IBM z/OS Connect EE V3.0

Developing RESTful APIs for CICS

WSC Wildfire Team
IBM z Systems

Lab Version Date: March 23, 2019

Table of Contents

Important: On the desktop there is a file named *Developing APIs CopyPaste.txt*. This file contains commands and other text used in this workshop. Locate that file and open it. Use the copy-and-paste function (**Ctrl-C** and **Ctrl-V**) to enter commands or text. It will save time and help avoid typo errors. As a reminder text that appears in this file will be highlighted in yellow.

Mitch Johnson (mitchj@us.ibm.com)

Overview

Important – You do not need any skills with CICS to perform this exercise. Even if CICS is not relevant to your current plans performing this exercise will give additional experience using the Toolkit to develop services and APIs.

The objective of these exercises is to gain experience with working with z/OS Connect EE and the 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.

If you have completed either the developing APIs exercise for Db2 or MQ and Batch you can start with section *z/OS Connect EE APIs and a CICS Containers program* on page 7.

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 IBM z/OS Explorer, 3270 screens, features or tools do not hesitate to ask the instructor for assistance.
- ✓ The exercises for CICS CONTAINER and COMMAREA are independent of each other and can be completed in any sequence
- ✓ 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.

Mitch Johnson (mitchj@us.ibm.com)

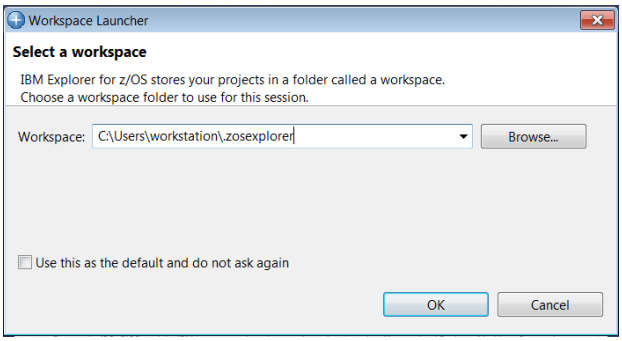
Connect the IBM z/OS Explorer to the z/OS Connect EE Server

Begin by establishing a connection to your z/OS Connect server from IBM z/OS Explorer. If you have performed one of the other exercises in this series of exercises this step may not be required.

> **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.

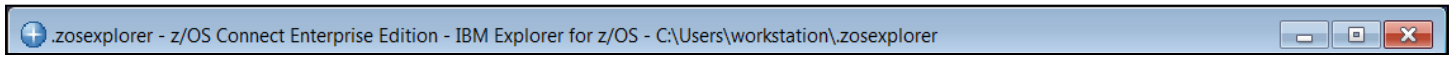
___1. On the workstation desktop, locate the *z/OS Explorer* icon and double click on it to open the Explorer.

___2. You will be prompted for a workspace:



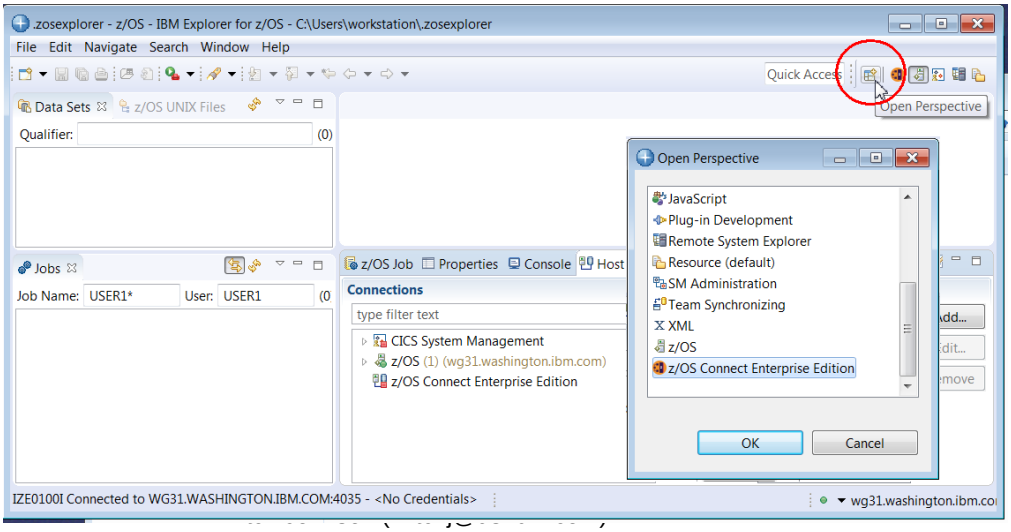
Take the default value by clicking **OK**.

___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:

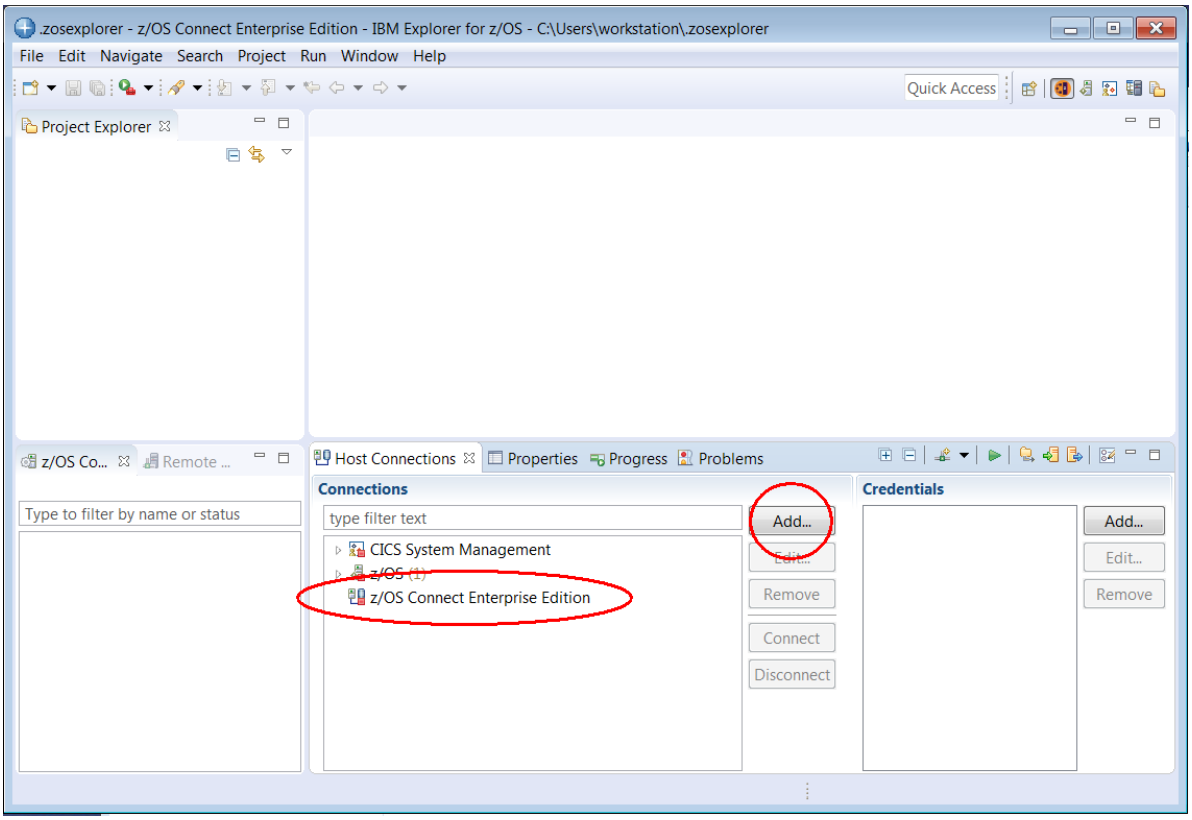


N.B. If a *Welcome* screen is displayed then click the white X beside *Welcome* to close this view.

___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 connection*s 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.

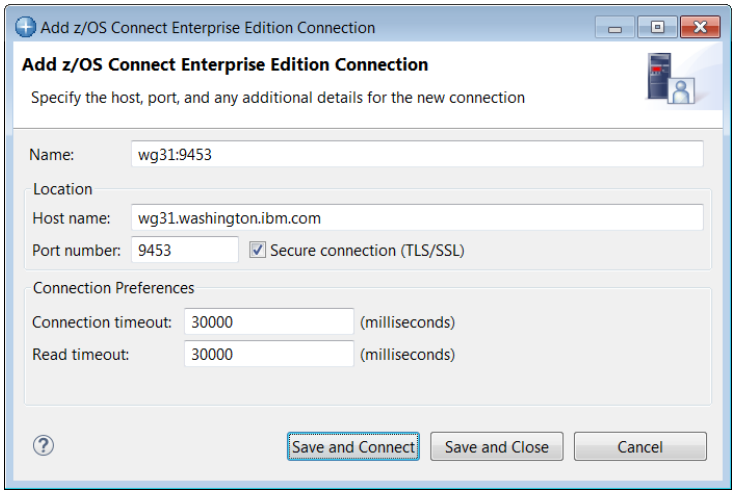
A view is an area in the window dedicated to providing a specific tool or function. For example, in the window above, *Host Connections* and *Project Explorer* are views that use different areas of the window for displaying information. At bottom on the right there is a single area for displaying the 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.

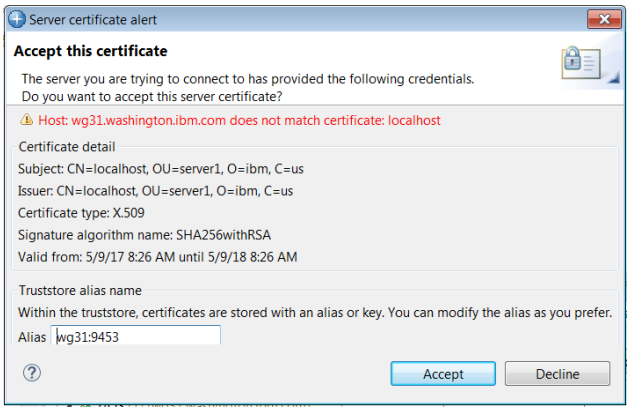
Mitch Johnson (mitchj@us.ibm.com)

___6. In the pop-up list displayed select *z/OS Connect Enterprise Edition* and on the *Add z/OS Connect Enterprise Edition Connection* screen enter ***wg31.washington.ibm.com*** for the *Host name*, ***9453*** 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 ***Fred*** and a *Password or Passphrase* of ***fredpwd*** (case matters). Remember the server is configured to use basic security. If SAF security had been enabled, then a valid RACF User ID and password will have to be used instead. 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 ***Fred*** and ***fredpwd*** again.



___9. The status icon beside *wg31:9453* 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.
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

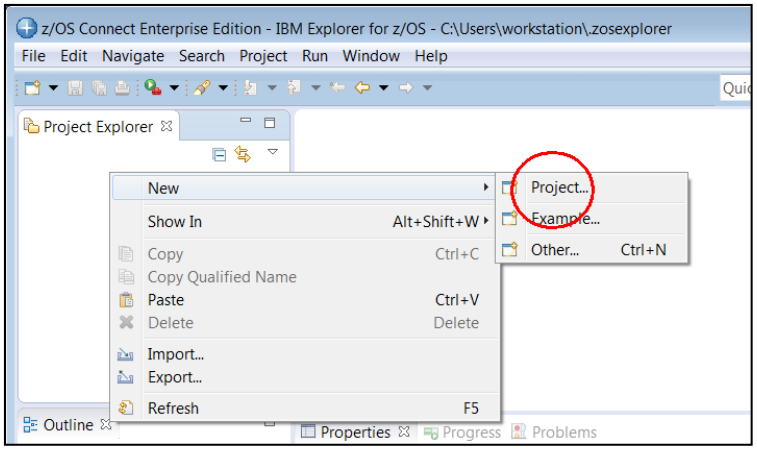
z/OS Connect EE APIs and a CICS Containers program

This section provides an opportunity to compose and deploy an API that invokes a CICS program using CICS Channels and Containers. The target CICS program, *CSCVINC* is a simple CICS program that based on a field in a container passed to the program accesses a VSAM file. The request could be insert a new record, replace an existing record, delete or record or retrieve an existing record.

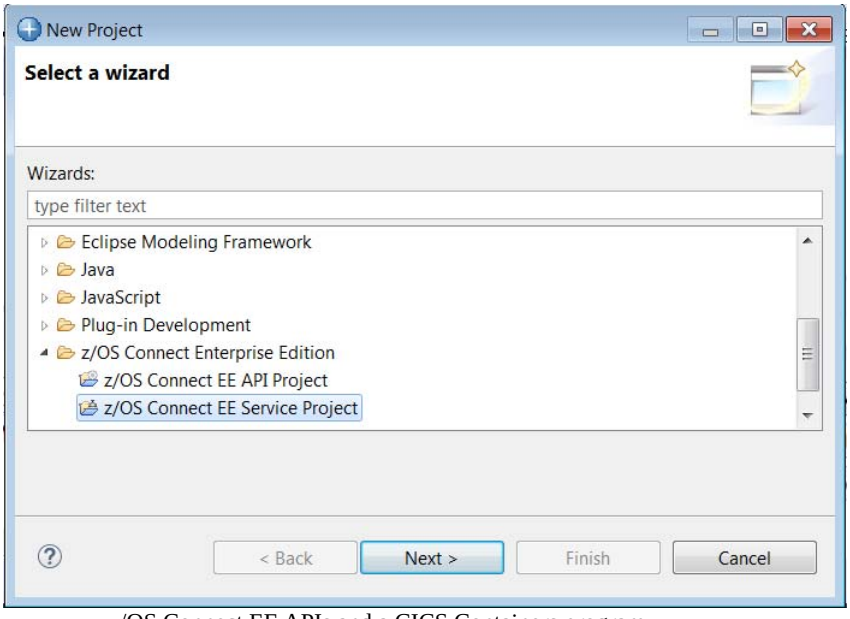
Create the service

The first step is to create the service which describes the interaction with the CICS program CSCVINC. Switch to the *z/OS Connect Enterprise Edition* perspective.

___1. In the upper left, position your mouse anywhere in the *Project Explorer* view and right-mouse click, then select *New →Project*:

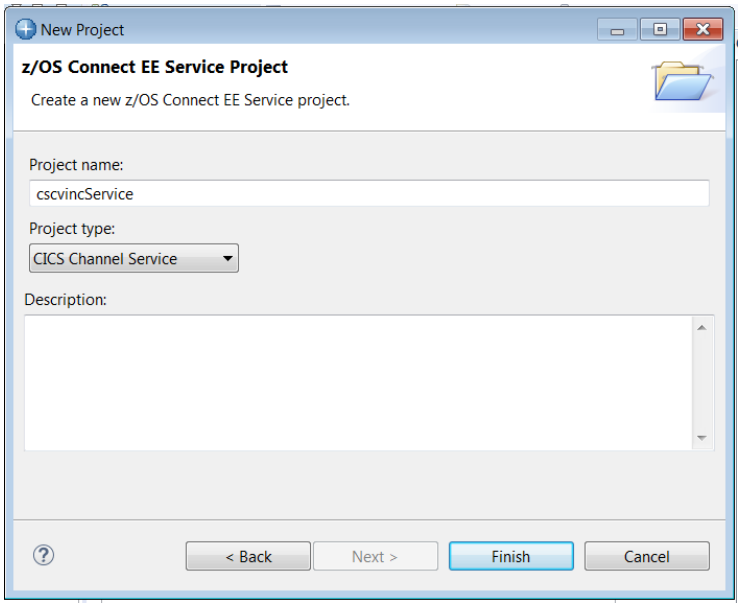


___2. In the *New Project* window, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE Service Project* and then click the **Next** button.

