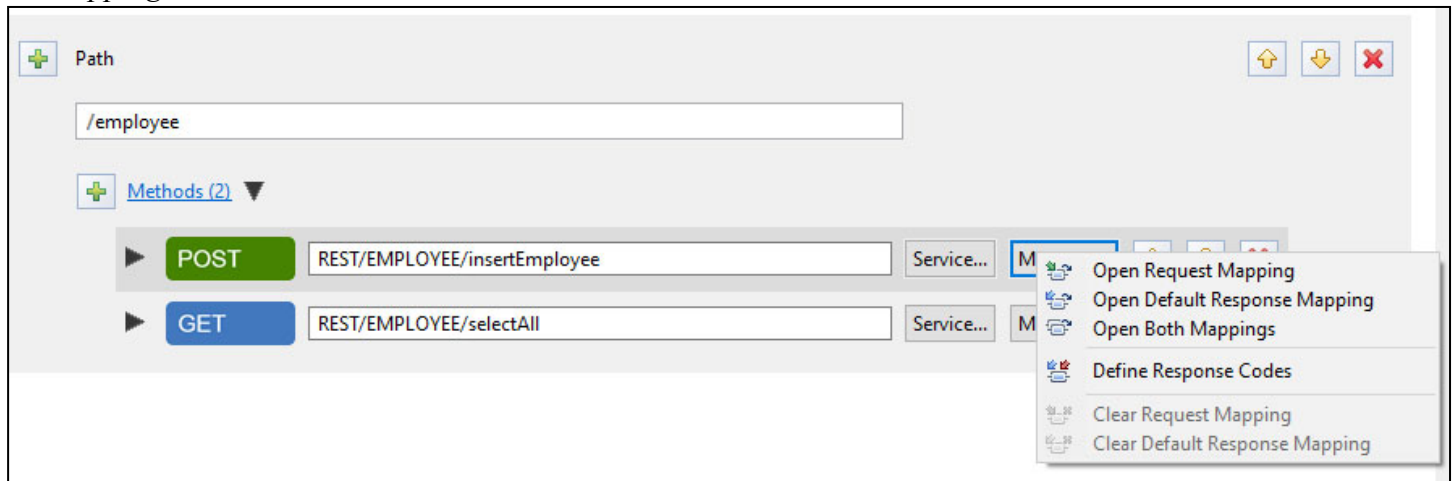
- ___ 5. Click on the **Service** button to the right of the **GET** method. Then select the *REST/EMPLOYEE/selectAll* service from the list of services and click **OK**. This will populate the field to the right of the method.



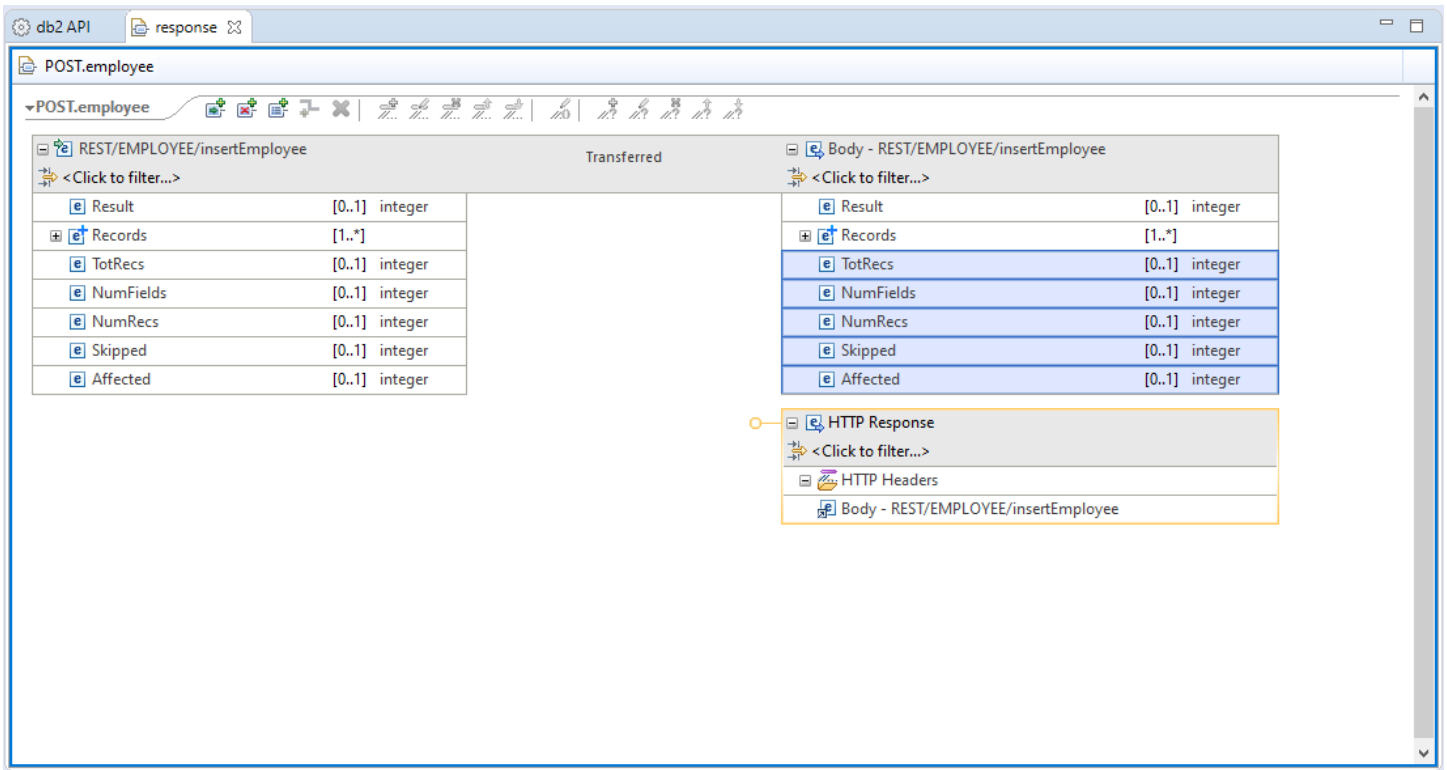
- ___ 6. Save the changes so far by using the key sequence **Ctrl-S**.

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.

- ___ 7. Next, click on the **Mapping** button beside the **POST** method and then select *Open Default Response Mapping*:

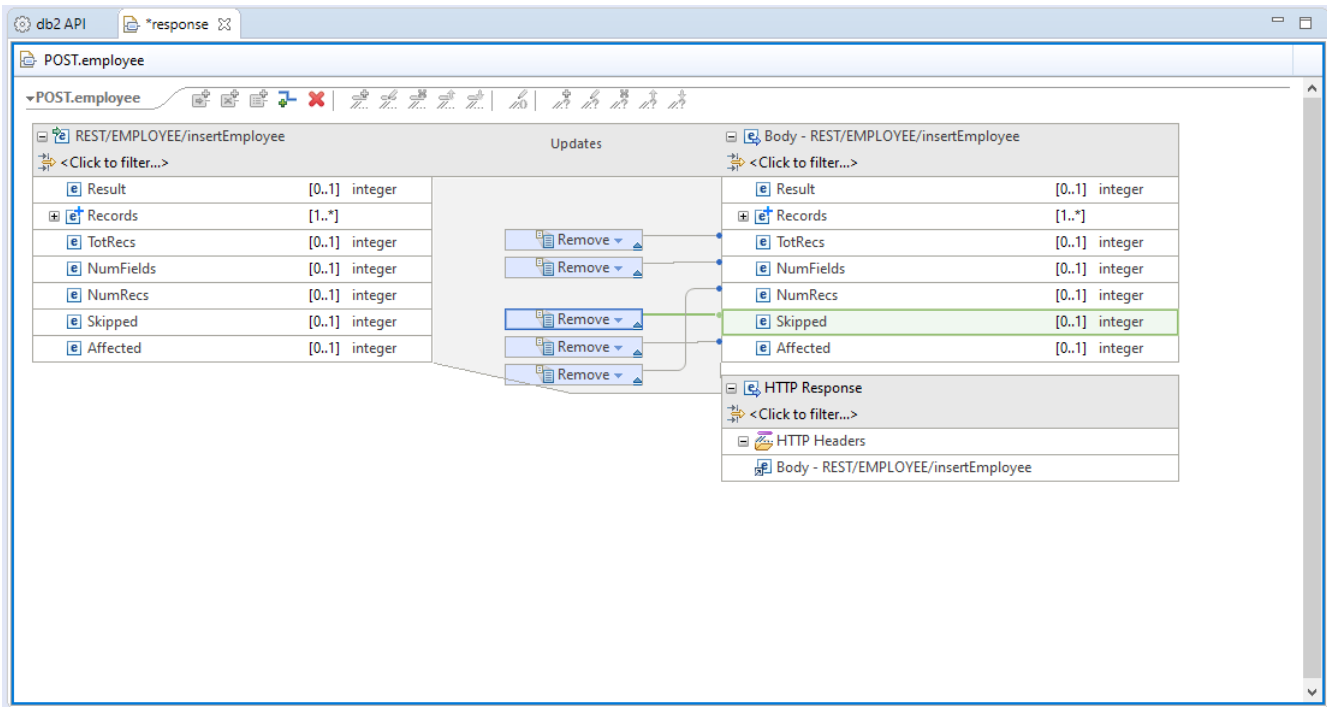


8. Use the left mouse button and draw a dotted line box that **fully** includes the *TotRecs*, *NumFields*, *NumRecs*, *Skipped* and *Affected* fields. When you release the button, these fields should be selected (the background should be blue).