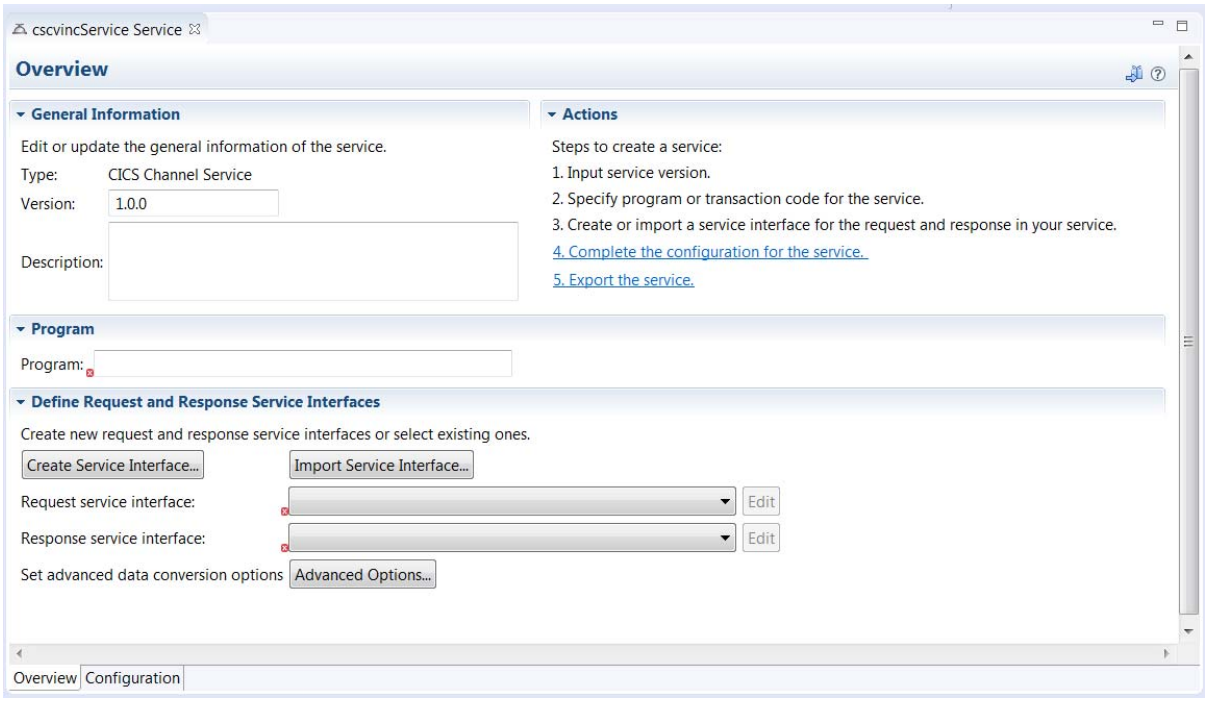


Mitch Johnson (mitchj@us.ibm.com)

___3. On the new *New Project* window enter ***cscvincService*** the *Project name* and use the pull-down arrow to select *CICS Channel Service* as the *Project type*. Click **Finish** to continue



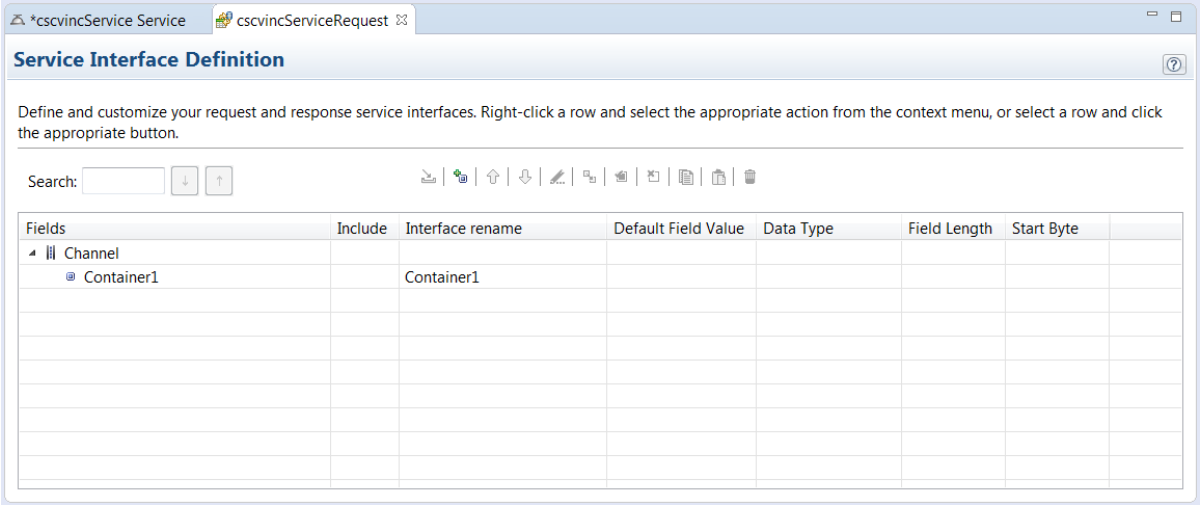
___4. This will open the *Overview* window for the *cscvincService*. For now, disregard the message about the 3 errors detected, they will be addressed shortly.



___5. Next enter the target CICS program name ***CSCVINC*** in the area beside *Program*.

___6. Click the **Create Service Interface** button to create the first service interface required by this API and enter a *Service interface name* of ***cscvincServiceRequest***. Click **OK** to continue.

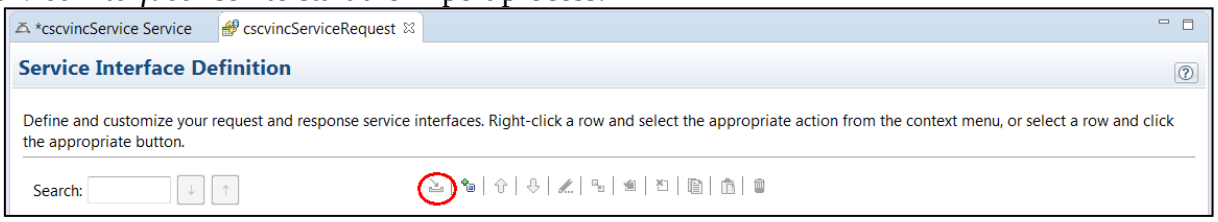
___7. This will open a *Service Interface Definition* window.



___8. The first step is to import the COBOL copy book that represents the inbound or request CONTAINER.

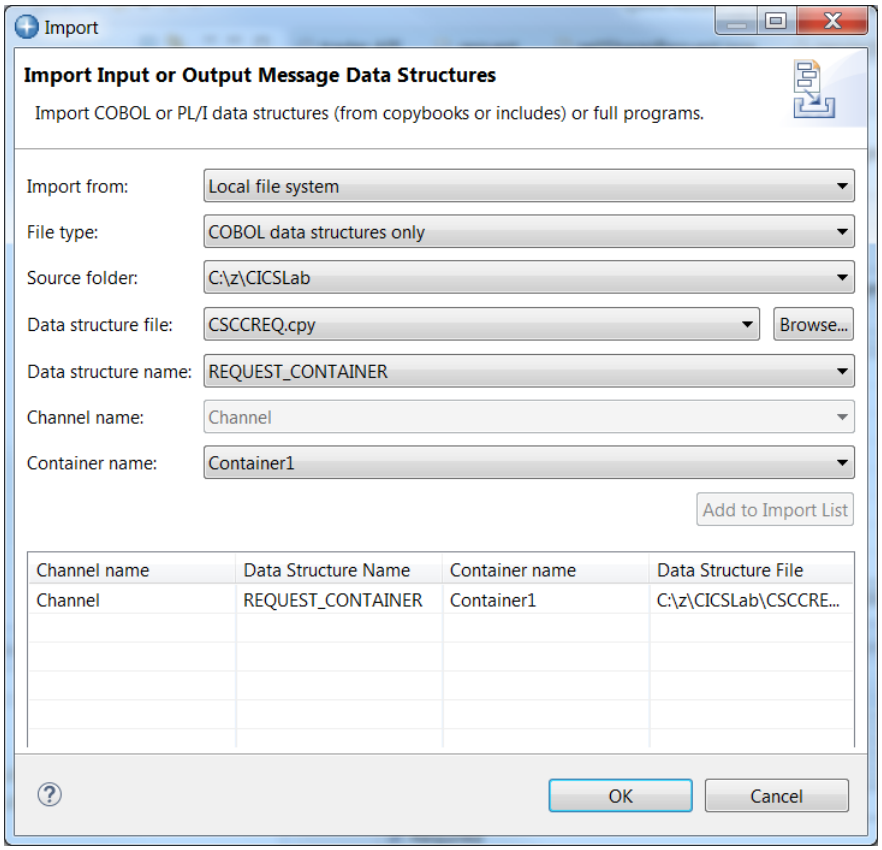
```
       01 Request-Container.
           03  ACTION           PIC X(1).
           03  USERID           PIC X(8).
           03  FILEA-AREA.
               05  STAT         PIC X.
               05  NUMB         PIC X(6).
               05  NAME         PIC X(20).
               05  ADDRX        PIC X(20).
               05  PHONE        PIC X(8).
               05  DATEX        PIC X(8).
               05  AMOUNT       PIC X(8).
               05  COMMENT      PIC X(9).
```

___9. On the *Service Interface Definition* window, there is a tool bar near the top. If you hover over an icon its function will be display as below. Select *Container1* a click the *Import COBOL or PL/I data structure into the service interface* icon to start the import process.

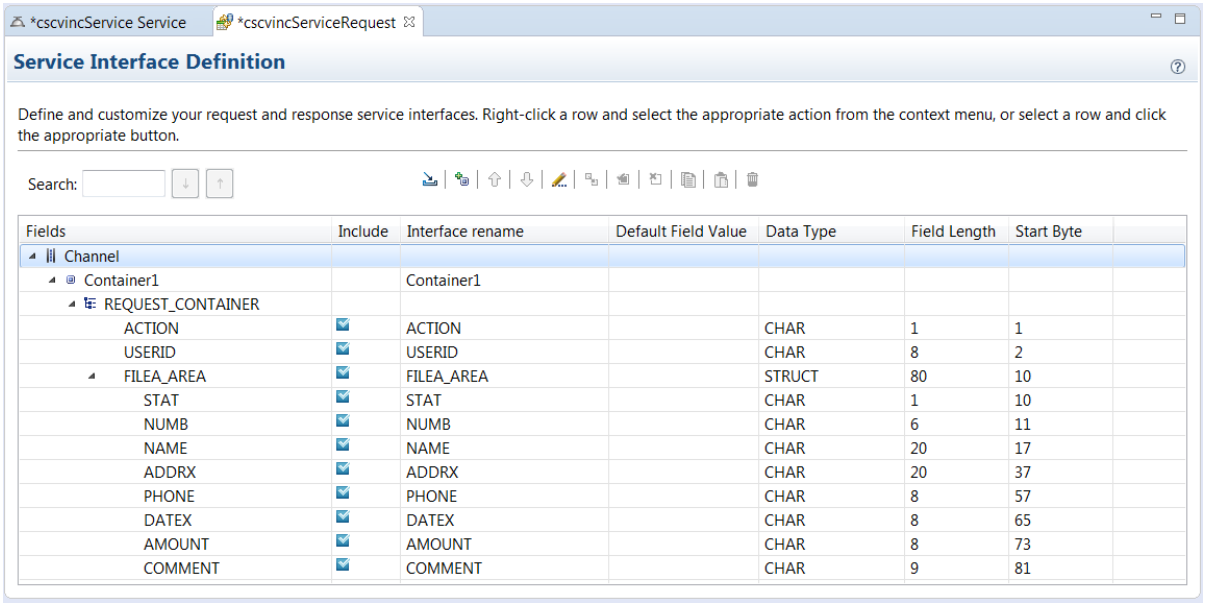


Mitch Johnson (mitchj@us.ibm.com)

___10. This will open the *Import* window. On this window select *Local file system* as source of the import and *COBOL data structure only* as the *File type*. Press the **Browse** button and **Open** directory *C:\z\CICSLAB* and then select file *CSCCREQ*.cpy and click **Open** to import this file into this project. Click the **Add to Import List** button to continue.



___11. Click **OK** and when you expand *Container1* you will see the COBOL 'variables' that have been imported into the service project.



Mitch Johnson (mitchj@us.ibm.com)

N.B. the interface names were derived from the COBOL source shown earlier.

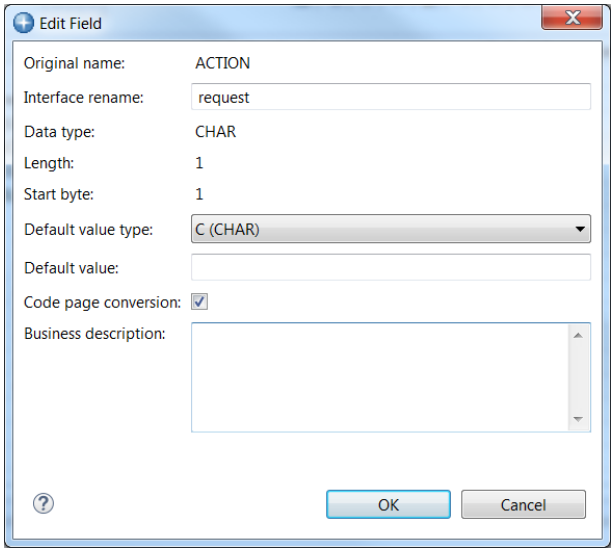
___12. In this window, you can edit and change the property name (e.g. *Interface name*) or exclude specific fields entirely from the interface. Either can be done by selecting a field and right mouse button clicking or by selecting a field and using the desired tool icon in the Service Interface toolbar. Let's try both techniques to remove the STAT and COMMENT fields.

___13. Select field *STAT* and right mouse button click and select the *Exclude field from interface* option on the list of options.

___14. Next select field *COMMENT* and use the *Exclude selected fields(s) from the interface* tool icon.

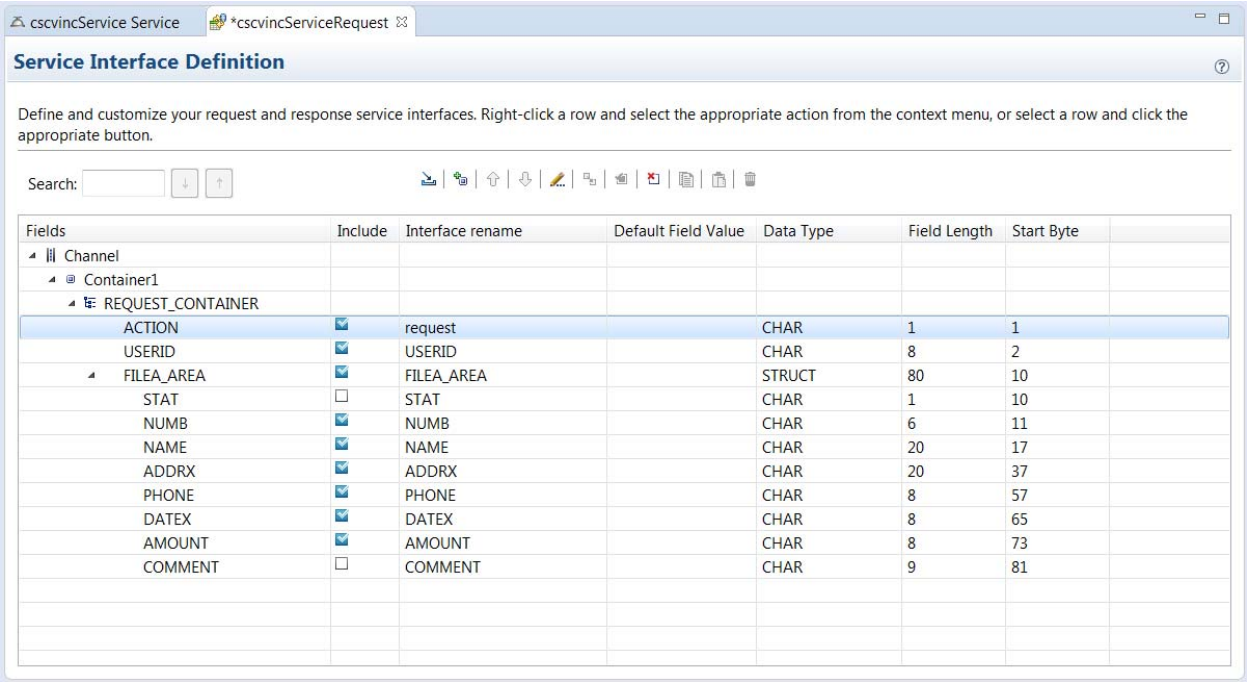
___15. Notice that the check boxes besides these two fields are now unchecked. (You could have simply unchecked the box to accomplish the same results.)

___16. Next select field *ACTION* and rename it's interface to ***request*** using the right mouse button technique or the *Edit selected field* icon in the tool bar.



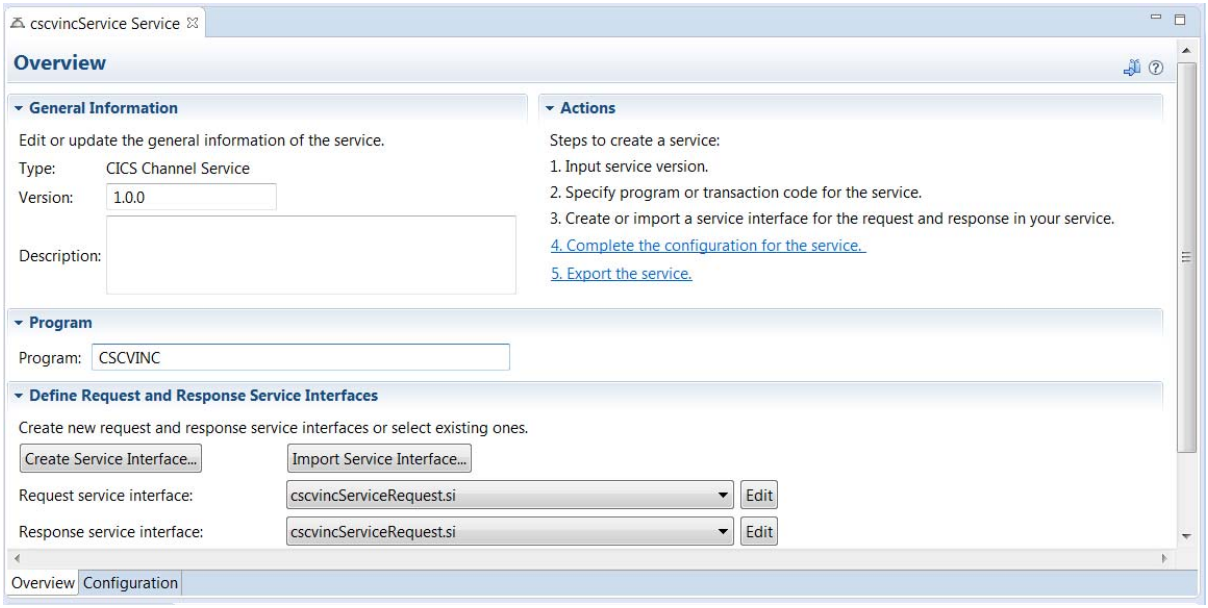
Mitch Johnson (mitchj@us.ibm.com)

When finished your service definition interface should look like this.