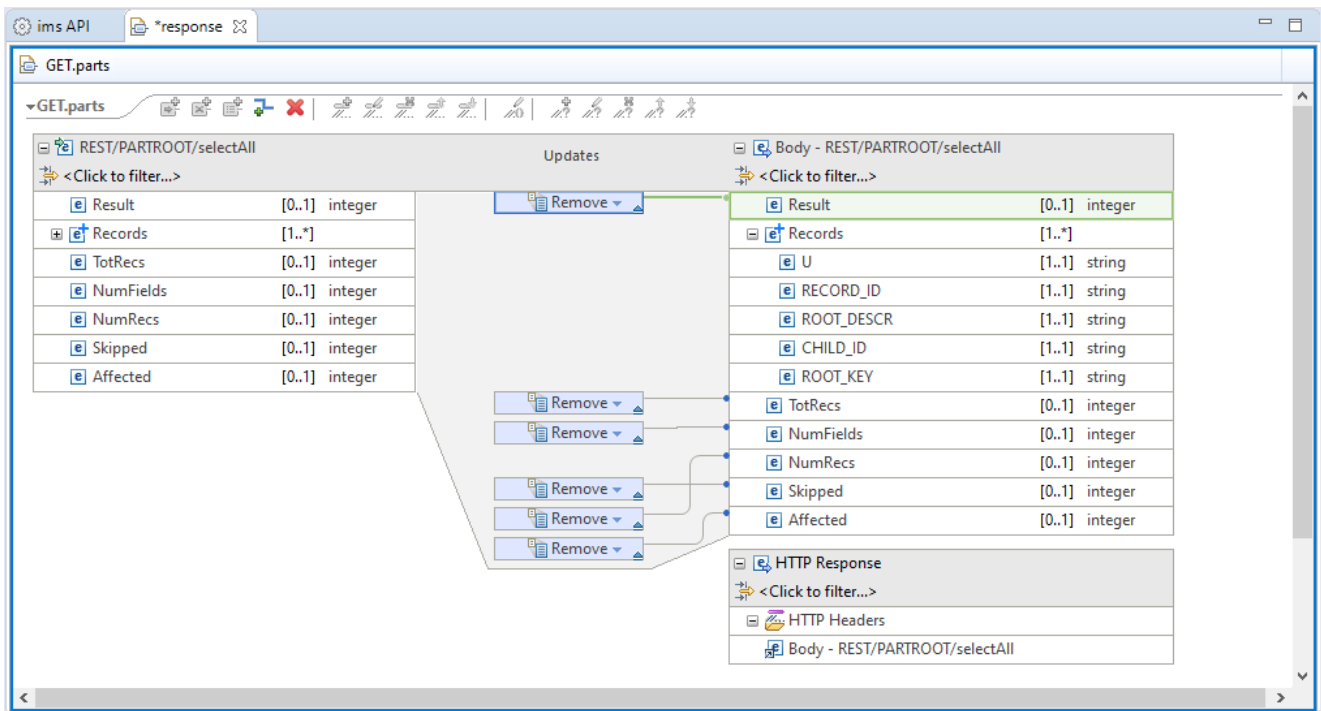


Tech-Tip: This step is being done just to show how fields can be removed from the service interface response message. Response fields like *TotRecs*, *Affected*, *NumRecs*, etc. can be checked like this to set appropriate HTTP response codes. For example, if you were doing a GET (a SELECT) and the value of *NumRecs* was zero, the HTTP response code could be set to *404 – Not Found*.

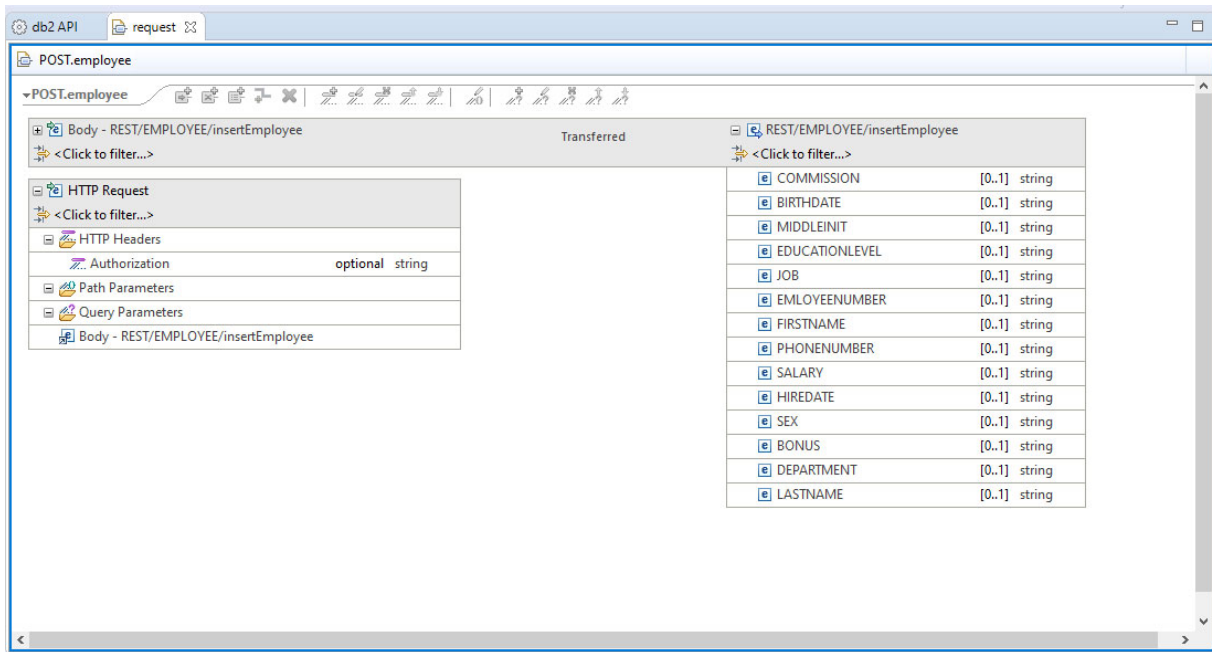
- ___ 9. Right mouse button click on any of the selected fields and select the *Add Remove transform* from the list of options.
- ___ 10. This action generates multiple “Remove” requests (see below) for the selected fields. These fields are not required to be display so they will be removed from the response.



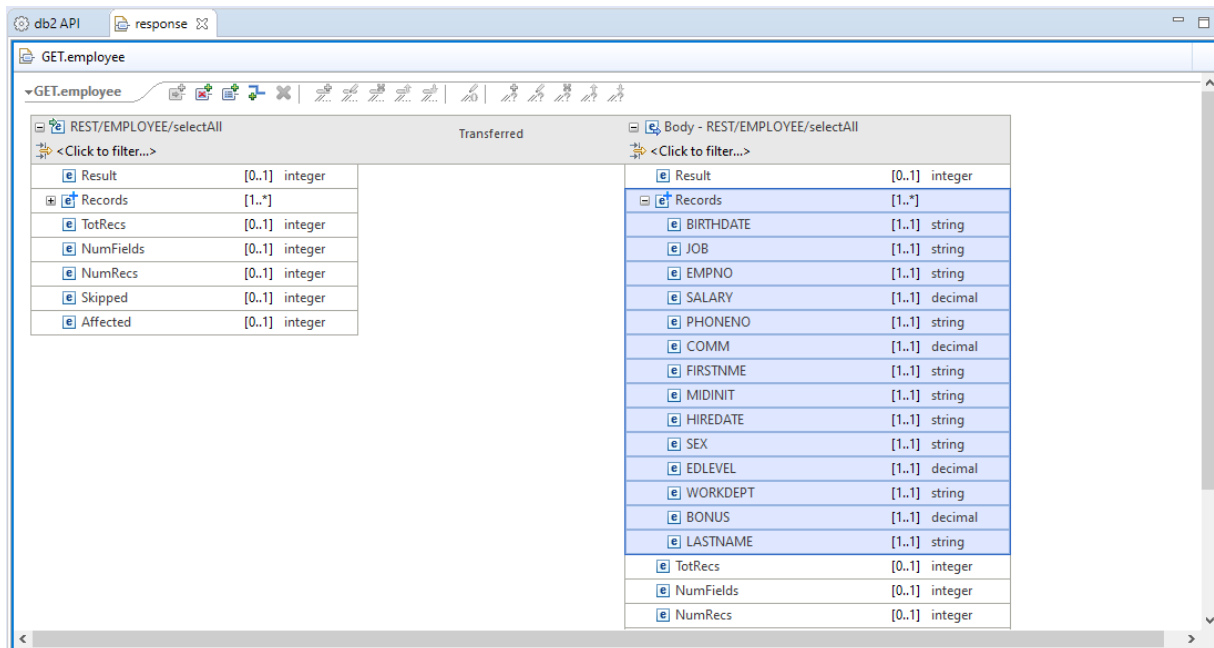
- ___ 11. Select the *Result* field and remove it from the response. If not expanded already, expand the *Records* structure and you should see the ‘columns’ that will be displayed in the response.



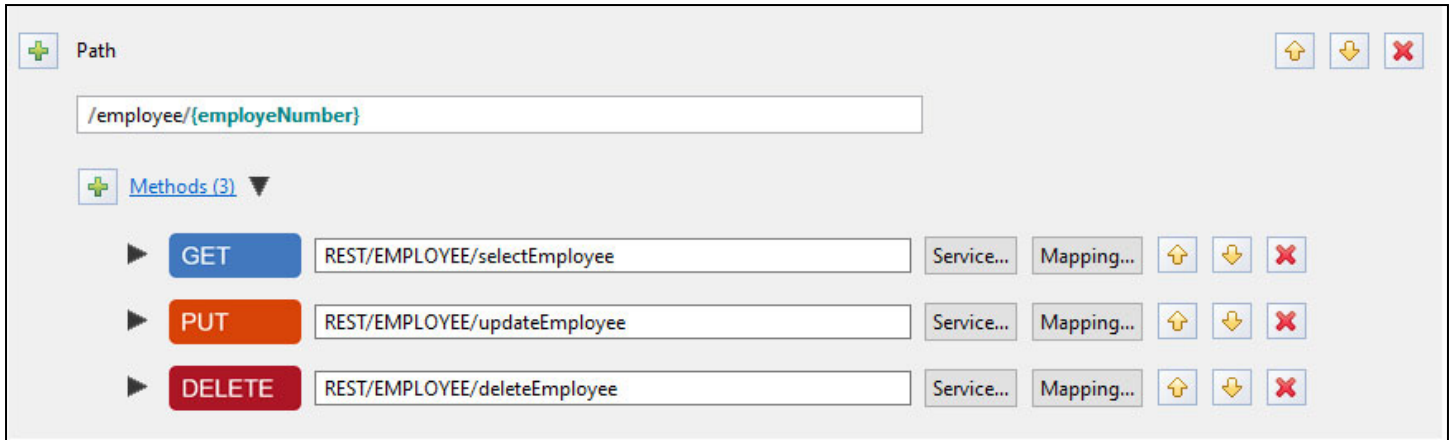
12. Use the **Ctrl-S** key sequence to save all changes and close the *POST.employee* response view.
13. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping* for this method. Notice that the property names you provided earlier are being used. Close the request view.



14. Next, click on the **Mapping** button beside the **GET** method and then select *Open Default Response Mapping*. Expand *Records* in the response and you see the property names that will appear in the response message.

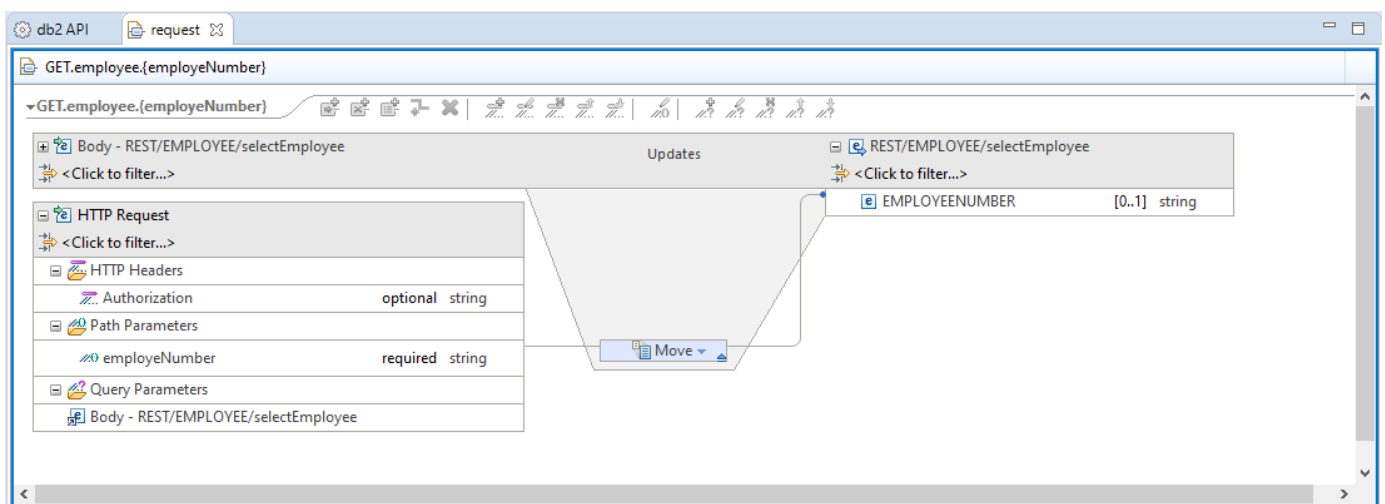


15. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping* for this method. Notice that there are no property names since no request message is required. Close the request view.
16. Next, we want to add a *Path* for the other three DVM services, *selectEmployee*, *updateEmployee* and *deleteEmployee*. Click the plus icon beside *Path* on the z/OS Connect API Editor view to add path */employee/{employeeNumber}* to the API. Remove the **POST** method and associate the **GET** method with service *REST/EMPLOYEE/selectEmployee*, the **PUT** method with service *REST/EMPLOYEE/updateEmployee* and the **DELETE** method with service *REST/EMPLOYEE/deleteEmployee*.



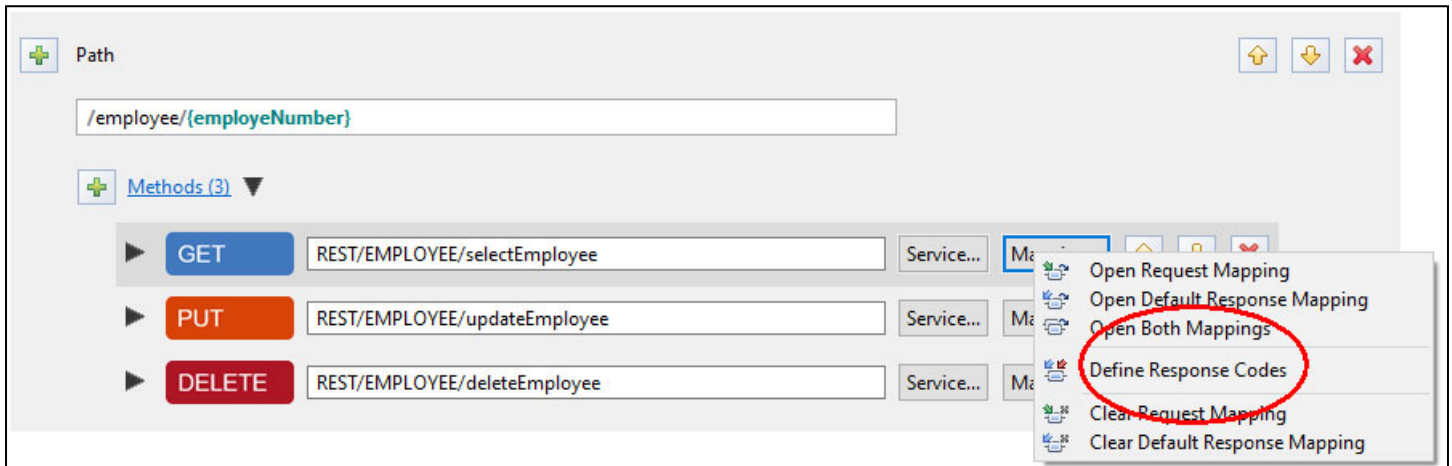
Tech-Tip: Additional *Paths* can be added by clicking the + icon beside *Path* and additional *Methods* can be added by clicking the + icon beside *Methods*.

17. Click on the **Mapping** button beside the **GET** method and then select *Open Request Mapping* for this method. Map the path parameter *employeeNumber* to the *employeeNumber* field in the request message, as shown below. This is done by selecting *employeeNumber* on the left-hand side and dragging it over to *employeeNumber* on the right hand side to make a *Move* connection so the value or contents of the *employeeNumber* path parameter are moved into *employeeNumber* field of the request

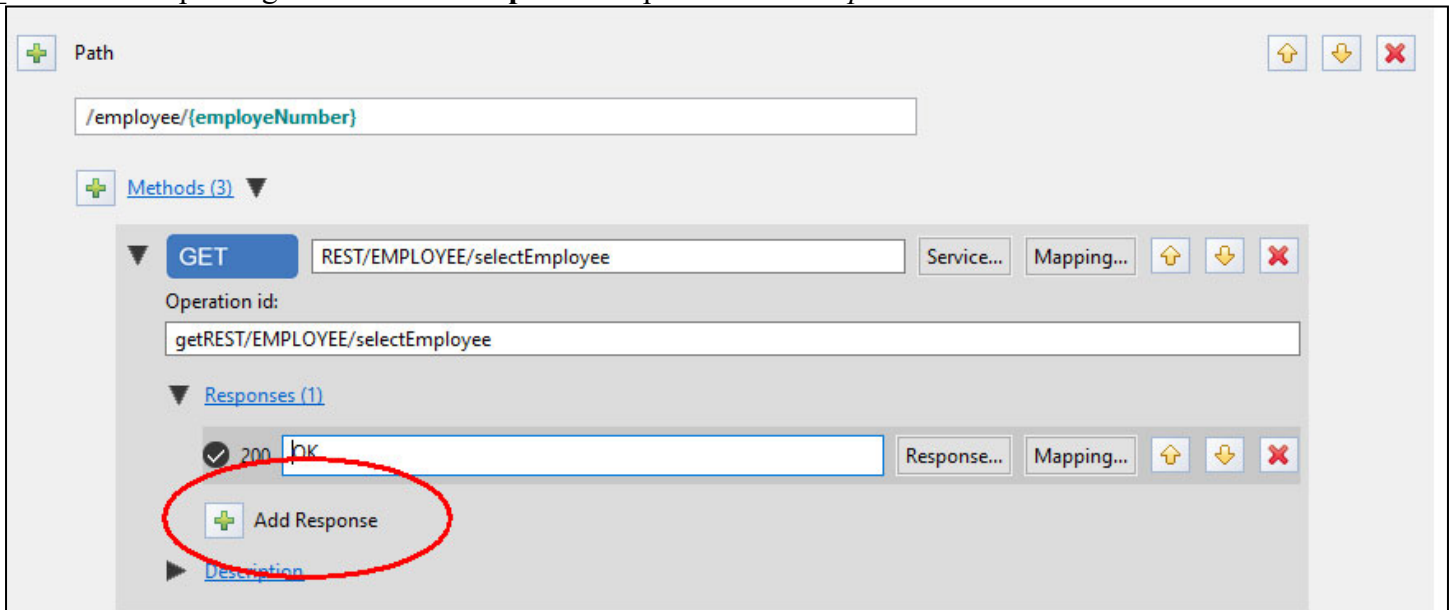


18. Repeat this mapping for the other two methods (**PUT** and **DELETE**).

19. Define a *Response Code* for a GET request that if the number of records returned equal zero that will set the HTTP response code to 404. Click on the **Mapping** button and select the *Define Response Codes* option.



20. Click the plus sign beside **Add Response** to open the *Add Response* window.



21. Use the pull-down arrow to select *404 – Not Found* for the *Response Code*. Use the pull-down arrows to select field *NumRecs* and the equal sign for *Rule 1*. Enter **0** in the open area for *Rule 1*. When finished your windows should look like the one below. Click **OK** to continue.

The screenshot shows the 'Add Response' dialog box. At the top, there's a title bar with a plus icon and the text 'Add Response'. Below it, there's a 'Response code' dropdown menu set to '404 - Not Found' and a 'Description' text box containing 'Not Found'. A horizontal line separates this from the 'Define rules' section. This section has a heading 'Define rules that indicate whether to use this response code and apply its response mapping, if defined.' Below the heading, there's a table-like structure for rules. The first rule, 'Rule 1', has a dropdown menu set to 'NumRecs', followed by an equals sign in a dropdown, and then a text box containing '0'. To the right of the text box are three small icons: a yellow arrow pointing up, a yellow arrow pointing down, and a red 'X'. Below the rule table is a green plus icon and the text 'Add Rule'. At the bottom, there's a 'Summary' section with a list box containing 'Rule 1'. In the bottom right corner, there are 'OK' and 'Cancel' buttons. A help icon (question mark in a circle) is in the bottom left corner.