___17. Close the Service Interface Definition window by clicking on the white X in the tab being sure to save the changes. Note now that the *Request service interface* and the *Response service interface* areas have now been populated with *cscvincServiceRequest.si*. Also note that you can use their respective **Edit** buttons to return to the *Service Interface Definition* window for each interface.

___18. The response container from CSCVINC is different from the request container so the Response service interface needs to be created.



Mitch Johnson (mitchj@us.ibm.com)

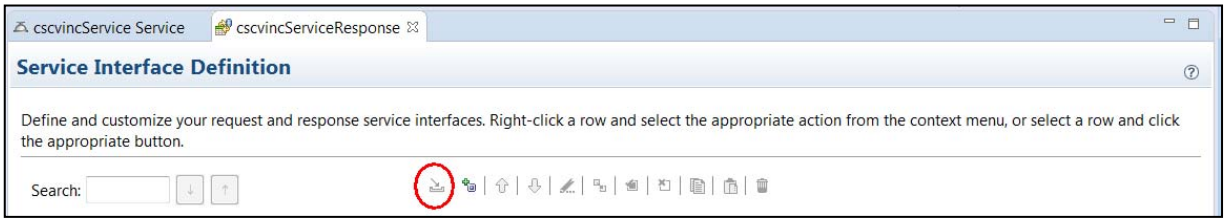
___19. Click the **Create Service Interface** button to create the response service interface required by this API and enter a *Service interface name* of ***cscvincServiceResponse***. Click **OK** to continue.

___20. This will open a *Service Interface Definition* window.

___21. The first step is to import the COBOL copy book that represents the outbound or response CONTAINER.

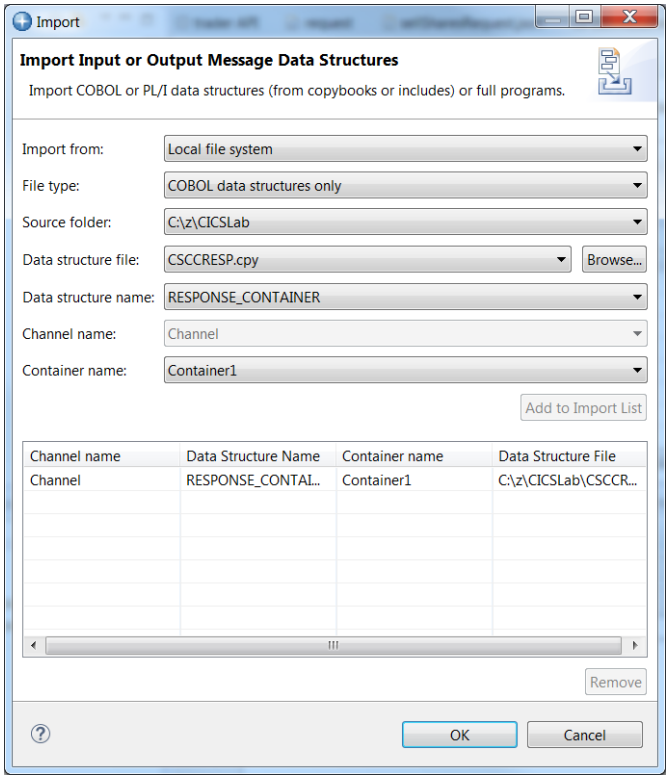
```
       01 Response-Container.
           03  ACTION           PIC X(1).
           03  CEIBRESP         PIC S9(8) COMP.
           03  CEIBRESP2        PIC S9(8) COMP.
           03  USERID           PIC X(8).
           03  FILEA-AREA.
               05  STAT         PIC X.
               05  NUMB         PIC X(6).
               05  NAME         PIC X(20).
               05  ADDRX        PIC X(20).
               05  PHONE        PIC X(8).
               05  DATEX        PIC X(8).
               05  AMOUNT       PIC X(8).
               05  COMMENT      PIC X(9).
```

___22. On the *Service Interface Definition* window select *CONTAINER1* and click the *Import COBOL or PL/I data structure into the service interface* icon to start the import process.

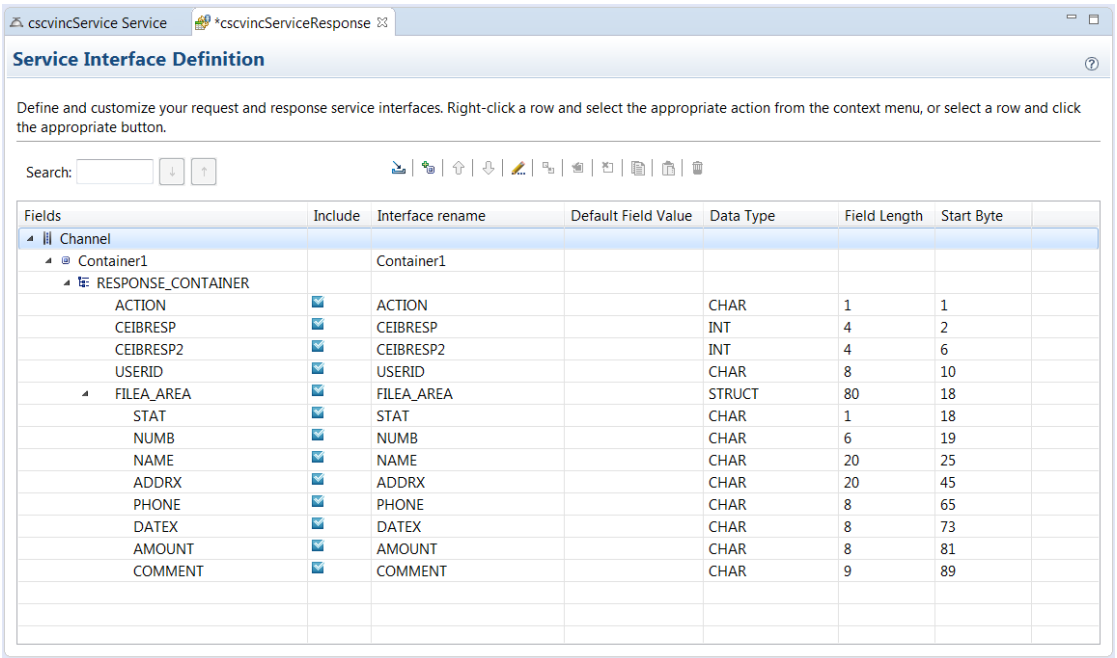


Mitch Johnson (mitchj@us.ibm.com)

___23. This will open the *Import* window. On this window select *Local file system* as source of the import and *COBOL data structure only* as the *File type*. Press the **Browse** button and **Open** directory *C:\z\CICSLAB* and then select file *CSCCRESP*.cpy and click **Open** to import this file into this project. Click the **Add to Import List** button to continue.



___24. Click **OK** and when you expand *Container1* you will see the COBOL 'variables' that have been imported into the service project.



Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
Channel						
Container1		Container1				
RESPONSE_CONTAINER						
ACTION	☑	ACTION		CHAR	1	1
CEIBRESP	☑	CEIBRESP		INT	4	2
CEIBRESP2	☑	CEIBRESP2		INT	4	6
USERID	☑	USERID		CHAR	8	10
FILEA_AREA	☑	FILEA_AREA		STRUCT	80	18
STAT	☑	STAT		CHAR	1	18
NUMB	☑	NUMB		CHAR	6	19
NAME	☑	NAME		CHAR	20	25
ADDRX	☑	ADDRX		CHAR	20	45
PHONE	☑	PHONE		CHAR	8	65
DATEX	☑	DATEX		CHAR	8	73
AMOUNT	☑	AMOUNT		CHAR	8	81
COMMENT	☑	COMMENT		CHAR	9	89

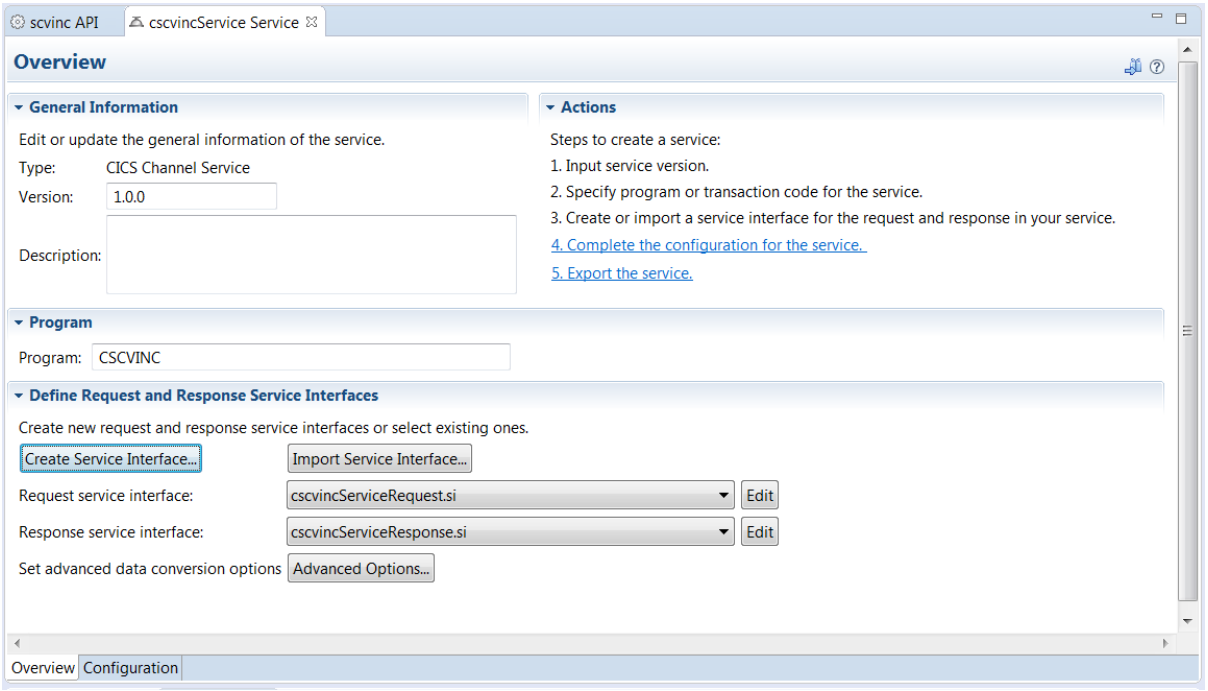
N.B. the interface names were derived from the COBOL source shown earlier.

Mitch Johnson (mitchj@us.ibm.com)

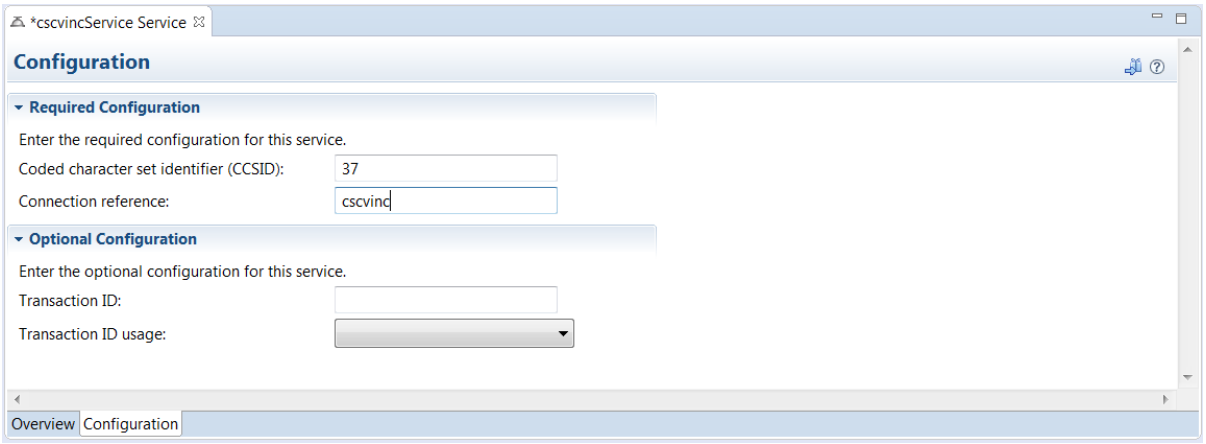
___25. Again, in this view we could exclude fields from the interface or rename fields. To save time, excluding and renaming fields will not be done in the response message in this exercise.

___26. Close the *cscvincServiceResponse* view by clicking on the white X.

___27. Use the pull-down arrow beside *Response service interface* to select *cscvincServiceResponse.si*.



___28. Next, we need to identify a connection reference for which CICS region will be used. Click on the Configuration tab at the bottom of the *Overview* window to display the *Configuration* window. Enter ***cscvinc*** in the area beside *Connection reference*.



Tech-Tip: The *Connection reference* identifies the *cicsIpicConnection* element or *cicsLocalConnection* element to be used to access a CICS region in the z/OS Connect EE configuration, see cscvinc.xml on page 34.

___28. Save the *cscvincService service* either by closing the tab or using the **Ctrl-S** key sequence.

The service now needs to be make available for developing the API and for deployment to the z/OS Connect EE server.

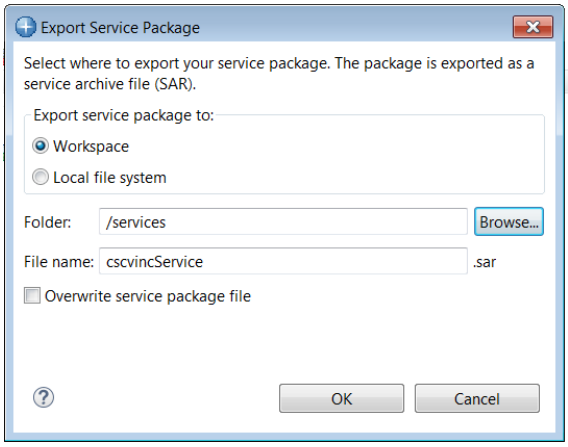
Export and deploy the Service Archive files

Before a service interface can be used it must be exported to create Service Archive (SAR) file and deployed as a SAR file to the server. The exported SAR is used in developing an API in the z/OS Connect EE Toolkit. This section describes the process for exporting and deploying SAR files.

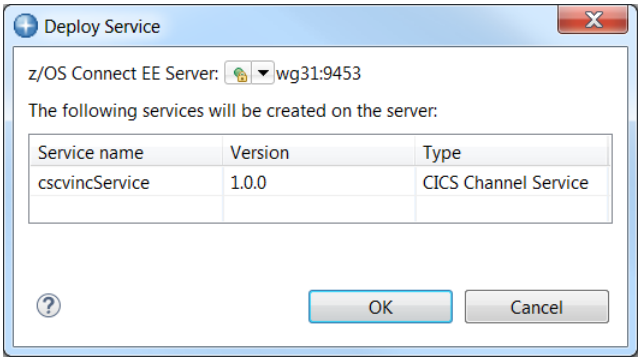
___1. First 'export' the service interface as a SAR into another project in the z/OS Connect EE Toolkit. Select ***File*** on the tool bar and then on the pop up select **New → Project**. Expand the *General* folder and select *Project* to create a target project for exporting the Service Archive (SAR) files. Click **Next** to continue.

___2. On the *New Project* window enter ***Services*** as the *Project name*. Click **Finish** to continue. This action will add a new project in the *Project Explorer* named *Services*. If this project already exists continue with Step 3.

___3. Select the *cscvincService* service project and right mouse button click. On the pop-up selection select **z/OS Connect EE → Export z/OS Connect EE Service Archive**. On the *Export Services Package* window select the radio button beside *Workspace* and use the **Browse** button to select the *Services* folder. Click **OK** to continue.



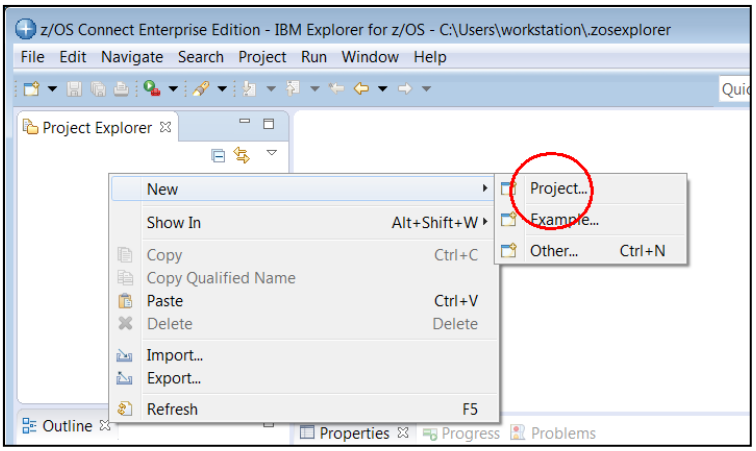
___4. Select the *cscvincService* service project and right mouse button click again and on the pop-up selection select **z/OS Connect EE → Deploy Service to z/OS Connect EE Server**. On the *Deploy Service* window select the target server (*wg31:9453*) and click **OK** twice to continue.