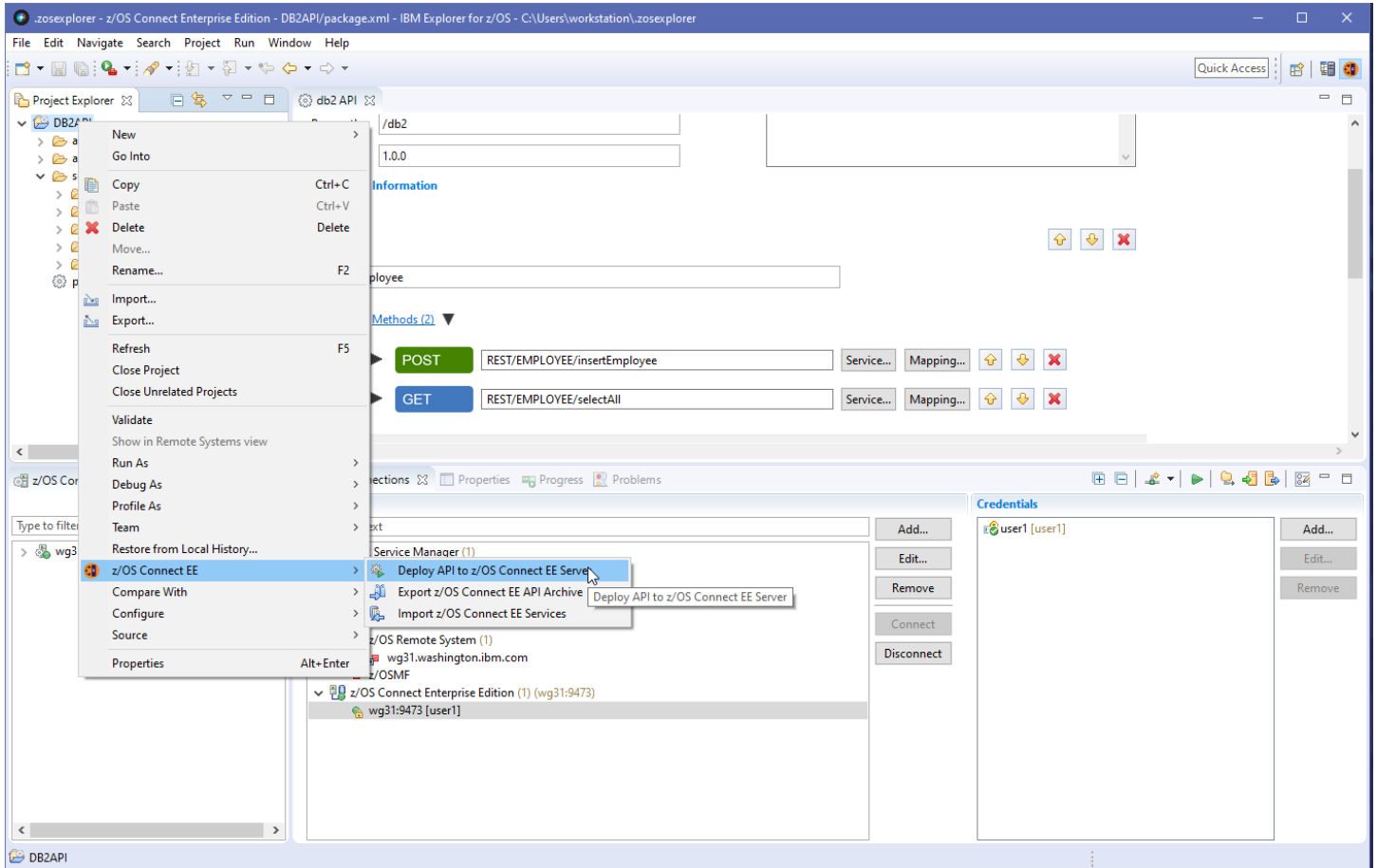
22. Add 404 response codes for the other two methods (**PUT** and **DELETE**).

Summary

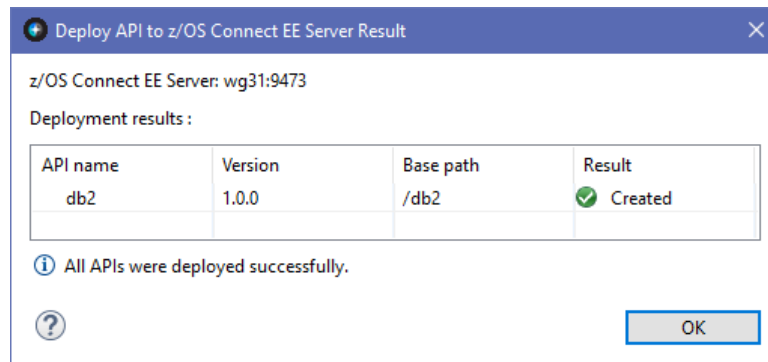
You created the API, which consists of multiple paths and the request and response mapping associated with each. That API will now be deployed into a z/OS Connect server.

Deploy the API to a z/OS Connect Server

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

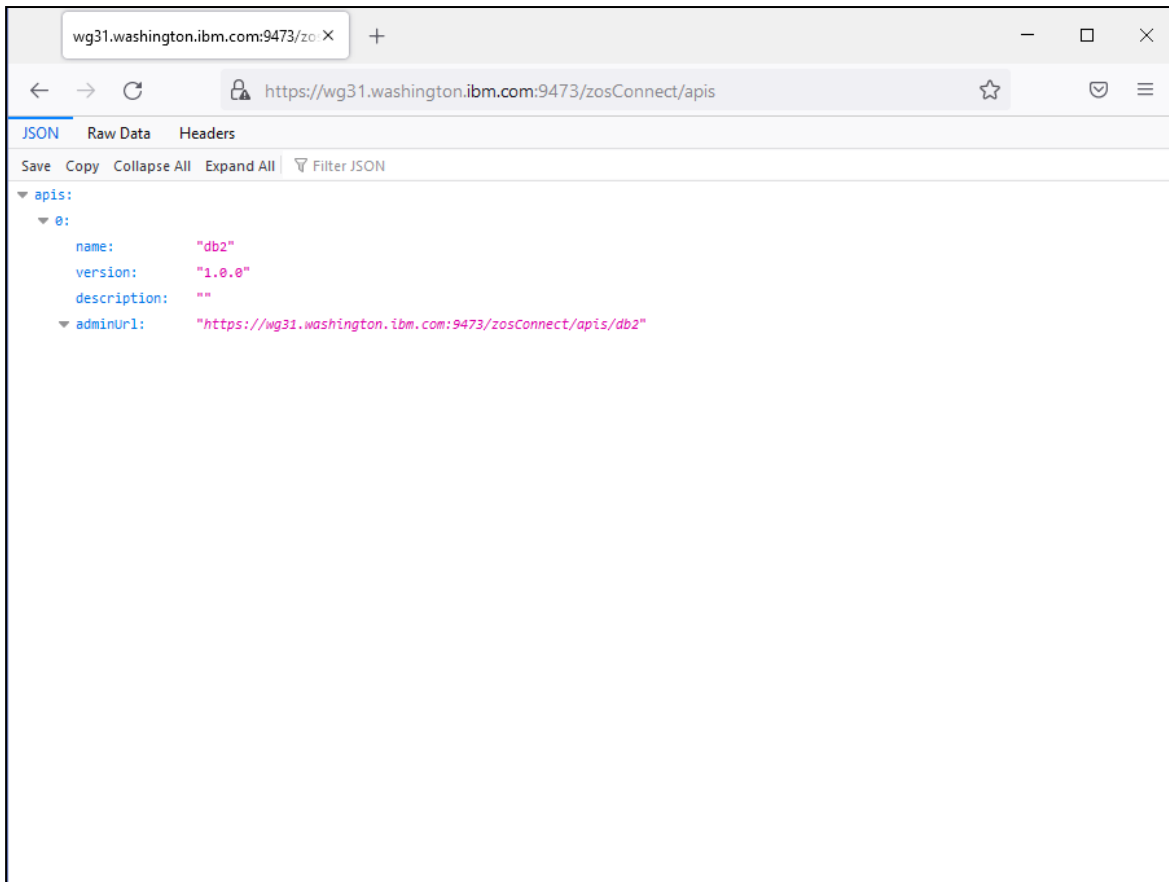


2. If the z/OS Explorer is connected to only one z/OS Connect server, there will be only one choice (*wg31:9473*). If z/OS Explorer had multiple connections to z/OS Connect servers then the pull-down arrow would allow a selection to which server to deploy, select *wg31:9473* from the list. Click **OK** 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 Db2 APIs using Swagger UI

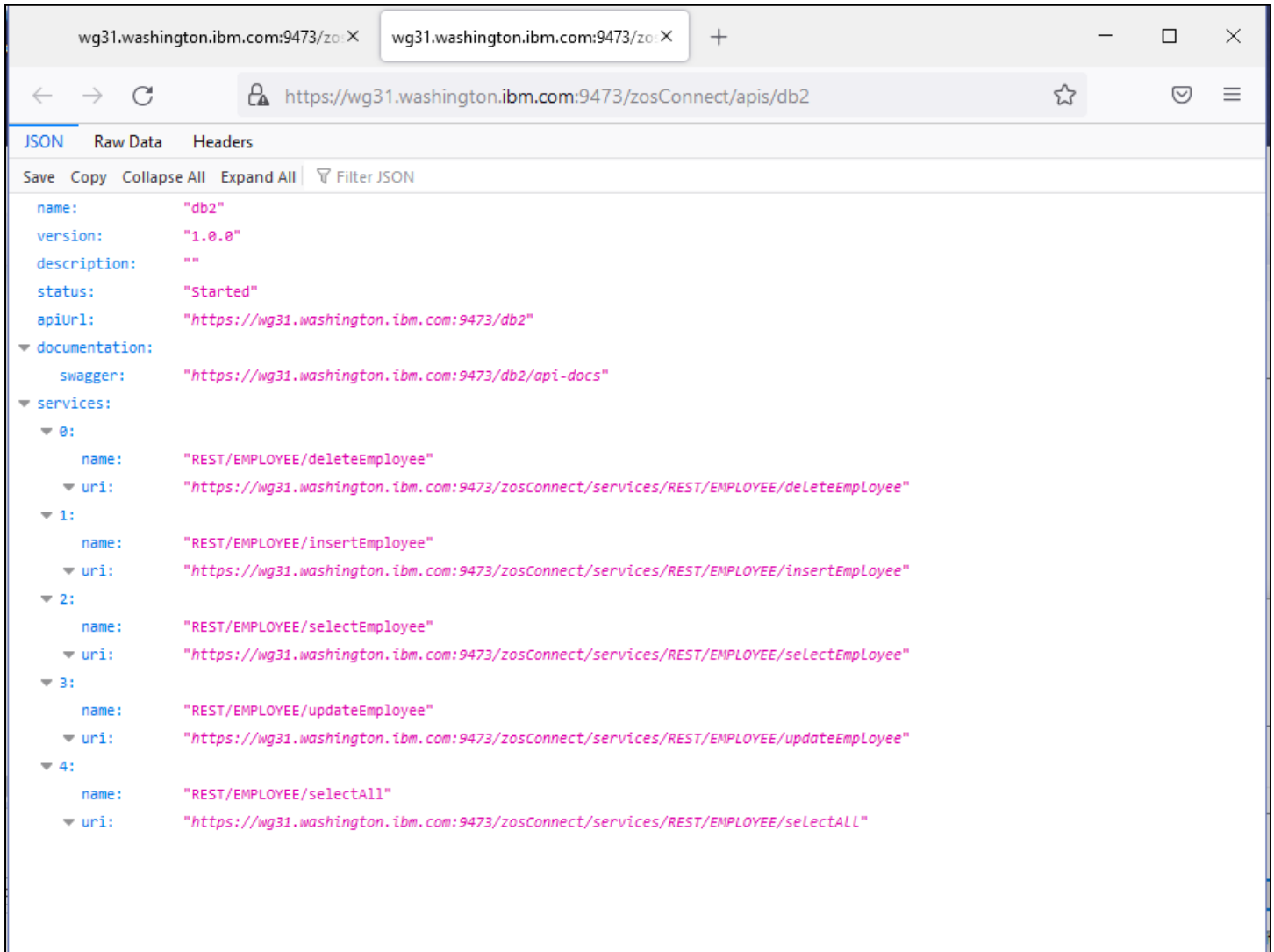
1. Next enter URL <https://wg31.washington.ibm.com:9473/zosConnect/apis> in the Firefox browser and you should see the window below.