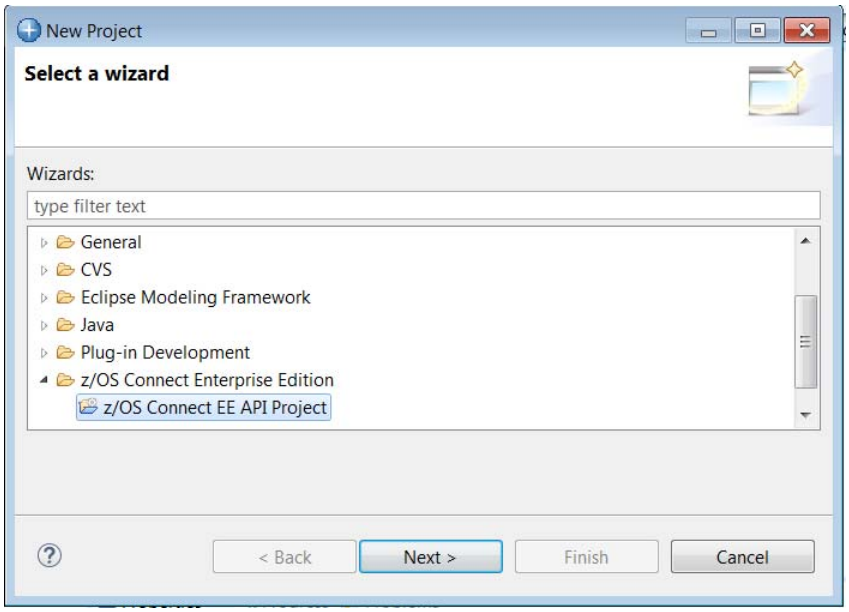
Mitch Johnson (mitchj@us.ibm.com)

Create the Cscvinc 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*:

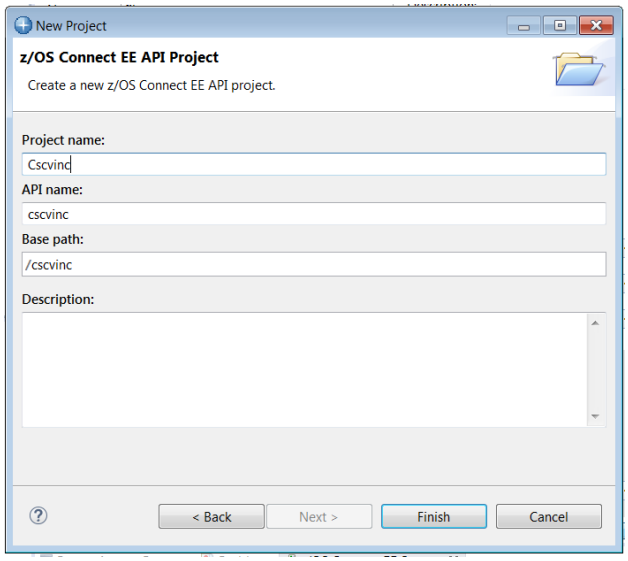


___2. In the *New Project* screen, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE API Project* and then click the **Next** button.



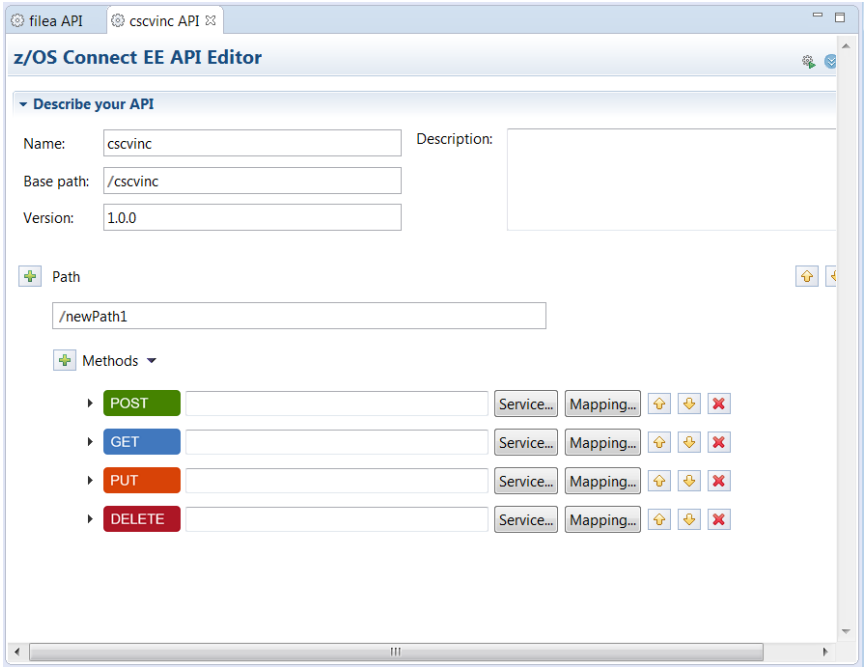
Mitch Johnson (mitchj@us.ibm.com)

___3. Enter ***Cscvinc*** for the *Project name.* Be sure the *API name* is set to ***cscvinc*** and the *Base path is* set to ***/cscvinc.*** Click **Finish** to continue.



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.



Summary

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

Mitch Johnson (mitchj@us.ibm.com)

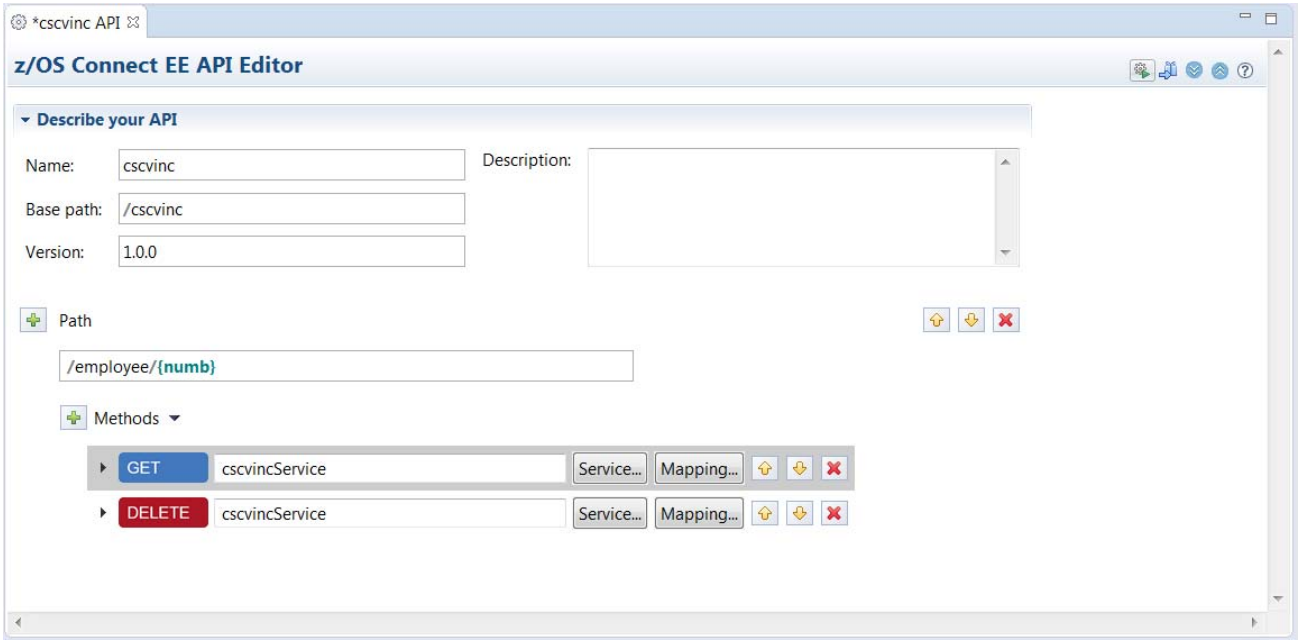
Compose the API for the CICS Container Application

The API for the CICS container application will have two paths. One path which includes a path parameter which identifies the key of the record to be retrieved or deleted and another path with no path parameter for inserting and/or updating a record. In this case the key is one of the fields in the request message.

___1. Start by entering a *Path* of ***/employee/{numb}*** in the *z/OS Connect EE API Editor* view. Remove the *POST* and *PUT* methods by the red X's next to each.

___2. Click the **Service** button beside the *GET* method and on the *Select a z/OS Connect EE Service* window click the ***Workspace*** *button*. On the *Import z/OS Connect EE Services* window expand the *services* folder and select *cscvincService.sar*. Click OK three times to import this SAR into this project.

___3. Click the **Service** button beside the *Delete* method and on the *Select a z/OS Connect EE Service* window select *cscvincService.sar* and click **OK**.

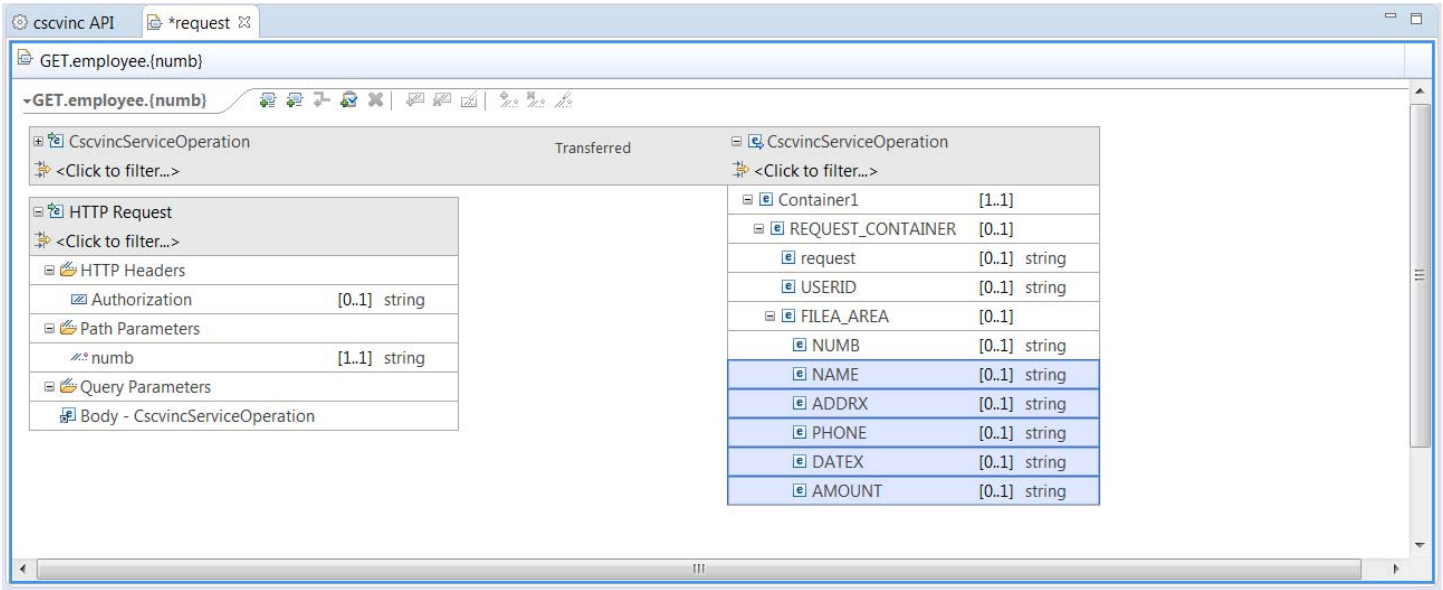


___4. Next, click on the **Mapping** button beside the **GET** method and then select *Open Request Mapping*:

___5. In the mapping view that opens, go to the right side of the mapping (which represents the COPYBOOK fields), and click the little + signs to expand *Request_Container*. You should see fields that correspond to the fields defined in the original COBOL copy book *CSCCREQ* in *USER1.ZCEE.CNTL*.

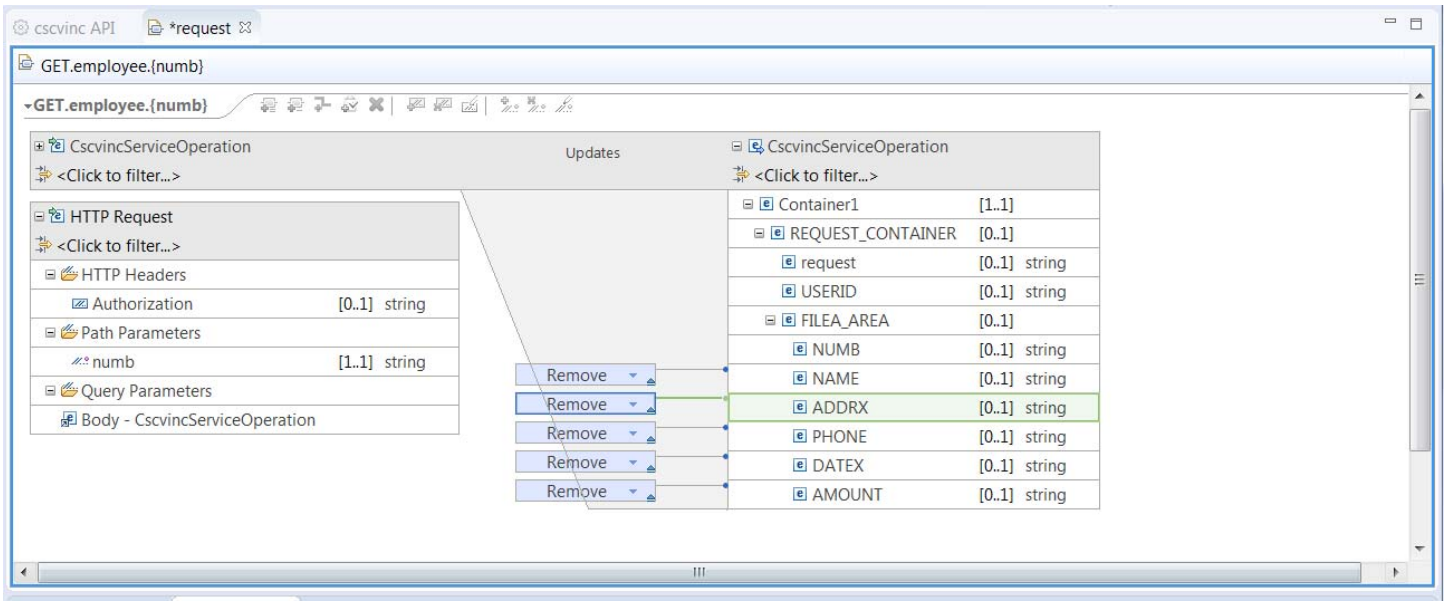
```
       01 Request-Container.
           03  ACTION           PIC X(1).
           03  USERID           PIC X(8).
           03  FILEA-AREA.
               05  STAT         PIC X.
               05  NUMB         PIC X(6).
               05  NAME         PIC X(20).
               05  ADDRX        PIC X(20).
               05  PHONE        PIC X(8).
               05  DATEX        PIC X(8).
               05  AMOUNT       PIC X(8).
               05  COMMENT      PIC X(9).        .
```

___6. Use the slider bar to fully expose the *Request_Container* structure. Use the left mouse button and draw a dotted line box that fully includes the *name*, *addrx*, *phone*, *datex* and *amount* and 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.

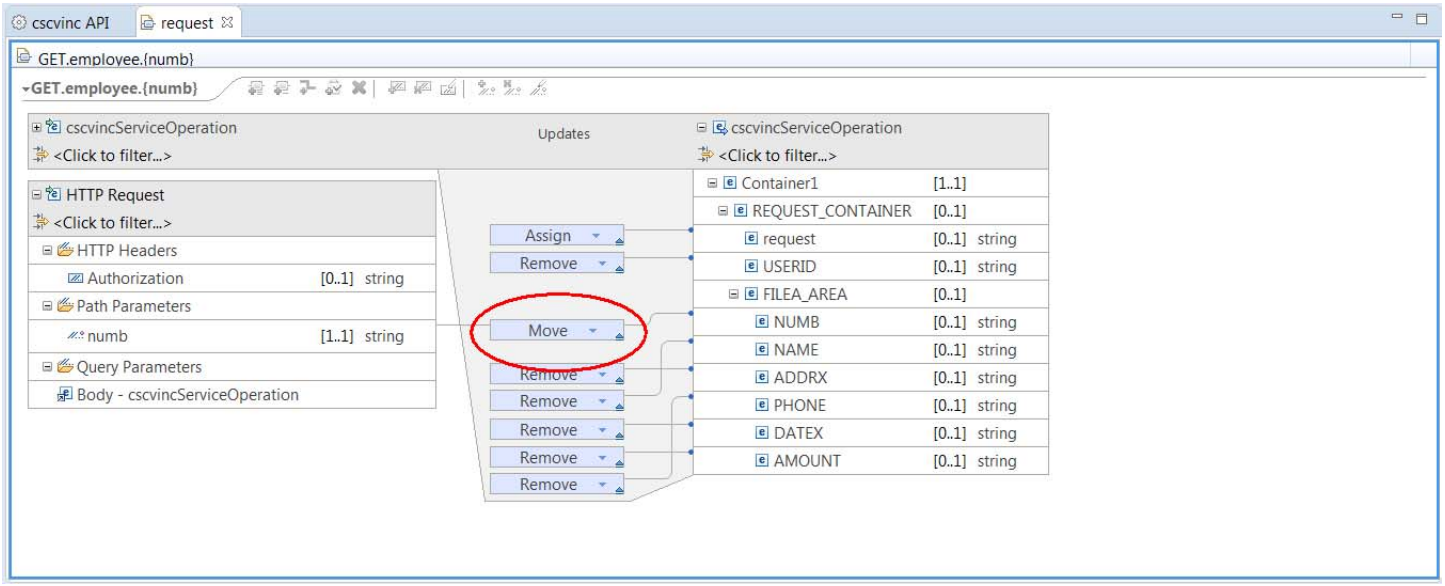
___7. This action generates multiple “remove” requests (see below) for the selected fields. These fields are not required for a **GET** method, so these will not be exposed to the REST clients using this method.



___8. Right mouse button click USERID and select *Add Remove transform* to remove this field from the interface.

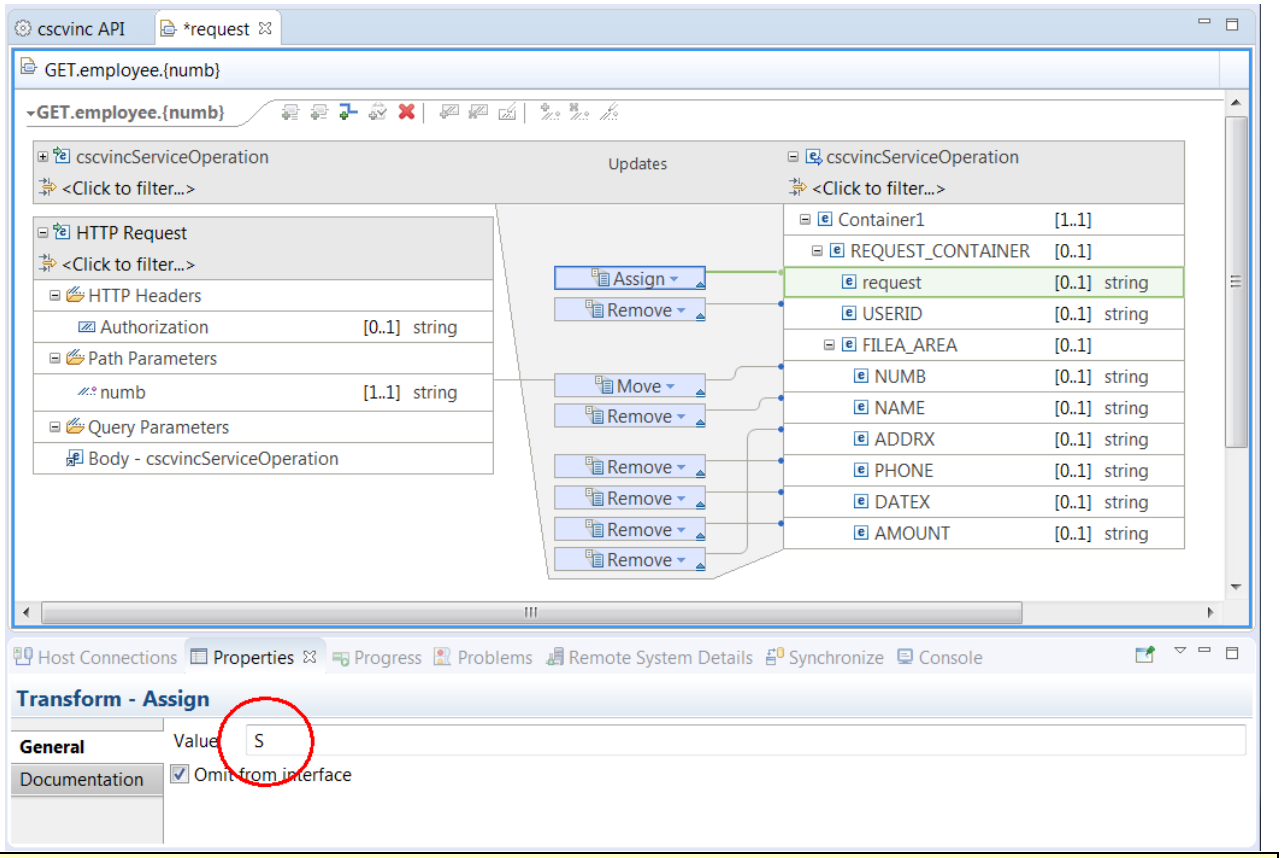
Mitch Johnson (mitchj@us.ibm.com)

___9. This leaves two fields in the *Request_Container* structure, *request* and *numb*. The contents of the *numb* filed should be provided by the *numb* path parameter from the URL. On the left-hand side of the view select the *numb* path parameter with the left mouse button and drag it without releasing the mouse button) to the *numb* field on the right-hand side to create a “move” connection from the *numb* path parameter to the *numb* field in the *Request_Container* structure.



Mitch Johnson (mitchj@us.ibm.com)

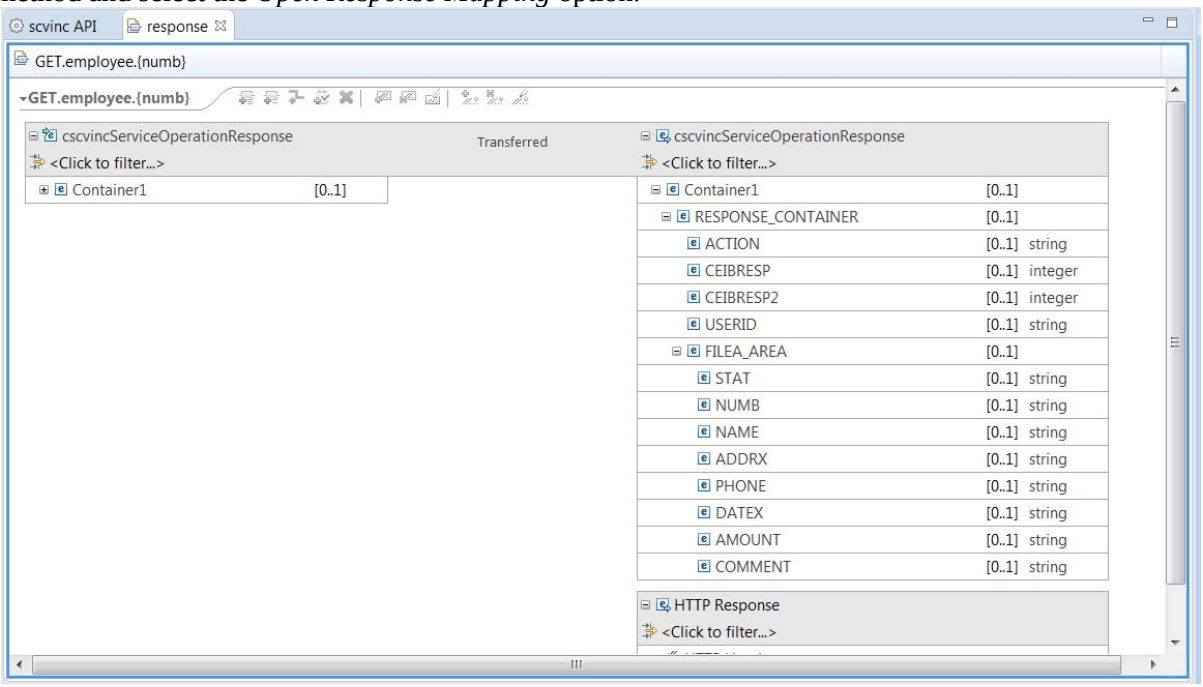
___10. The only *Request_Container* field left unhandled now is the *request* field. The target application uses this field to determine what action should be performed. For a **GET** method request this field should be set to a value of ***S***. Select the *request* field and right button click and select the *Add Assign transform* option. This action opens a *Properties* tab in the lower view. In this tab, a value can be entered which will be used to populate this field when a **GET** method is invoked. Enter ***S*** in the area beside *Value* in the *Properties* tab.



Remember: The request field determines which function the application performs. The valid values for *request* are I(insert) for the POST method, S(select) for the GET method, U(update) for the PUT method and D(delete) for the DELETE method. Creating another service. This API can be extended to support PUT, DELETE and POST by simply remapping the request message as appropriate.

___11. Use the ***Ctrl-S*** key sequence to save all changes and close the *GET.employee.{numb}* view.

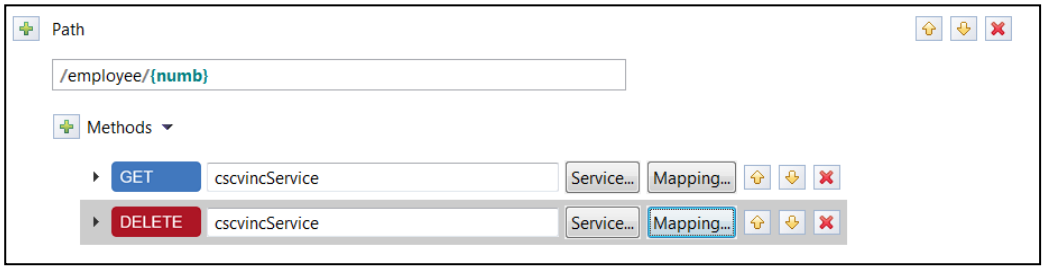
___12. For **GET** methods we want all fields exposed to the REST client. The default response mapping will return all fields so no special mapping is required for this method. To review the fields that will be returned click the **Mapping** button beside the **GET** method and select the *Open Response Mapping* option.



The Response_Container structure was built using the contents of the CSCCRESP COPYBOOK

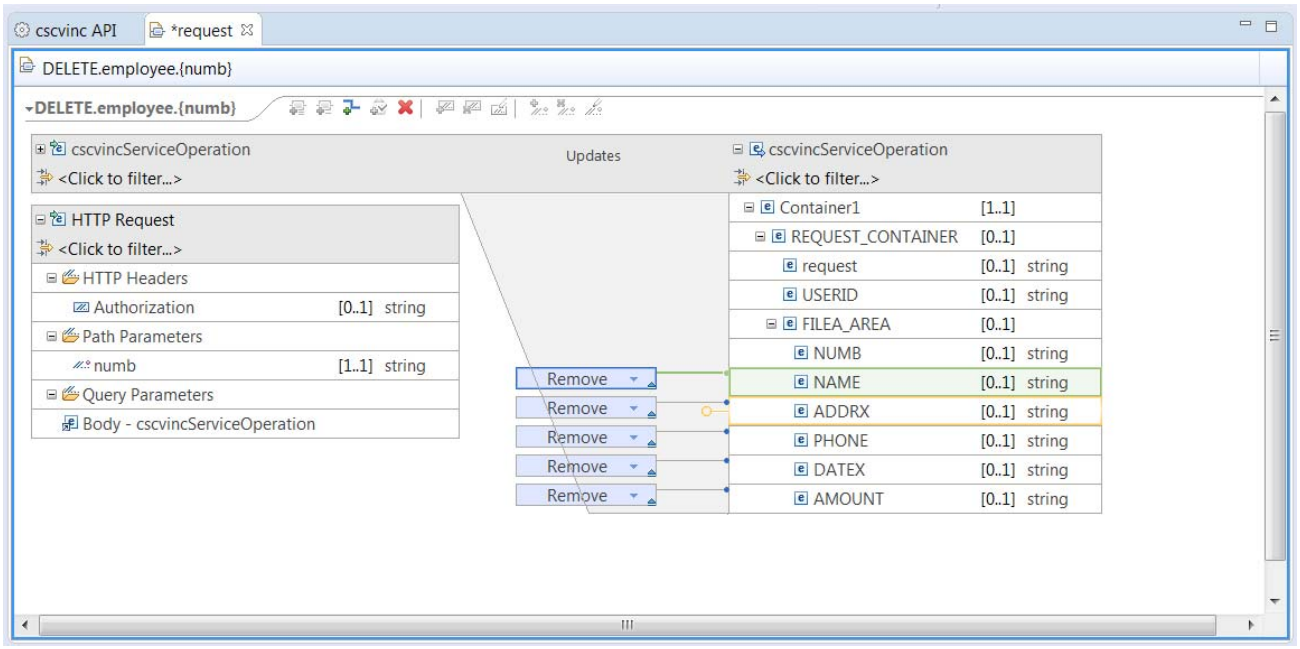
```
01 Response-Container.
   03  ACTION         PIC X(1).
   03  CEIBRESP      PIC S9(8) COMP.
   03  CEIBRESP2     PIC S9(8) COMP.
   03  USERID         PIC X(8).
   03  FILEA-AREA.
      05  STAT          PIC X.
      05  NUMB          PIC X(6).
      05  NAME          PIC X(20).
      05  ADDRX         PIC X(20).
      05  PHONE         PIC X(8).
      05  DATEX         PIC X(8).
      05  AMOUNT      PIC X(8).
      05  COMMENT   PIC X(9).
```

___13. Next, click on the **Mapping** button beside the **DELETE** method and then select *Open Request Mapping*:



___14. In the mapping view that opens, go to the right side of the mapping (which represents the COPYBOOK fields), and click the little + signs to expand *Request_Container*. You should see fields that correspond to the fields defined in the original COBOL copy book *CSCCREQ* in *USER1.ZCEE.CNTL*.

___15. Use the slider bar to fully expose the *Request_Container* structure. Use the left mouse button and draw a dotted line box that fully includes the *name*, *addrx*, *phone*, *datex* and *amount* fields. When you release the button, these fields should be selected (the background should be blue). Right mouse button click on any of the selected fields and select the *Add Remove transform* from the list of options.

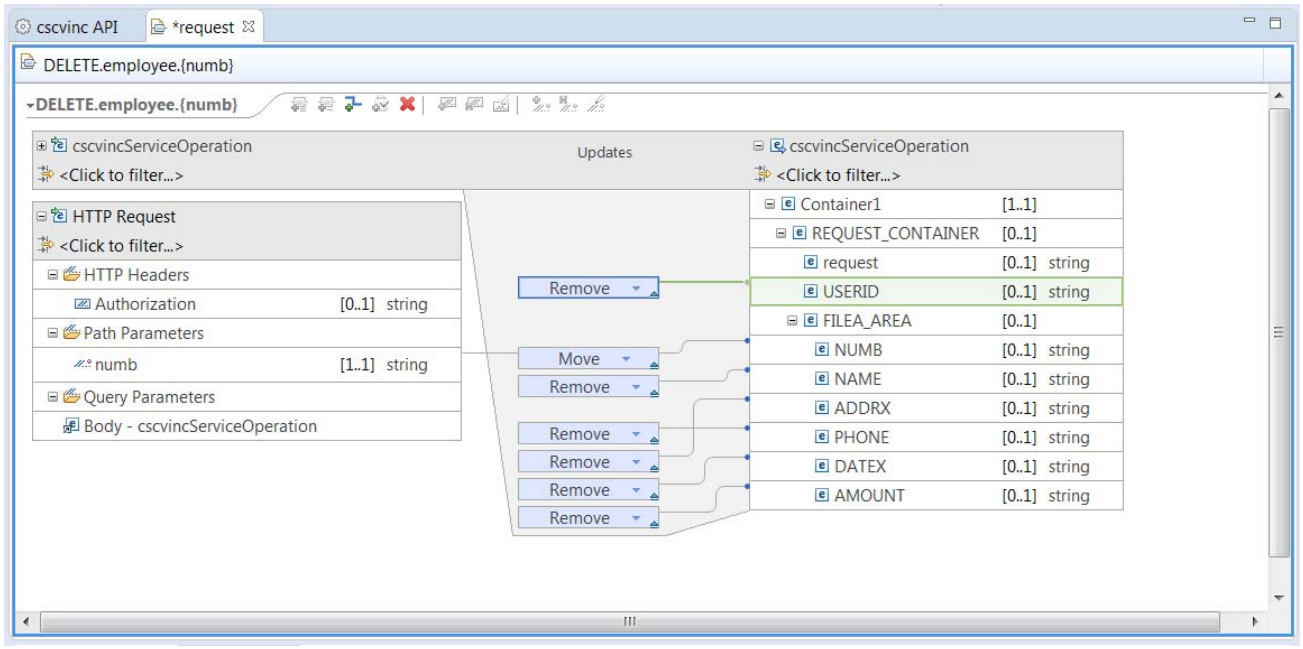


___13. This action generates multiple “remove” requests (see below) for the selected fields. These fields are not required for a **DELETE** method so these will not be exposed to the REST clients using this method.