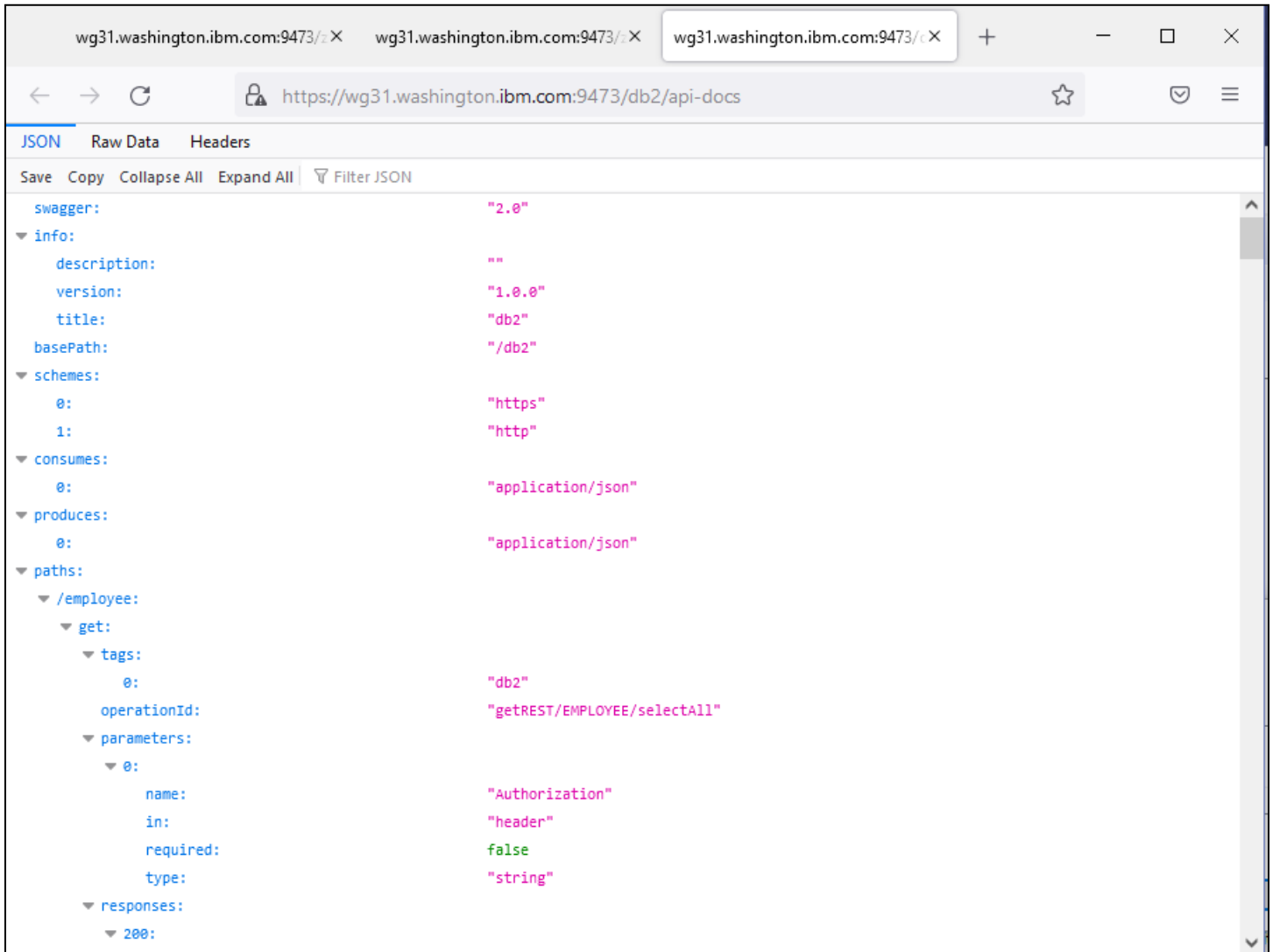
Tech Tip: You may be challenged by Firefox because the digital certificate used by the Liberty z/OS server is self-signed. Click the **Advanced** button to continue. Scroll down and then click on the **Accept the Risk and Continue** button. Next you may see a prompt for a userid and password. If you do see the prompt, enter the username **USER1** and password **user1** and click **OK**.

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.

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

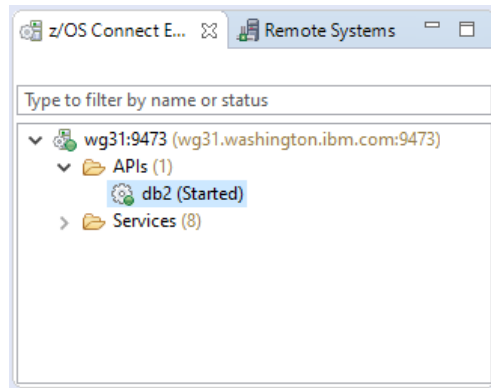


3. Finally click on the *swagger* URL for and you should see the Swagger document associated with this API.

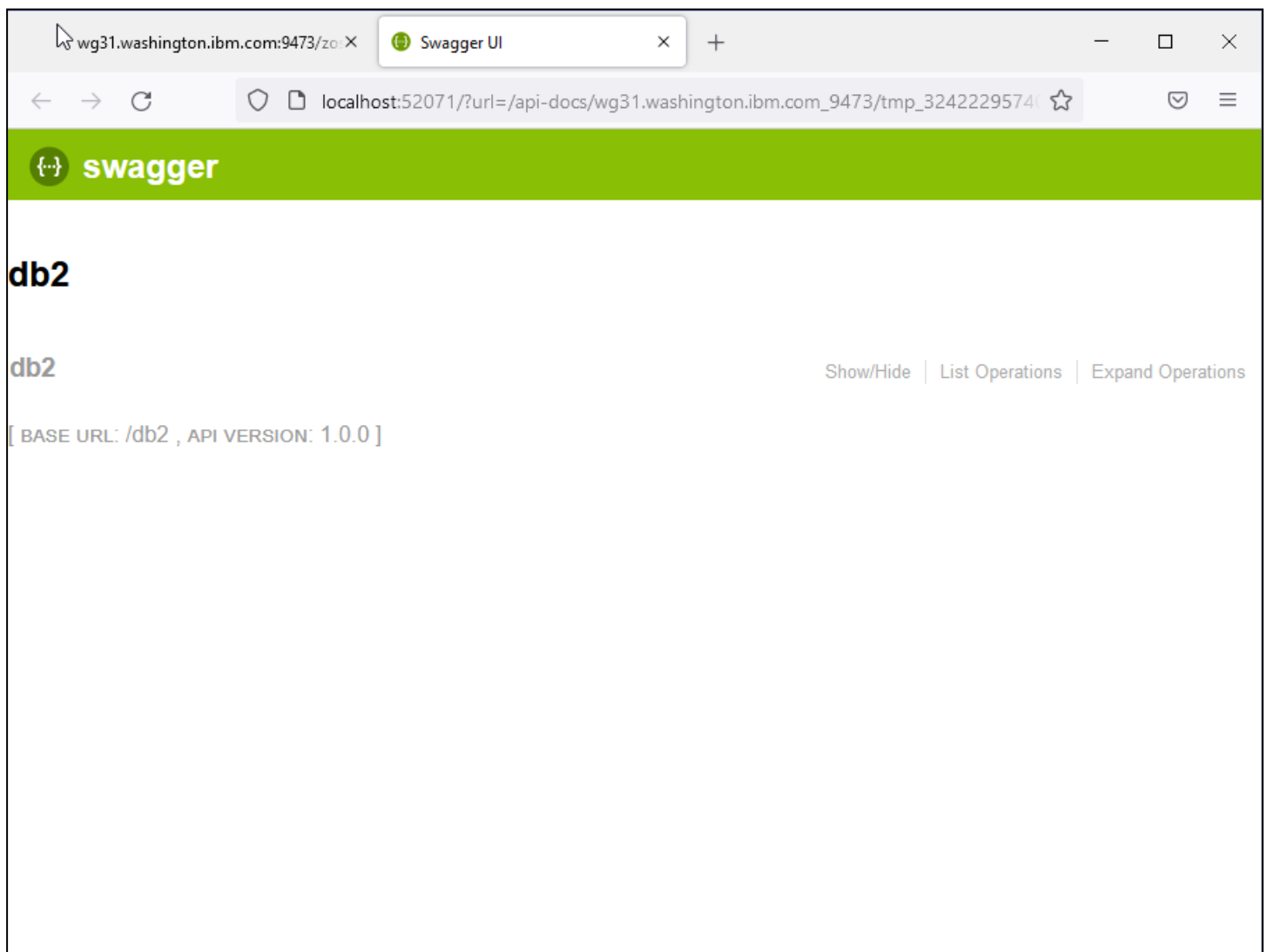


Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This Swagger document can be used by a developer or other tooling to develop REST clients for this specific API.

4. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect Servers*. Expand *wg31:9473* and then expand the *APIs* folder. You should see a list of the APIs installed in the server.



5. Right mouse button click on *db2* 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).



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

The screenshot shows a web browser window with the Swagger UI. The browser's address bar displays the URL: `localhost:52071/?url=/api-docs/wg31.washington.ibm.com_9473/tmp_3242229574/`. The Swagger UI header is green with the 'swagger' logo. The main content area is titled 'db2' and includes a 'List Operations' tab. Below the tab, a list of API endpoints is shown with their corresponding HTTP methods in colored boxes:

Method	Endpoint
GET	/employee
POST	/employee
DELETE	/employee/{employeeNumber}
GET	/employee/{employeeNumber}
PUT	/employee/{employeeNumber}

At the bottom of the interface, the text '[BASE URL: /db2 , API VERSION: 1.0.0]' is displayed.

7. Select the *GET* method for selecting all rows from the table by clicking on the */employee* URI string.

Remember this was the *Path* specified for the *GET* method for the *selectAll* service when the API was defined. This action will expand this method in this view and provide a Swagger UI test client (you may have to use the slider bar and adjust the perspective to see the entire client).

GET /employee

Response Class (Status 200)
OK

Model | Example Value

```
{
  "Result": 0,
  "Records": [
    {
      "BIRTHDATE": "string",
      "JOB": "string",
      "EMPNO": "string",
      "SALARY": 0,
      "PHONENO": "string",
      "COMM": 0,

```

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	<input type="text"/>		header	string

Try it out!

8. 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 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.