___14. Right mouse button click USERID and select *Add Remove transform* to remove this field from the interface.

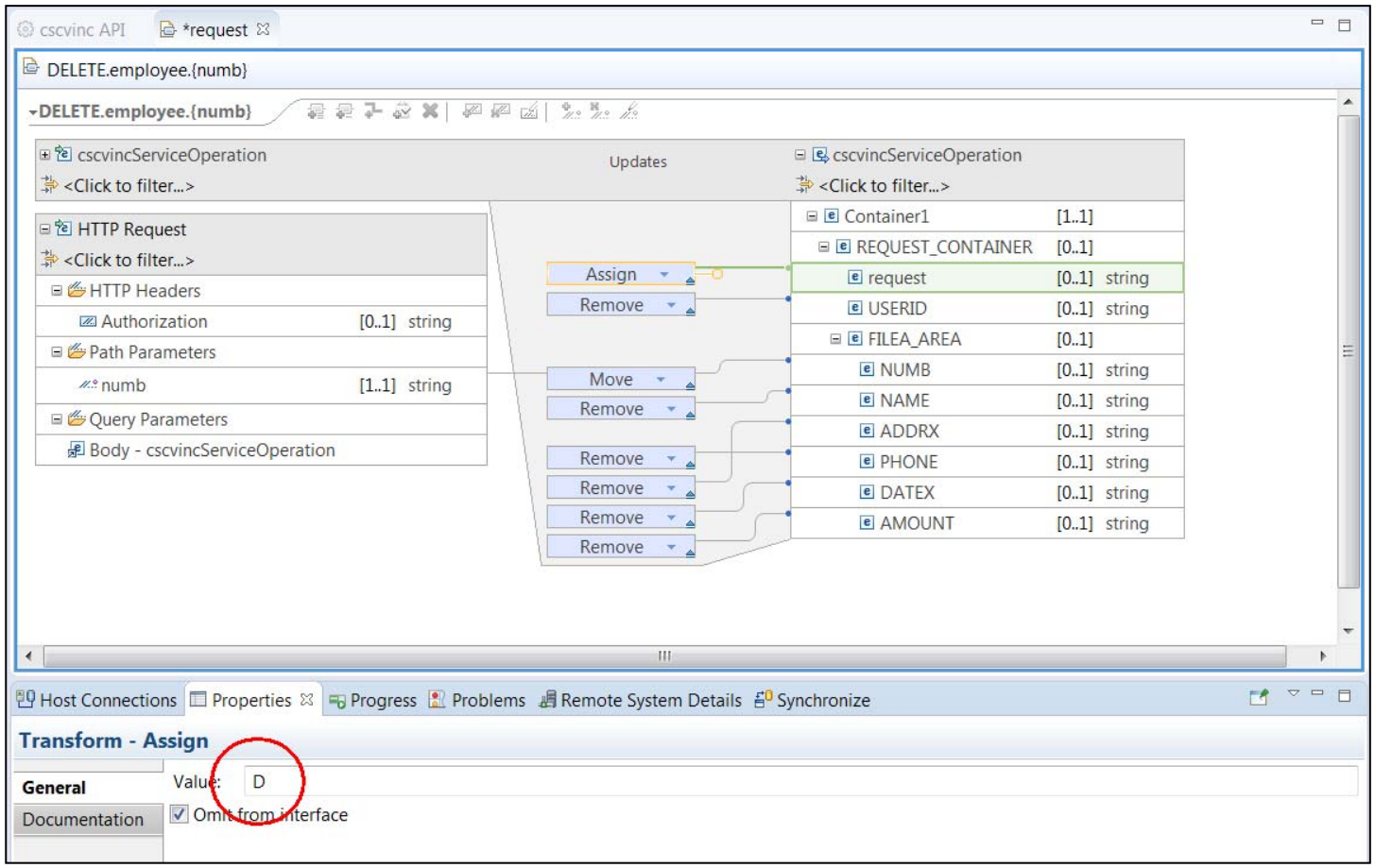
Mitch Johnson (mitchj@us.ibm.com)

___15. This leaves two fields in the *Request_Container* structure, *request* and *numb*. The contents of the *numb* filed should be provided by the *numb* path parameter from the URL. On the left-hand side of the view select the *numb* path parameter with the left mouse button and drag it without releasing the mouse button) to the *numb* field on the right-hand side to create a “move” connection from the *numb* path parameter to the *numb* field in the *Request_Container* structure.



Mitch Johnson (mitchj@us.ibm.com)

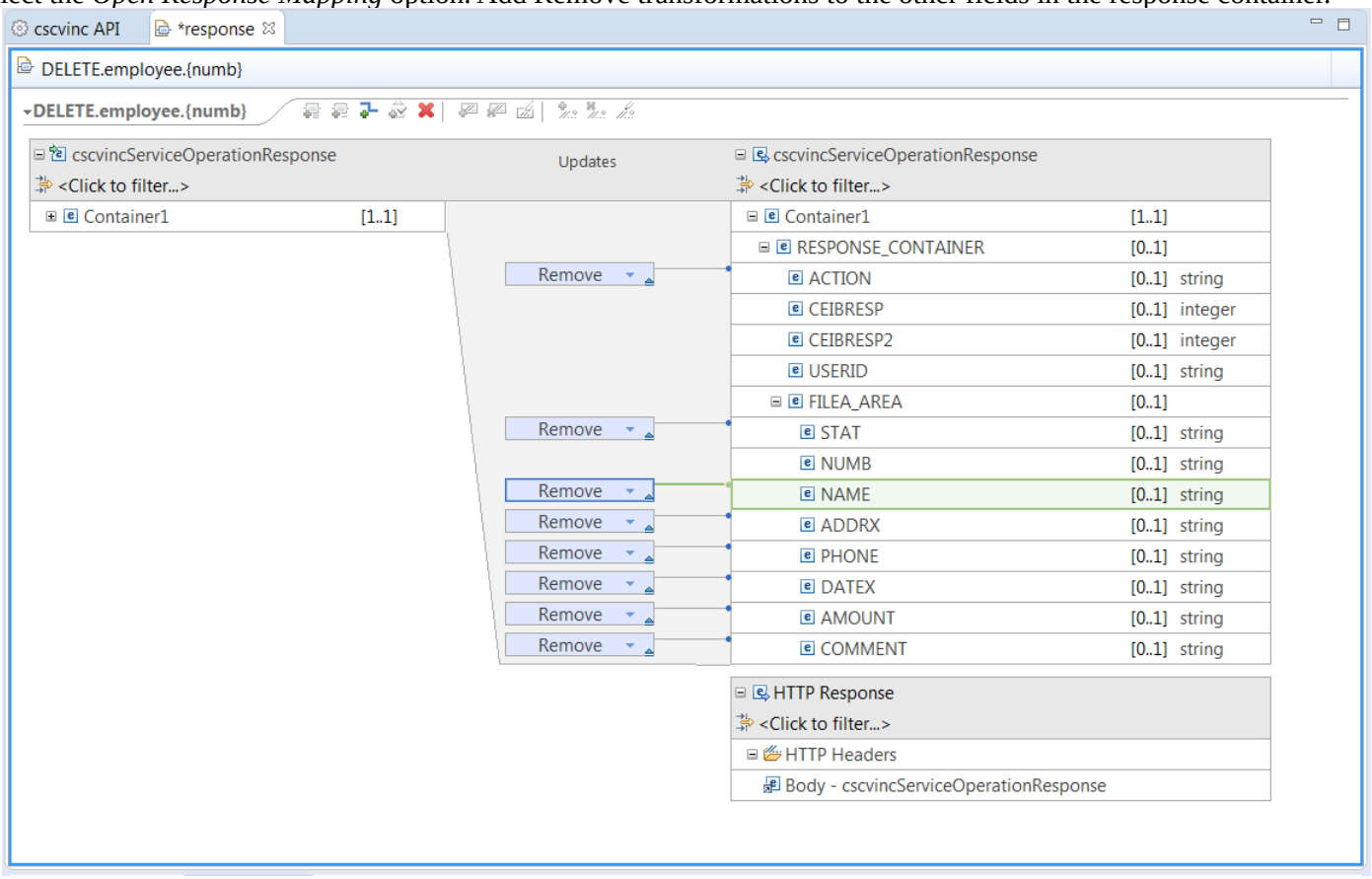
___16. The only *Request_Container* field left unhandled now is the *request* field. The target application uses this field to determine what action should be performed. For a **DELETE** method request this field should be set to a value of ***D***. Select the *request* field and right button click and select the *Add Assign transform* option. This action opens a *Properties* tab in the lower view. In this tab, a value can be entered which will be used to populate this field when a **DELETE** method is invoked. Enter ***D*** in the area beside *Value* in the *Properties* tab.



___17. Use the ***Ctrl-S*** key sequence to save all changes and close the *DELETE.employee.{numb}* view.

Mitch Johnson (mitchj@us.ibm.com)

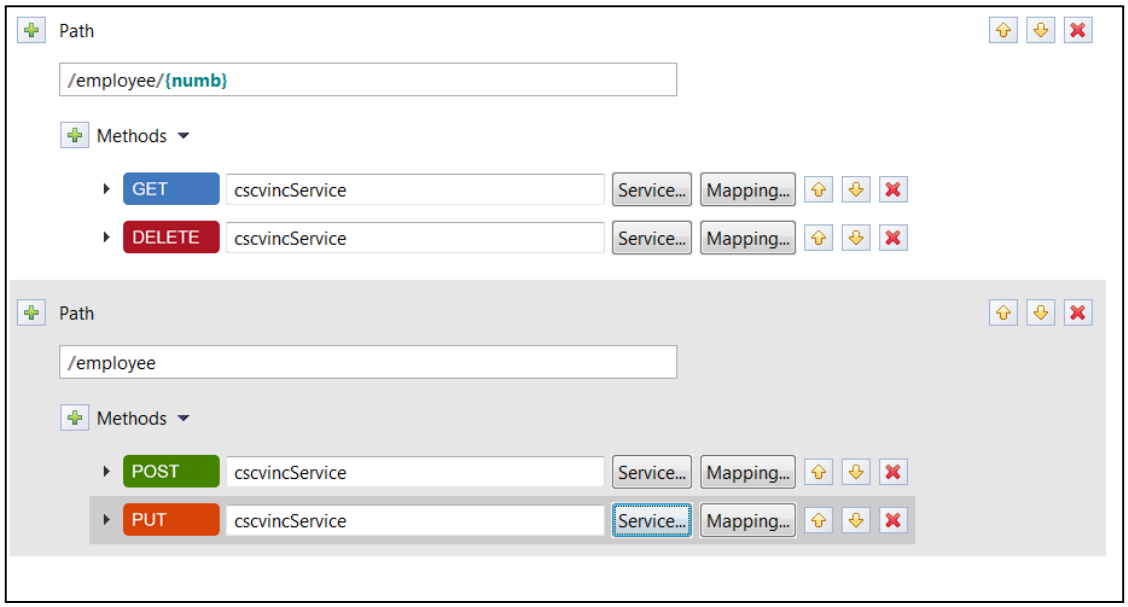
___18. For the **DELETE** method we only want to expose the CEIBRESP, CEIBRESP2, USERID and NUMB fields to the REST client. To make changes to the fields that will be returned click the **Mapping** button beside the **DELETE** method and select the *Open Response Mapping* option. Add Remove transformations to the other fields in the response container.



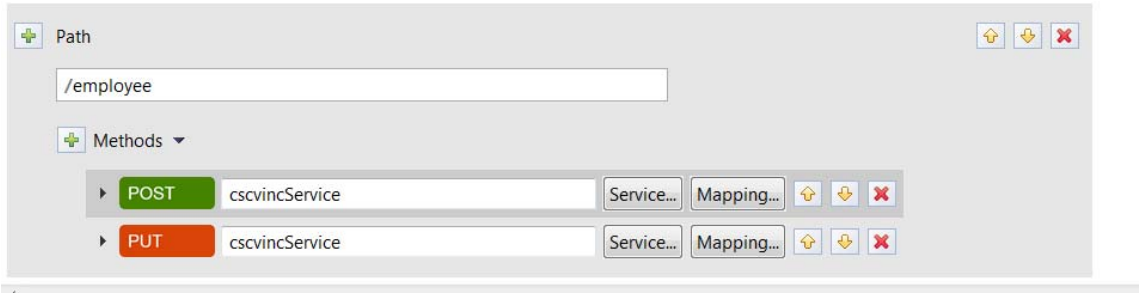
The Response_Container structure was built using the contents of the CSCCRESP copy book.

___19. Next, we want to add a new *Path* for the **POST** and **PUT** methods. Click the plus sign beside PATH and enter a *Path* of ***/employee*** in the *z/OS Connect EE API Editor* view. Remove the *DELETE* and *GET* methods by the red X's next to each.

___20. Click the **Service** button beside the *POST* method and on the *Select a z/OS Connect EE Service* window select *cscvincService.sar* and click **OK**. Repeat these steps for the *PUT* method.

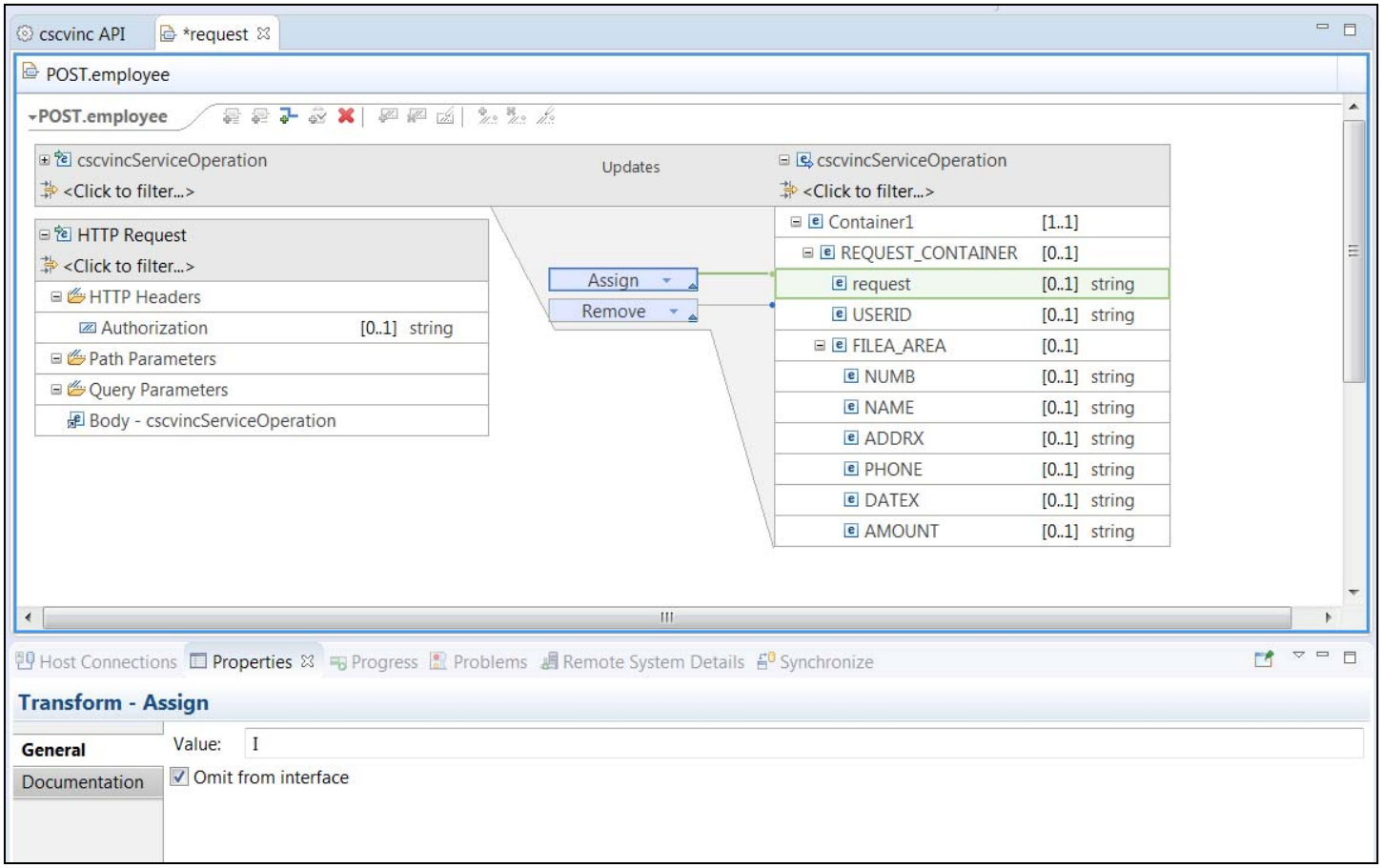


___21. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping*:



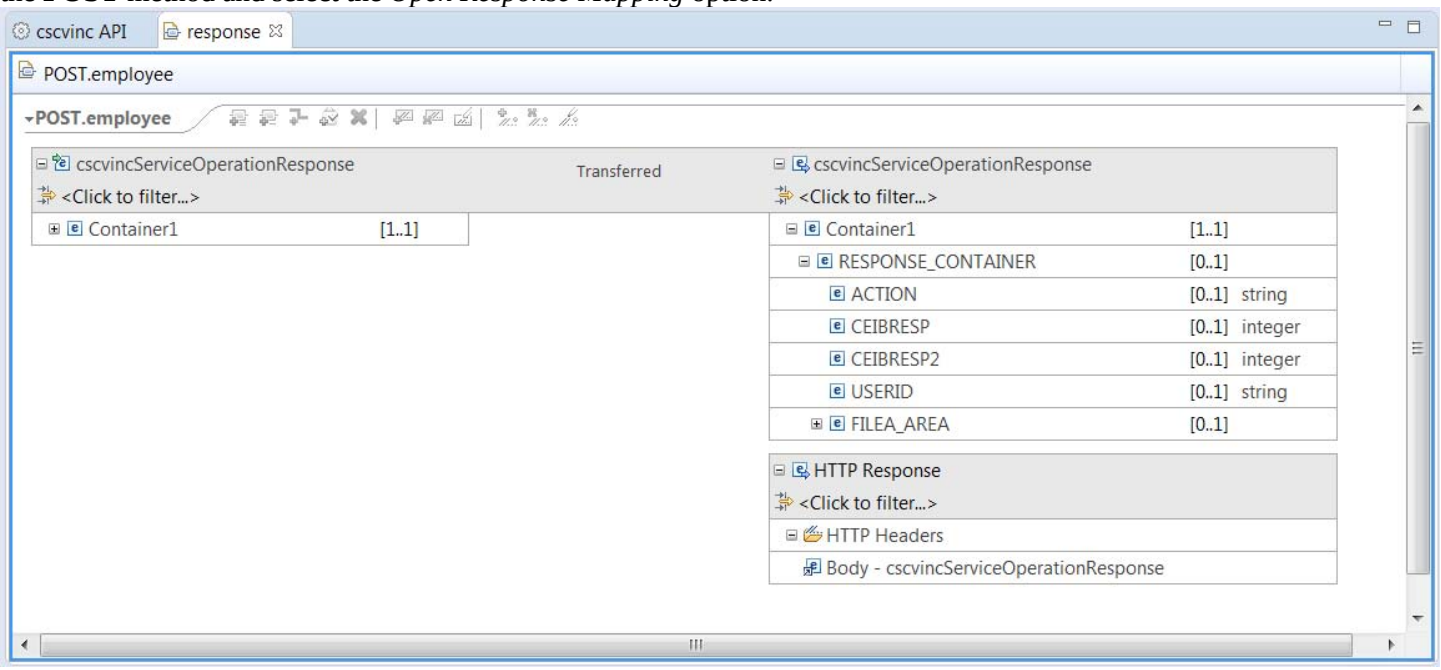
___22. In the mapping view that opens, go to the right side of the mapping (which represents the COPYBOOK fields), and click the little + signs to expand *Request_Container*. You should see fields that correspond to the fields defined in the original COBOL copy book *CSCCREQ* in *USER1.ZCEE.CNTL*.