9. Scroll down the view and you should see the *Response Body* which contains the results of the GET method (see below).

Response Body

```

{
  "Affected": 0,
  "TotRecs": 43,
  "Skipped": 0,
  "NumRecs": 43,
  "NumFields": 14,
  "Records": [
    {
      "lastName": "HAAS",
      "comm": 4220,
      "birthdate": "1933-08-14",
      "bonus": 1000,
      "sex": "F",
      "workdept": "A00",
      "salary": 52750,
      "hiredate": "1965-01-01",
      "phoneno": "3978",
      "employeeNumber": "000010",
      "firstName": "CHRISTINE",
      "job": "DESS"
    }
  ]
}

```

Response Code

```

200

```

10. Select the *GET* method for selecting a single record from the Db2 table by clicking on the */employee/{employeeNumber}* URI string. Remember this was the *Path* specified for the *GET* method for the *selectEmployee* service when the API was defined. This action will expand this method in this view and provide a Swagger UI test client (you may have to use the slider bar and adjust the perspective to see the entire client).

11. Enter **Basic VVNFUjE6VVNFUjE=** in the box beside *Authorization* and **0000`0** in the area beside *employeeNumber* and press the **Try it out!** button.

Response Class (Status 200)

OK

Model | Example Value

```
{
  "Result": 0,
  "Records": [
    {
      "BIRTHDATE": "string",
      "JOB": "string",
      "EMPNO": "string",
      "SALARY": 0,
      "PHONENO": "string",
      "COMM": 0,

```

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	Basic VVNFUjE6VVNFUjE=		header	string
employeeNumber	000010		path	string

Try it out!

12. Scroll down the view and you should see the *Response Body* which contains the results of the GET method (see below).



The screenshot displays the API response interface. The 'Response Body' section shows a JSON object with the following structure:

```
{
  "Affected": 0,
  "TotRecs": 1,
  "Skipped": 0,
  "NumRecs": 1,
  "NumFields": 14,
  "Records": [
    {
      "lastName": "HAAS",
      "hireDate": "1965-01-01",
      "birthdate": "1933-08-14",
      "bonus": 1000,
      "sex": "F",
      "salary": 52750,
      "employeeNumber": "000010",
      "firstName": "CHRISTINE",
      "phone": "3978",
      "educationLevel": 18,
      "commission": 4220,
      "lastName": "HAAS"
    }
  ]
}
```

The 'Response Code' section shows the status code 200.

13. Select the *POST* method for inserting a new employee in to the Db2 table by clicking on the *POST /employee* URI string. Remember this was the *Path* specified for the *POST* method for the *insertEmployee* service when the API was defined. Enter **Basic VVNFUjE6VVNFUjE=** in the box beside *Authorization*

Enter the following values for the other fields

- a value of **1000** in the area under *COMMISSION*
- a value of **1900-10-10** in the area under *BIRTHDATE*
- a value of **T** in the area under *MIDDLEINIT*
- a value of **25** in the area under *EDUCATIONLEVEL*
- a value of **Analyst** in the area under *JOB*
- a value of **948478** in the area under *EMPLOYEE NUMBER*
- a value of **Matt** in the area under *FIRSTNAME*
- a value of **0065** in the area under *PHONENUMBER*
- a value of **10000** in the area under *SALARY*
- a value of **1970-01-01** in the area under *HIREDATE*
- a value of **M** in the area under *SEX*
- a value of **4000** in the area under *BONUS*
- a value of **A00** in the area under *DEPARTMENT*
- a value of **Johnson** in the area under *LASTNAME*

Then press the **Try it out!** button.

type: VARCHAR}

HIREDATE

1980-10-30

{name: HIREDATE, columnType: 1, dataFormat: CHARACTER, type: VARCHAR}

SEX

M

{name: SEX, columnType: 1, dataFormat: CHARACTER, type: VARCHAR}

BONUS

4000

{name: BONUS, columnType: 1, dataFormat: CHARACTER, type: VARCHAR}

DEPARTMENT

A00

{name: DEPARTMENT, columnType: 1, dataFormat: CHARACTER, type: VARCHAR}

LASTNAME

Johnson

{name: LASTNAME, columnType: 1, dataFormat: CHARACTER, type: VARCHAR}

Parameter content type: application/json

Try it out!

Hide Response

14. The results should show a **Response Code** of 200. Note that the columns removed from the interface in an earlier step are not present.

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: Basic < >
```

Request URL

```
https://wg31.washington.ibm.com:9473/db2/employee
```

Request Headers

```
{  "Accept": "application/json",  "Authorization": "Basic VVNFUjE6VVNFUjE="}
```

Response Body

```
{  "Records": []}
```

Response Code

```
200
```

15. Select the *PUT* method for updating a row in the Db2 table by clicking on the *PUT* */employee/{employeeNumber}* URI string. Remember this was the *Path* specified for the *PUT* method for the *updateEmployee* when the API was defined. Enter **Basic VVNFUjE6VVNFUjE=** in the box beside *Authorization*, a value of **948478** in the area under *employeeNumber* value of **75000** in the area under *Bonus*. Then press the **Try it out!** button.

Parameters				
Parameter	Value	Description	Parameter Type	Data Type
Authorization	<input "="" type="text" value="Basic VVNFUjE6VVNFUjE="/>		header	string
employeeNumber	<input type="text" value="948478"/>		path	string
putREST/EMPLOYEE /updateEmployee_request	<div> <input type="text" value="-"/> </div> <div> <div>BONUS</div> <input type="text" value="75000"/> <small>{name: BONUS, columnType: 1, dataFormat: CHARACTER, type: VARCHAR}</small> </div> <div>Parameter content type: <input type="text" value="application/json"/></div>	request body	body	<div>Model</div> <div>Example Value</div> <pre>{ "BONUS": "string" }</pre>

16. The results should show a **Response Code** of 200.

Response Body

```
{
  "Affected": 1,
  "TotRecs": 0,
  "Skipped": 0,
  "NumRecs": 0,
  "NumFields": 0,
  "Records": [],
  "Result": 0
}
```

Response Code

200

17. Use the GET method for retrieving a single row from the Db2 table to retrieve the row for *employeeNumber* 948478 to confirm the *Bonus* column has been updated.

Response Body

```
records : [
  {
    "lastName": "Johnson",
    "hireDate": "1980-10-30",
    "birthdate": "1900-10-10",
    "bonus": 75000,
    "sex": "M",
    "salary": 10000,
    "employeeNumber": "948478",
    "firstName": "Matt",
    "phone": "0065",
    "educationLevel": 25,
    "commission": 1000,
    "department": "A00",
    "job": "analyst",
    "middleInit": "T"
  }
],
"Result": 0
}
```

Response Code

200

18. Select the *DELETE* method for deleting a row from the Db2 table by clicking on the *DELETE* */employee/{employeeNumber}* URI string. Remember this was the *Path* specified for the *PUT* method for the *deleteEmployee* service when the API was defined. Enter **Basic VVNFUjE6VVNFUjE=** in the box beside *Authorization*, a value of **948478** in the area beside **employeeNumber**. Then press the **Try it out!** button. It should complete with a 200 *Response Code*.
19. Press the **Try it out!** button again for the delete request. This time it should fail with a 404 (not found) *Response code*. This occurred because of the check for the number of records response test added to the API.

Response Body

```
{
  "Affected": 0,
  "TotRecs": 0,
  "Skipped": 0,
  "NumRecs": 0,
  "NumFields": 0,
  "Records": [],
  "Result": 0
}
```

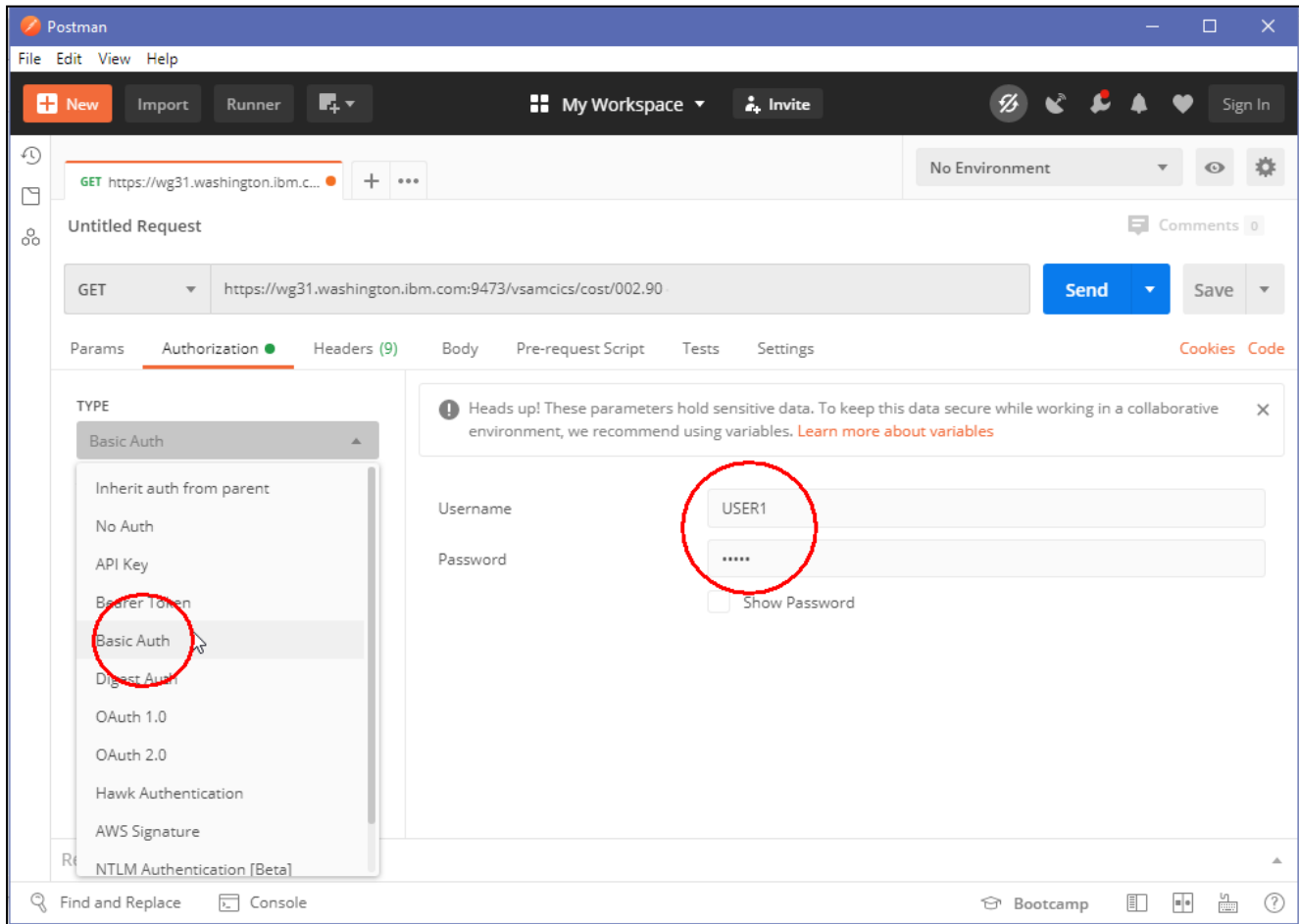
Response Code

404

Test the DB2 APIs using Postman

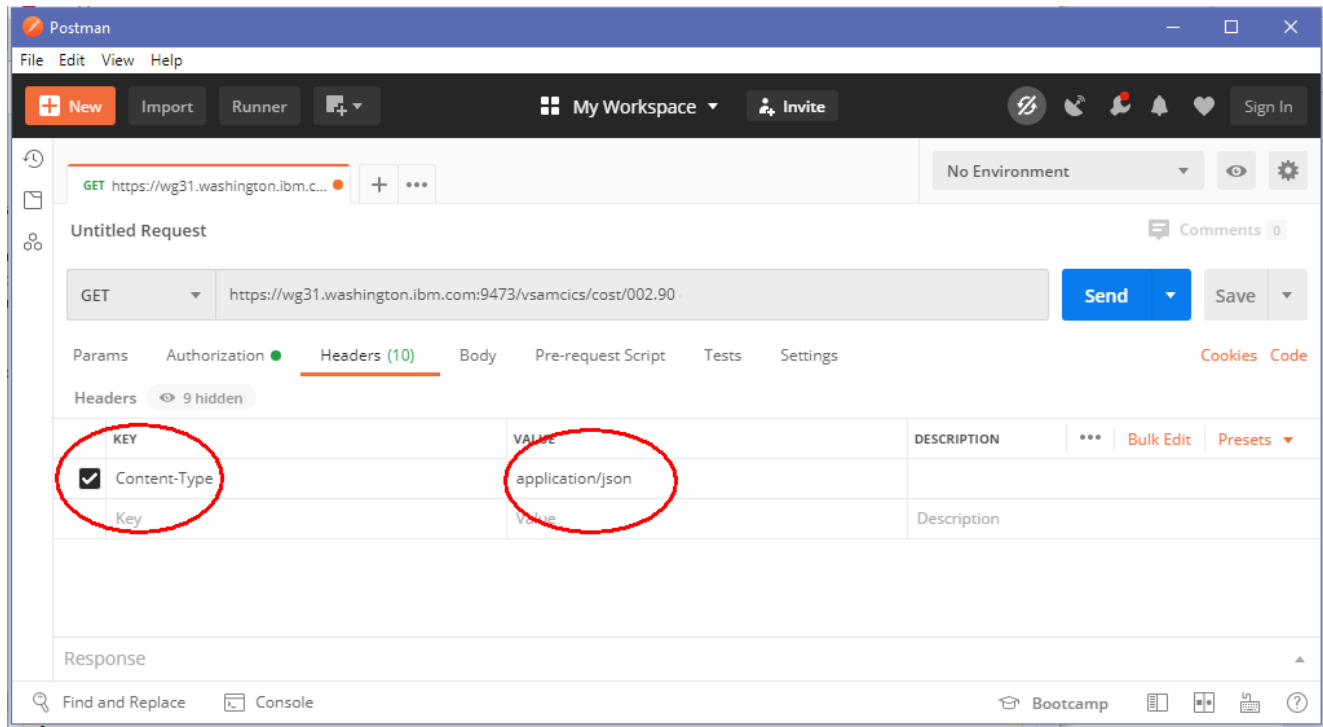
The other API services will be tested using Postman.

1. Open the *Postman* tool icon on the desktop. If necessary reply to any prompts and close any welcome messages.
2. Next, select the *Authorization* tab to enter an authorization identity and password. Use the pull down arrow to select *Basic Auth* and enter *USER1* as the *Username* and *USER1* as the *Password*.



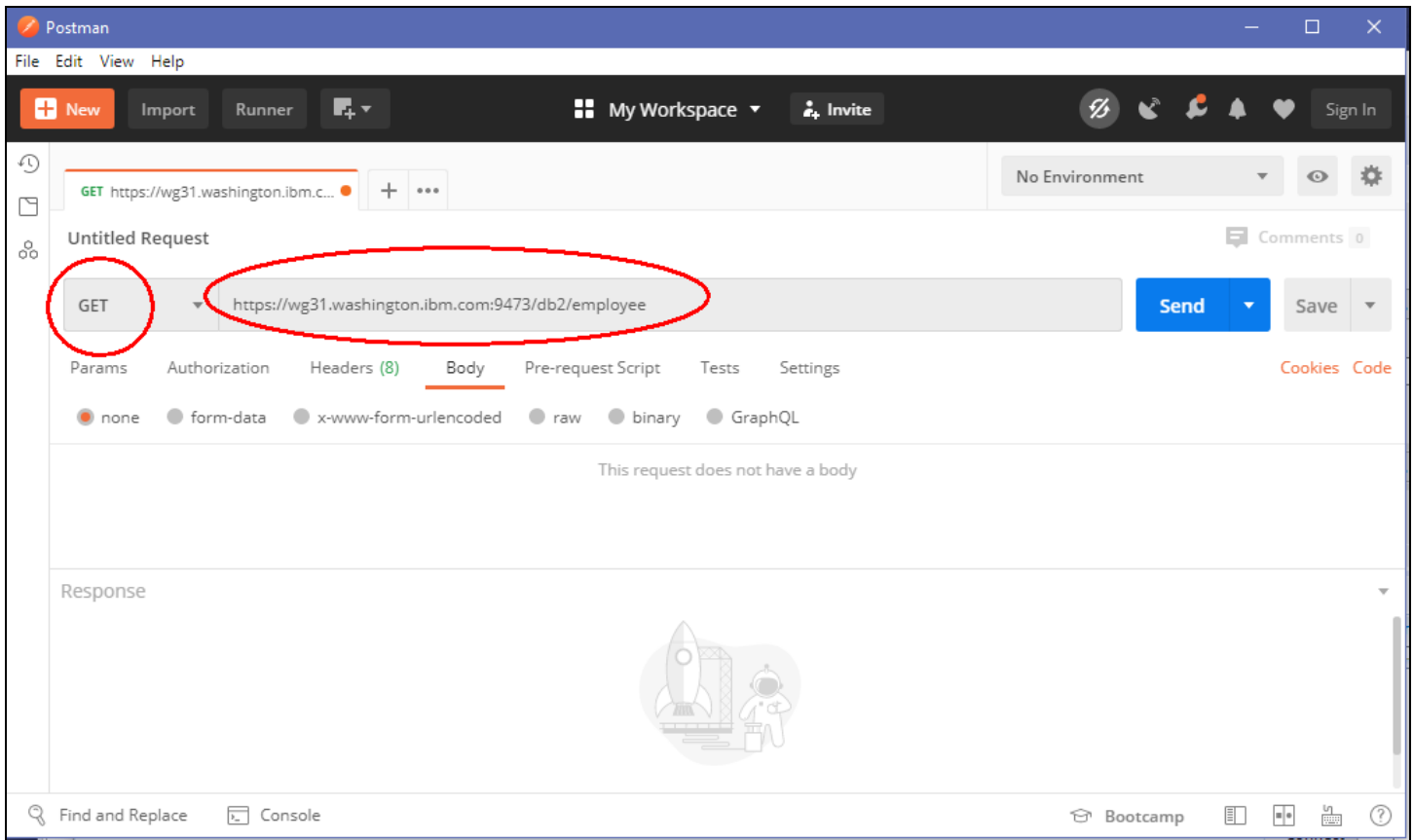
Tech-Tip: If the above Postman view is not displayed select *File* on the toolbar and then choose *New Tab* on the pull down. Alternatively, if the *Launchpad* view is displayed, click on the *Create a request* option.

3. Next, select the *Headers* tab. Under *KEY* use the code assist feature to enter ***Content-Type***, and under *VALUE*, use the code assist feature to enter ***application/json***.

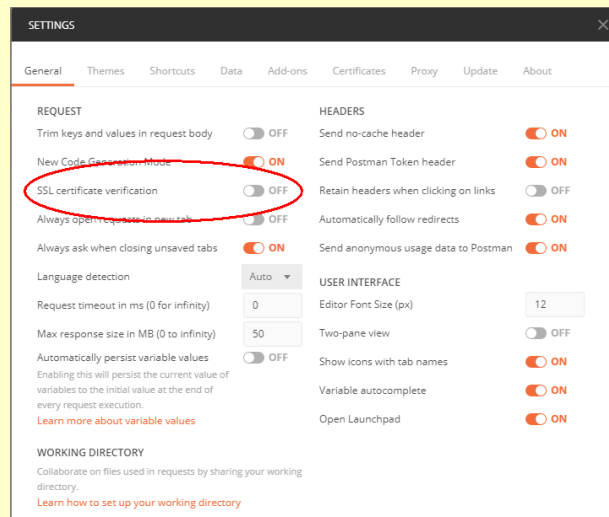


Tech-Tip: Code assist simply means that when text is entered in field, all the valid values for that field that match the typed text will be displayed. You can select the desired value for the field from the list displayed and that value will populate that field.

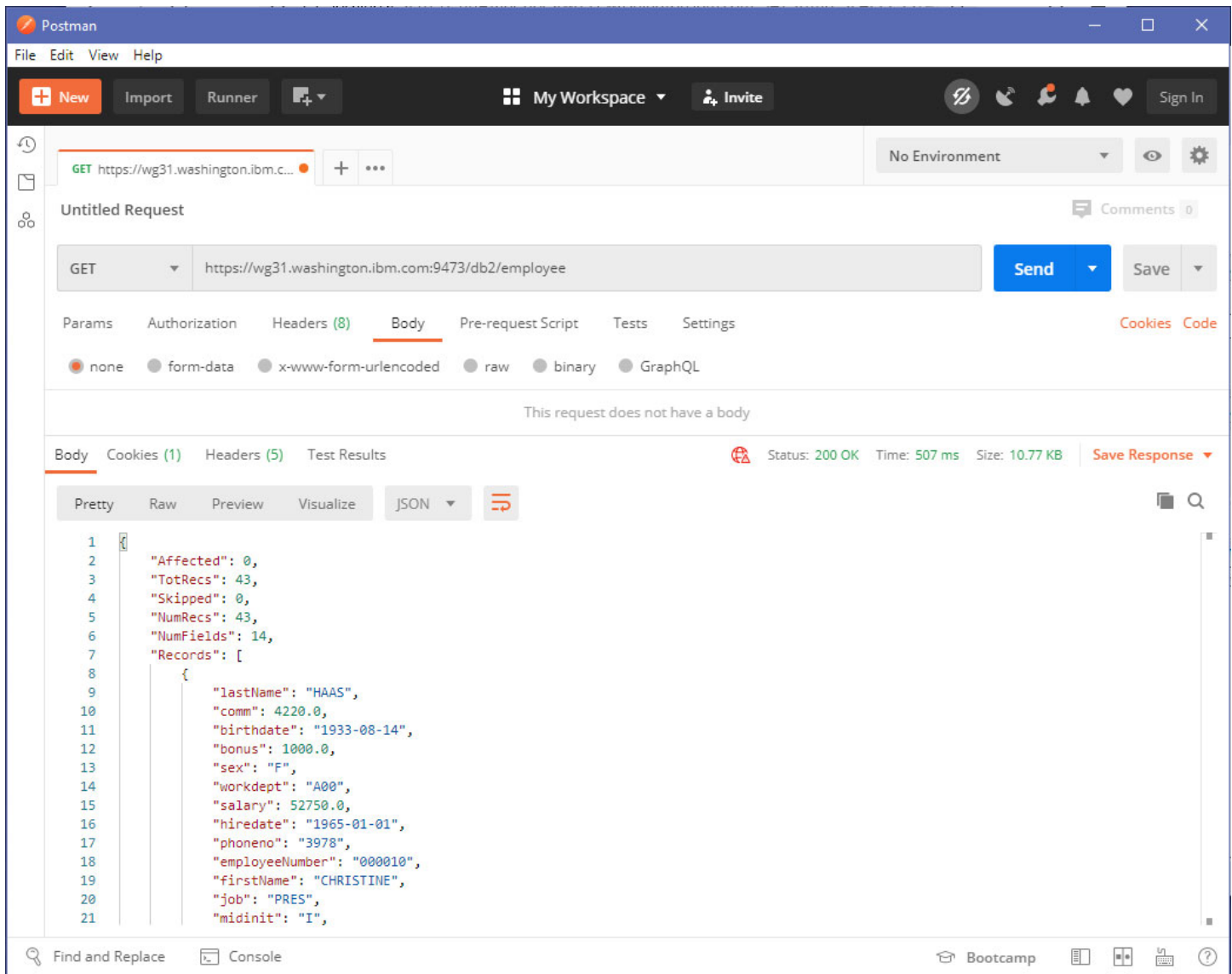
4. To test the *selectByCost* service, use the down arrow to select **GET** and enter <https://wg31.washington.ibm.com:9473/db2/employee> in the URL area (see below) to select all stock status segments for this root segemen.