___19. Use the slider bar to fully expose the *Request_Container* structure. Right mouse button click USERID and select *Add Remove transform* to remove this field from the interface. Select the *action* field and right button click and select the *Add Assign transform* option. This action opens a *Properties* tab in the lower view. In this tab, a value can be entered which will be used to populate this field when a **POST** method is invoked. Enter ***I*** (letter I) in the area beside *Value* in the *Properties* tab



___20. Use the ***Ctrl-S*** key sequence to save all changes and close the *POST.employee.{numb}* view.

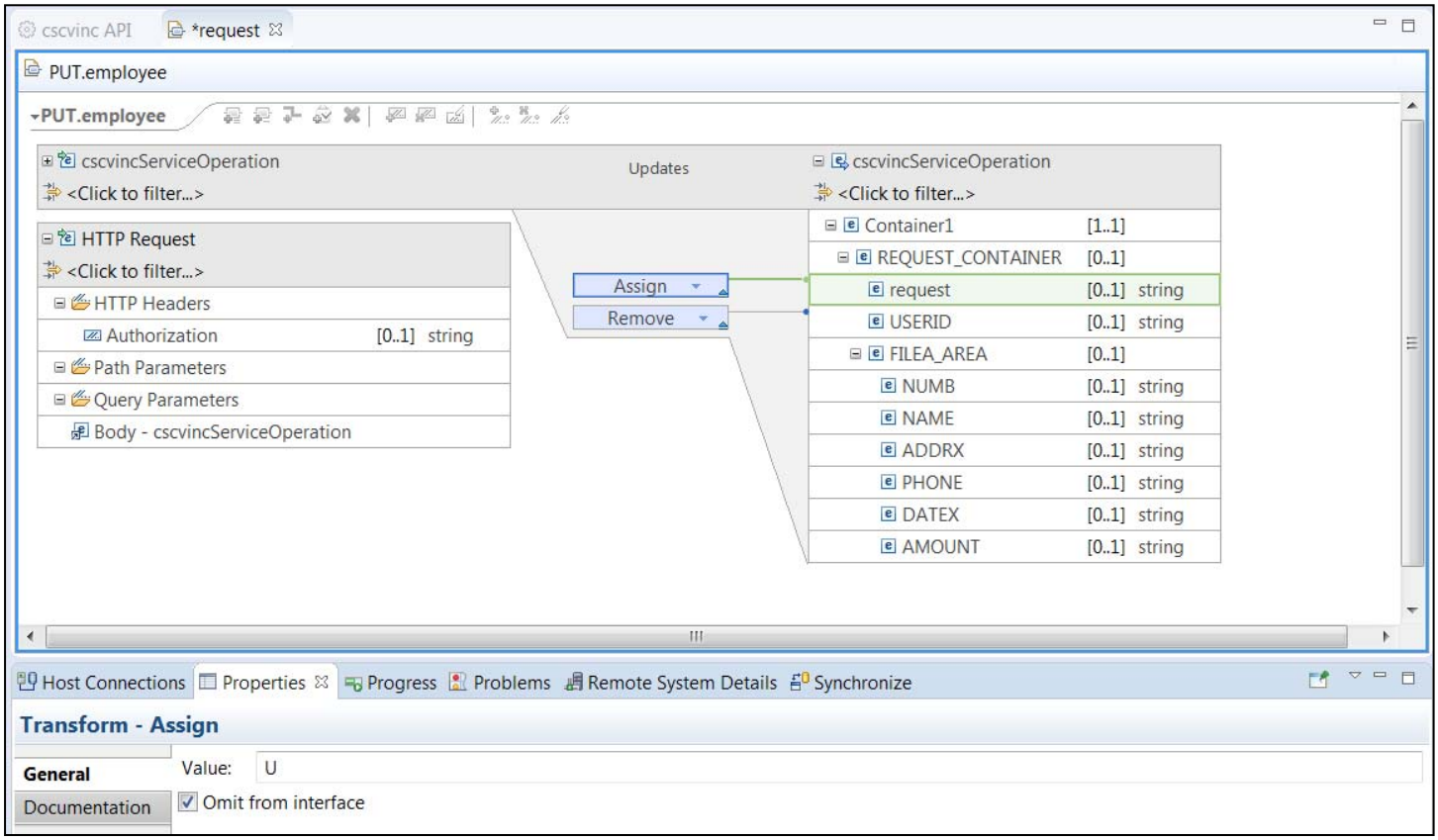
___21. For **POST** methods we want all fields exposed to the REST client. The default response mapping will return all fields so no special mapping is required for this method. To review the fields that will be returned click the **Mapping** button beside the **POST** method and select the *Open Response Mapping* option.



___23. Next, click on the **Mapping** button beside the **PUT** method and then select *Open Request Mapping*:

___24. In the mapping view that opens, go to the right side of the mapping (which represents the COPYBOOK fields), and click the little + signs to expand *Request_Container*. You should see fields that correspond to the fields defined in the original COBOL copy book *CSCCREQ* in *USER1.ZCEE.CNTL*.

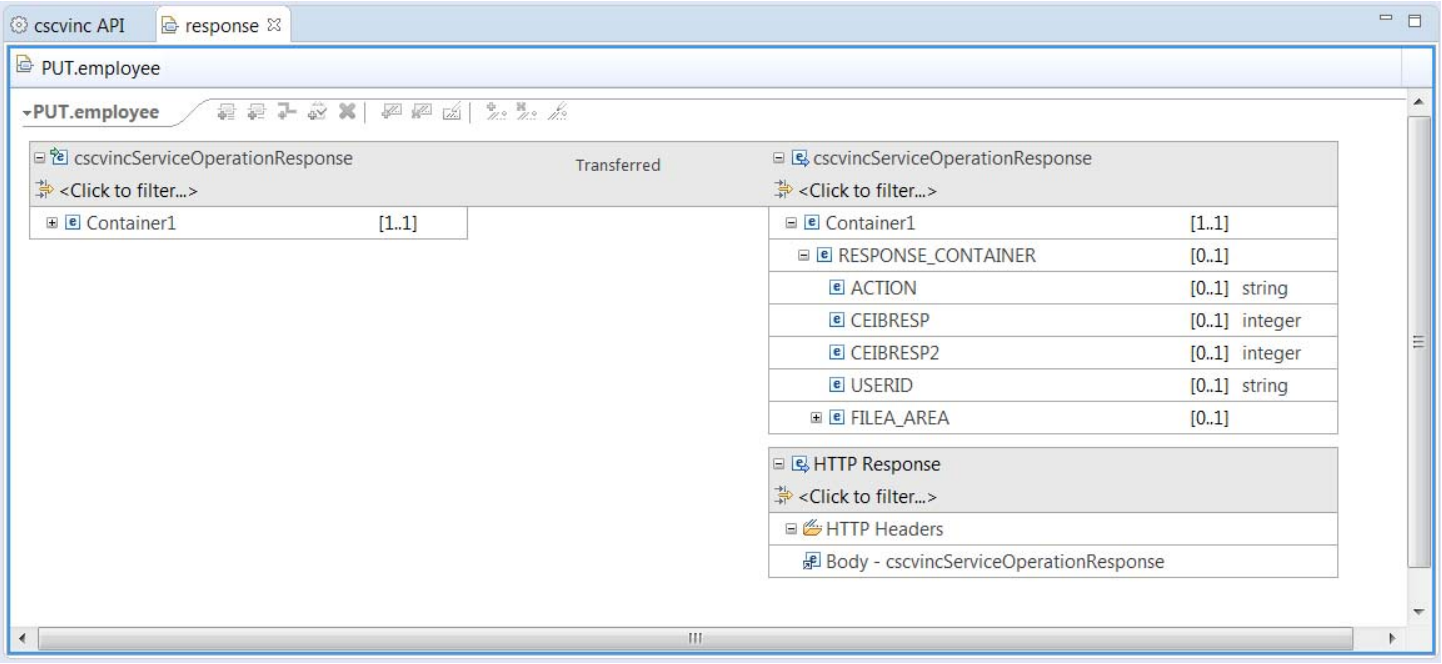
___22. Use the slider bar to fully expose the *Request_Container* structure. Right mouse button click USERID and select *Add Remove transform* to remove this field from the interface. Select the *request* field and right button click and select the *Add Assign transform* option. This action opens a *Properties* tab in the lower view. In this tab, a value can be entered which will be used to populate this field when a **PUT** method is invoked. Enter ***U*** in the area beside *Value* in the *Properties* tab



___23. Use the ***Ctrl-S*** key sequence to save all changes and close the *PUT.employee.{numb}* view.

___24. For **PUT** methods we want all fields exposed to the REST client. The default response mapping will return all fields so no special mapping is required for this method. To review the fields that will be returned click the **Mapping** button beside the **PUT** method and select the *Open Response Mapping* option.

___25. Close all open tabs.



Summary

You created the API, which consists of two paths and the HTTP methods and request and response mapping associated with each. That API will now be deployed into z/OS Connect EE V3.0. Note in this scenario there was one service and the API developer used this one service to support a RESTful API with PUT, POST, GET and DELETE HTTP methods.

Mitch Johnson (mitchj@us.ibm.com)

Deploy the API to a z/OS Connect EE Server

___1. The ***cscvinc*** API *and* ***cscvincService*** service were defined by the inclusion of the *cscvinc.xml* file in the *server.xml*.

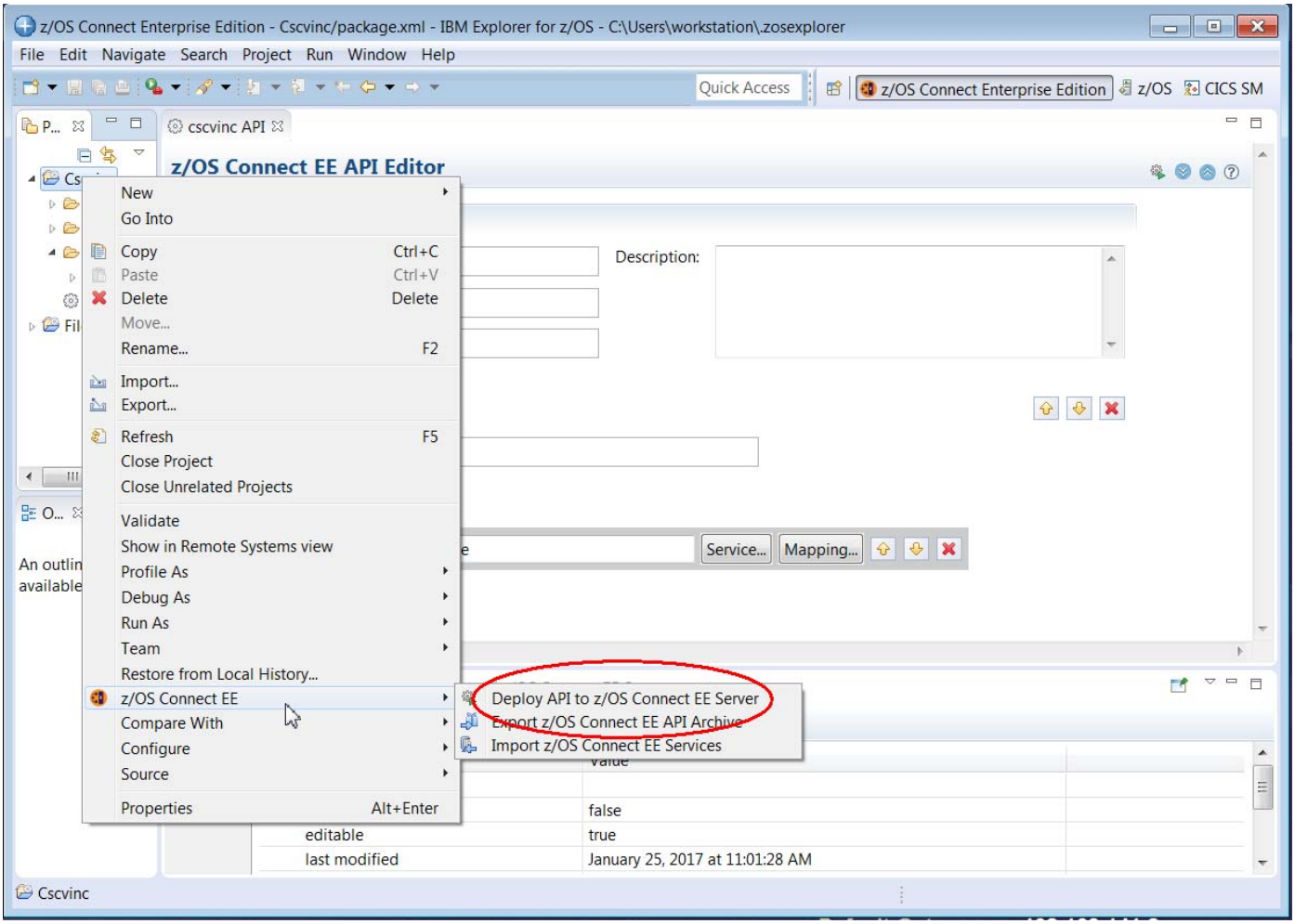
```
<server description="CICS FILEA">

     <featureManager>
       <feature>zosconnect:cicsService-1.0</feature>
     </featureManager>

     <zosconnect_cicsIpicConnection id="cscvinc"
       host="wg31.washington.ibm.com"
       port="1491"/>

</server>                                                    "
```

Figure 1 - cscvinc.xml

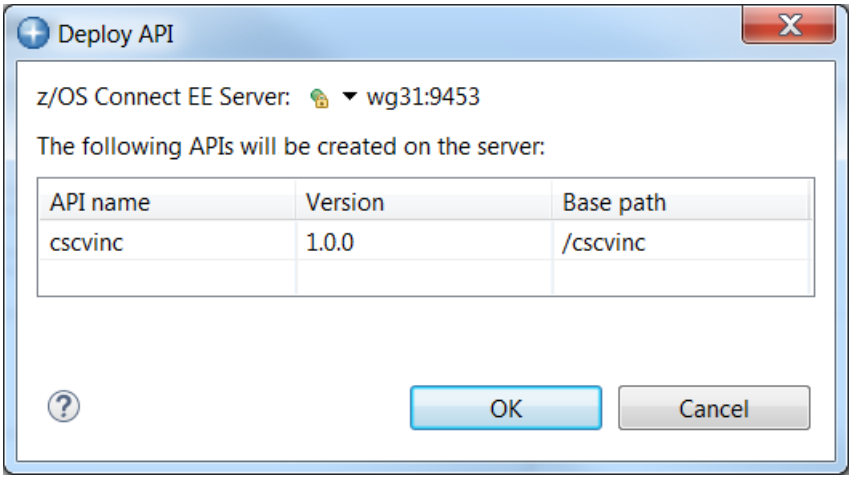
The *zosconnect_cicsIpicConnection* element provides the CICS IPIC information that will be used for communications with the CICS region,

Mitch Johnson (mitchj@us.ibm.com)

___2. In the *Project Explorer* view (upper left), right-mouse click on the *Cscvinc* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.

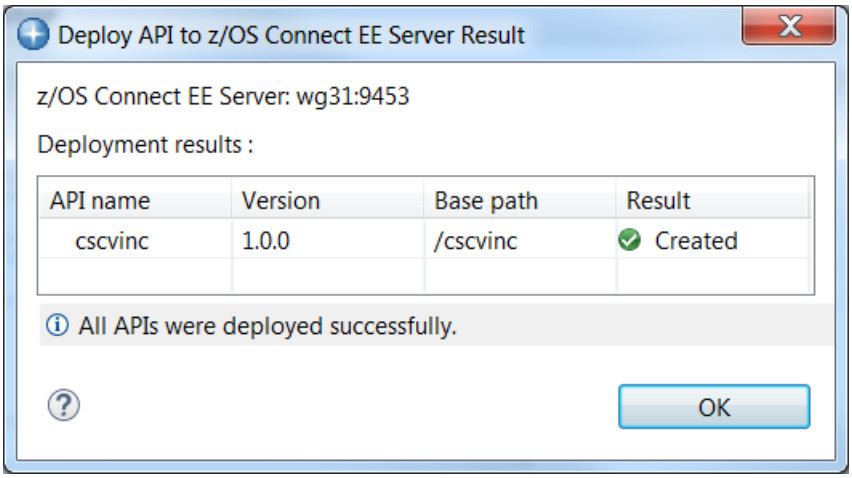


___3. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.



Mitch Johnson (mitchj@us.ibm.com)

___4. The API artifacts will be transferred to z/OS and copied into the */var/ats//zosconnect/servers/zceesrv1/resources/zosconnect/apis* directory.