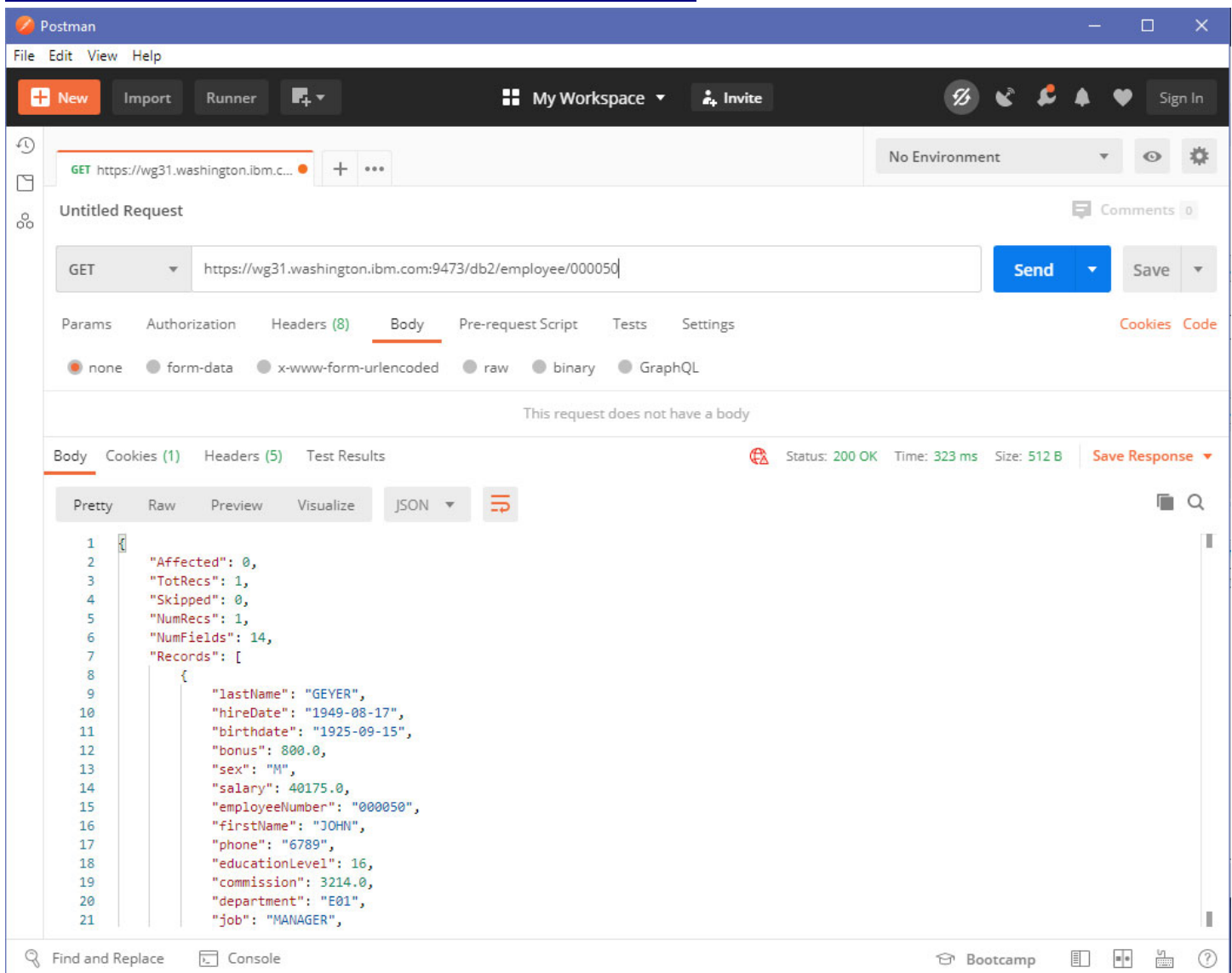
Tech-Tip: The Postman settings have been changed to disable *SSL certificate verification*, a settings option.



5. Next select the *Body* tab and select the *raw* radio button. Then press the **Send** button. A response message should come back indicating the service has been started and other details about the service. You may have to ‘drag’ the response body area up to display the response message.



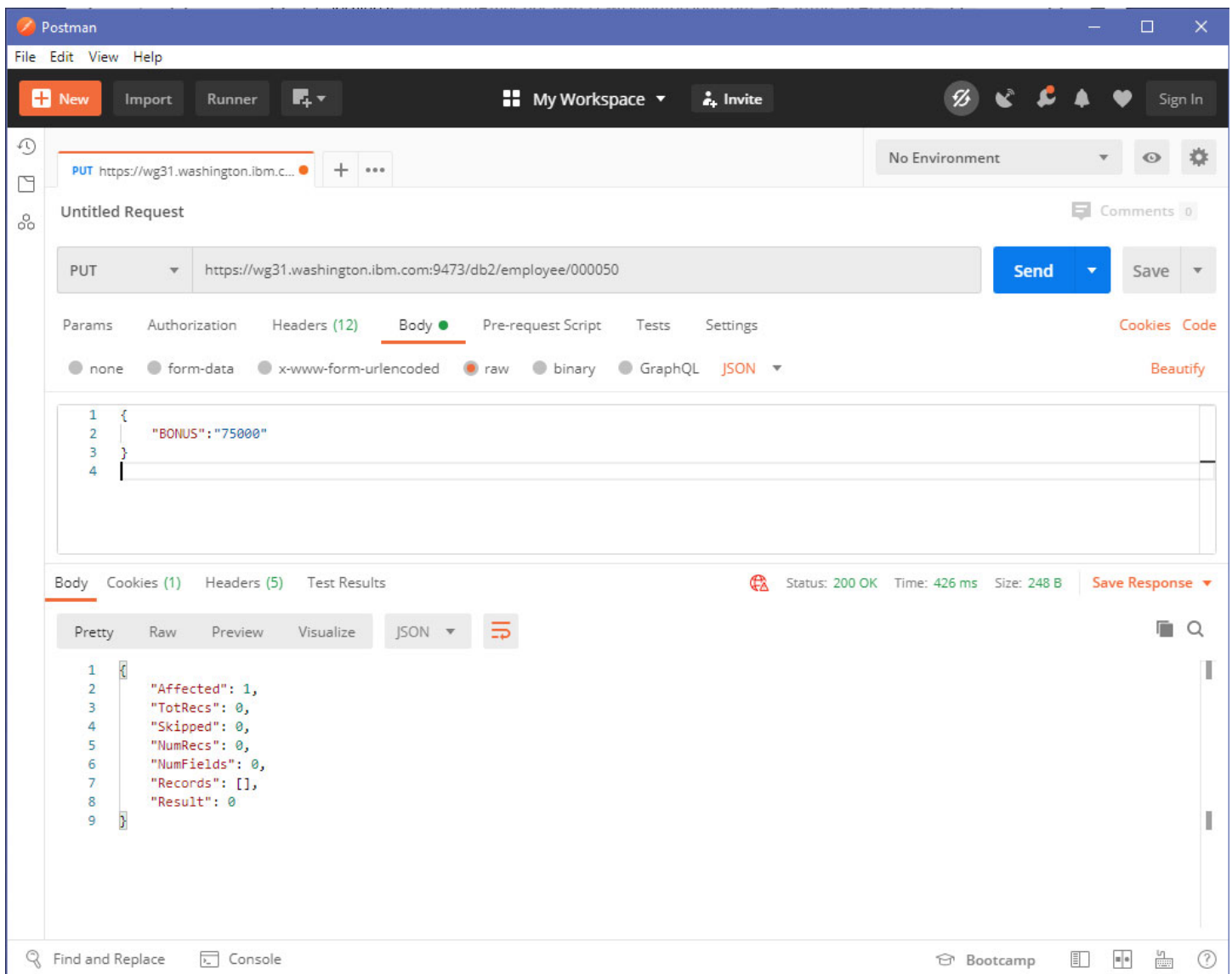
6. To test the *selectEmployee* API, use a **GET** method with URL
<https://wg31.washington.ibm.com:9473/db2/employee/000050>



7. To test the *updateEmployee*, use the down arrow to select **PUT** and enter **<https://wg31.washington.ibm.com:9473/employee/00050>** in the URL area (see below). Enter the JSON request message below.

```
{
  "BONUS": "75000"
}
```

Press the **Send** button to update this row



Tech-Tip: Use a GET to URL `https://wg31.washington.ibm.com:9473/db2/employee/000050` to display the item and confirm the update has taken place.

Summary

You use DVM to develop 5 DVM services. The SAR files for the DVM services were imported in the API Editor of z/OS Connect. The API Editor was used to develop a RESTful API. Then you have verified the API. The API layer provided a further level of abstraction and allows a more RESTful use of HTTP verbs, better mapping of data via the API editor function and a more variety of client security options.