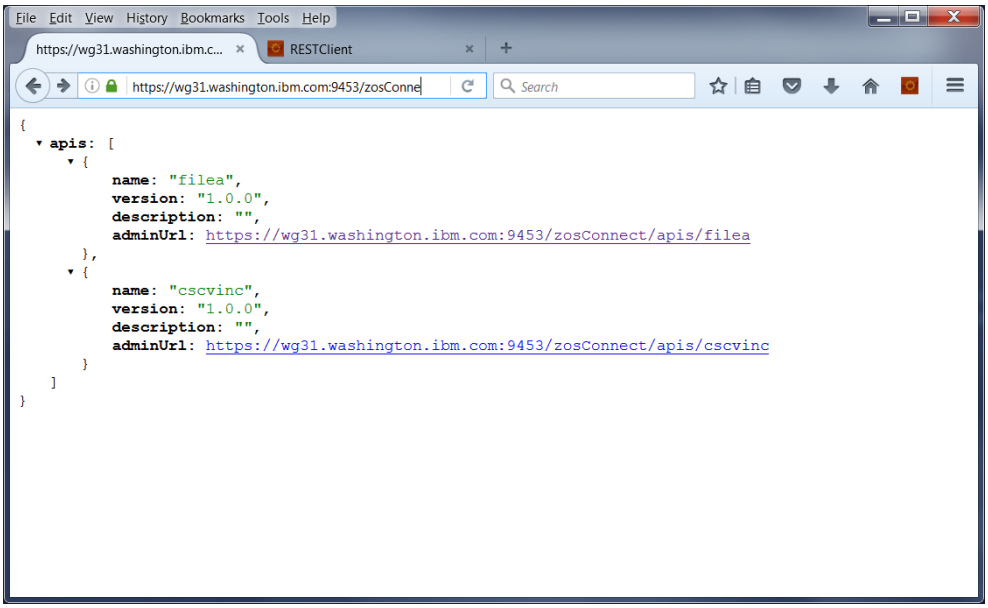


Test the APIs with a CICS Container Application

The CICS application used to test the API accesses the same VSAM file used in the previous section. It supports 4 RESTful APIs by provide a *request* field in the request, adding a record (**I** for **POST**), updating a record (**U** for **PUT**), retrieving a record (**S** for **GET**) and deleting a record (**D** for **DELETE**). To test the z/OS Connect API with this CICS application we will use the same browser plugin used earlier to test the API to the batch application.

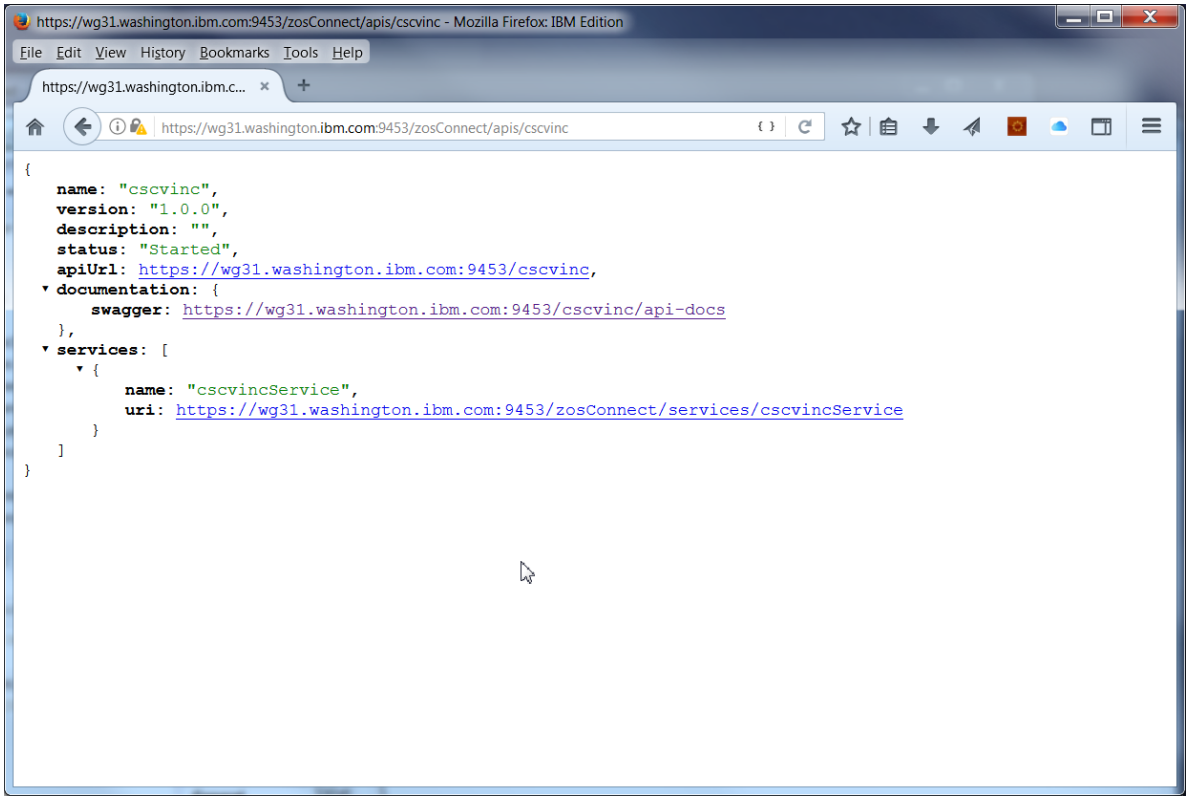
___1. First enter URL ***https://wg31.washington.ibm.com:9453/zosConnect/apis*** in the Firefox browser and you should see the window below. The API *cscvinc* is now displayed. This is because this API was just deployed to this server.

> **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 you for a userid and password. If you do see the prompt, enter the username ***Fred*** and password **fredpwd** (case matters) and click **OK**. Remember we are using basic security and this is the user identity and password defined in the server.xml file.

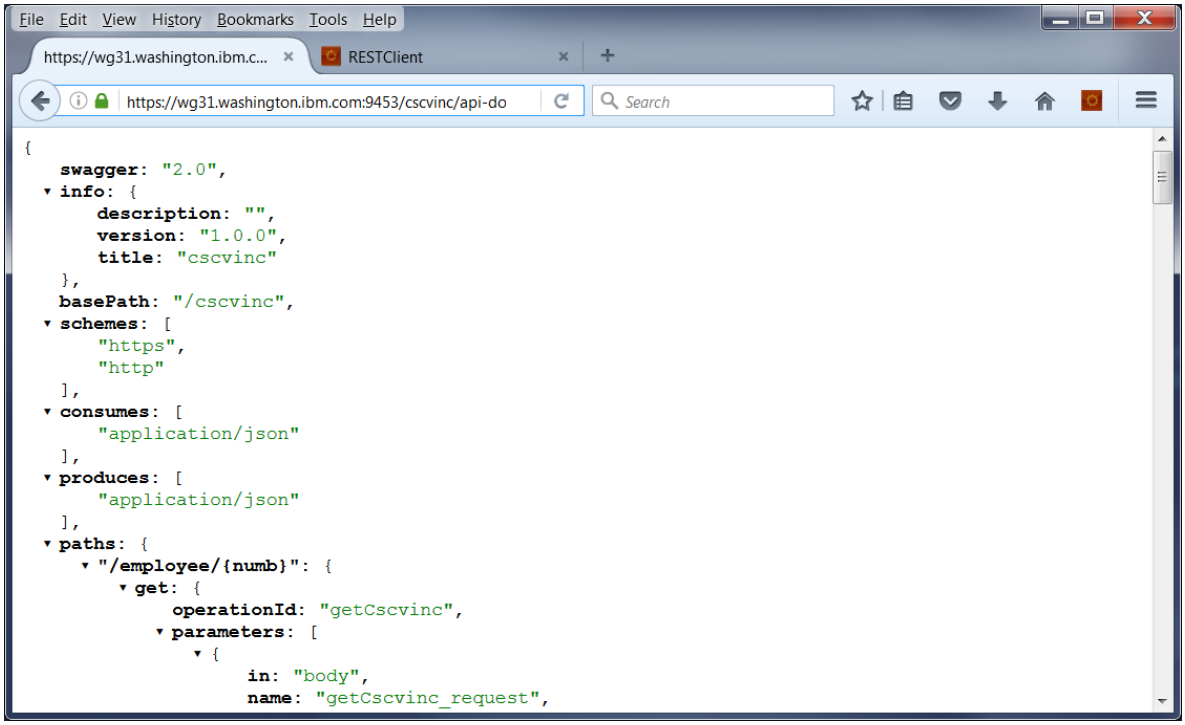


> **Tech Tip:** It is very important to access the z/OS Connect server from a browser prior to any testing using the Swagger UI. Accessing a z/OS Connect URL from a browser starts an SSL handshake between the browser and the server. If this handshake has not performed prior to performing any test the 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 during this exercise.

___4. If you click on *adminUrl* URL the window below should be displayed.



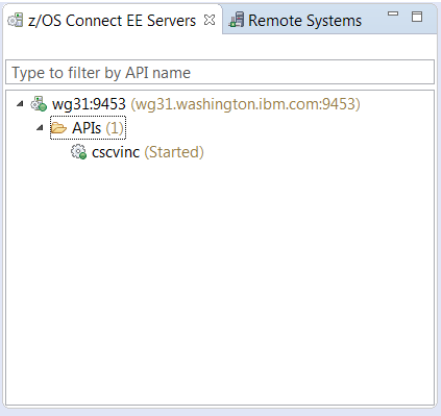
___3. Finally click on the *swagger* URL and you should see the Swagger document associated with this API.



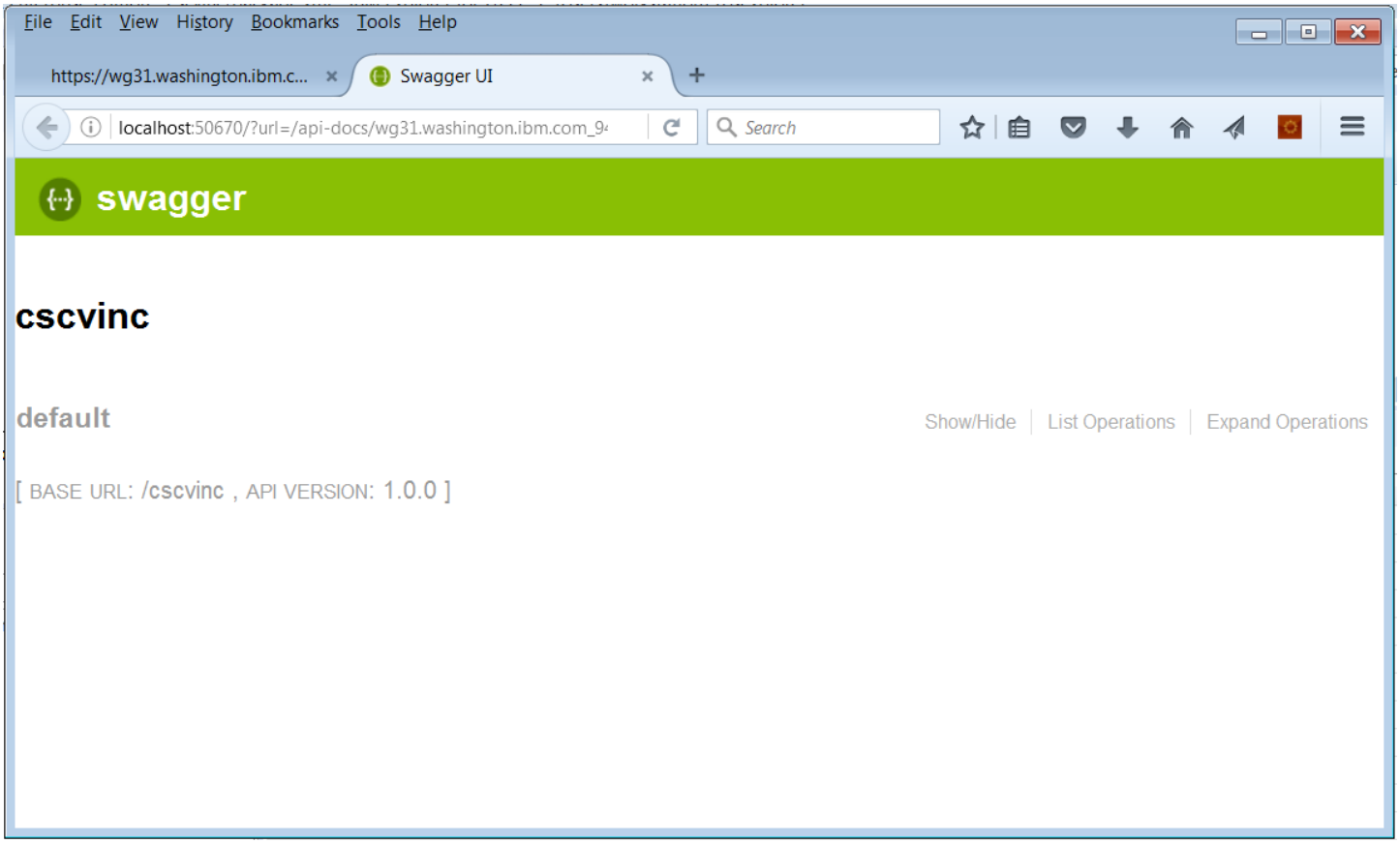
Mitch Johnson (mitchj@us.ibm.com)

___4. Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This document can be used by a developer or other tooling to develop REST clients for this specific API.

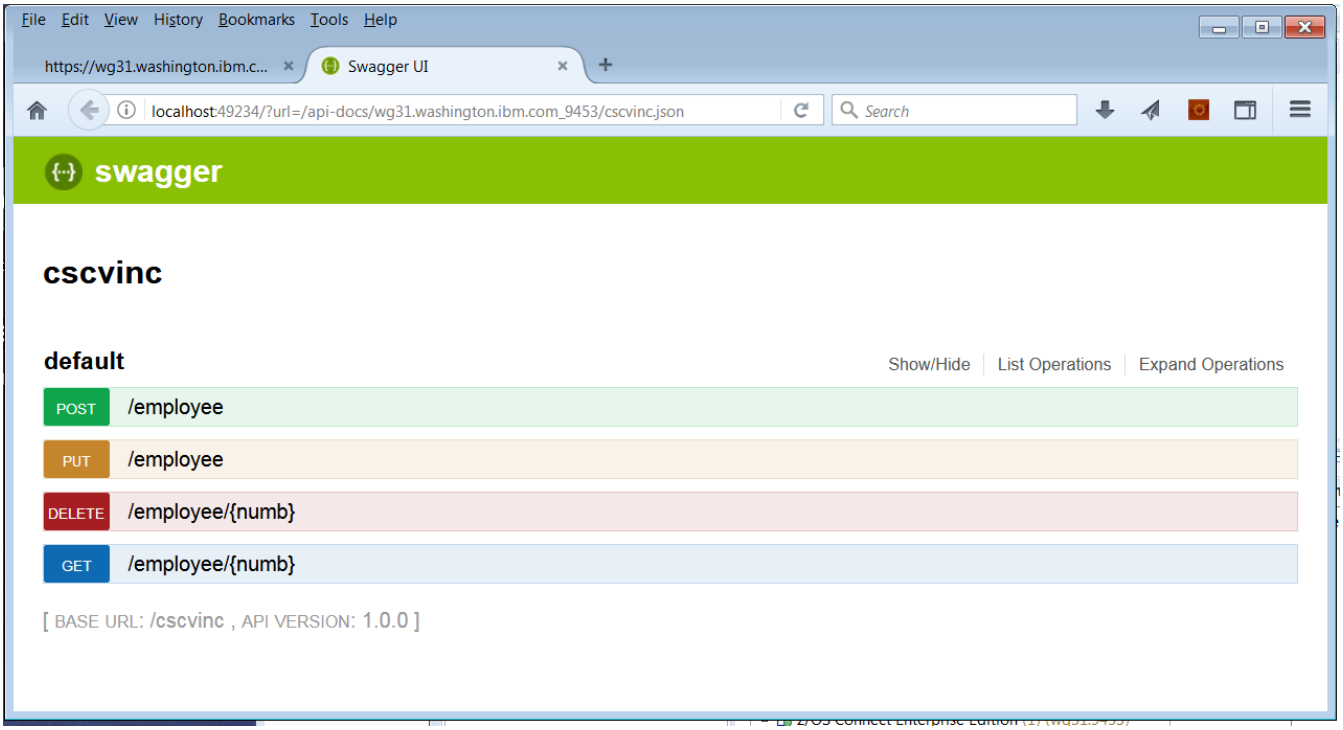
___5. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect EE Servers*. Expand *wg31:9453* and the expand the *APIs* folder. You should see a list of the APIs installed in the server.



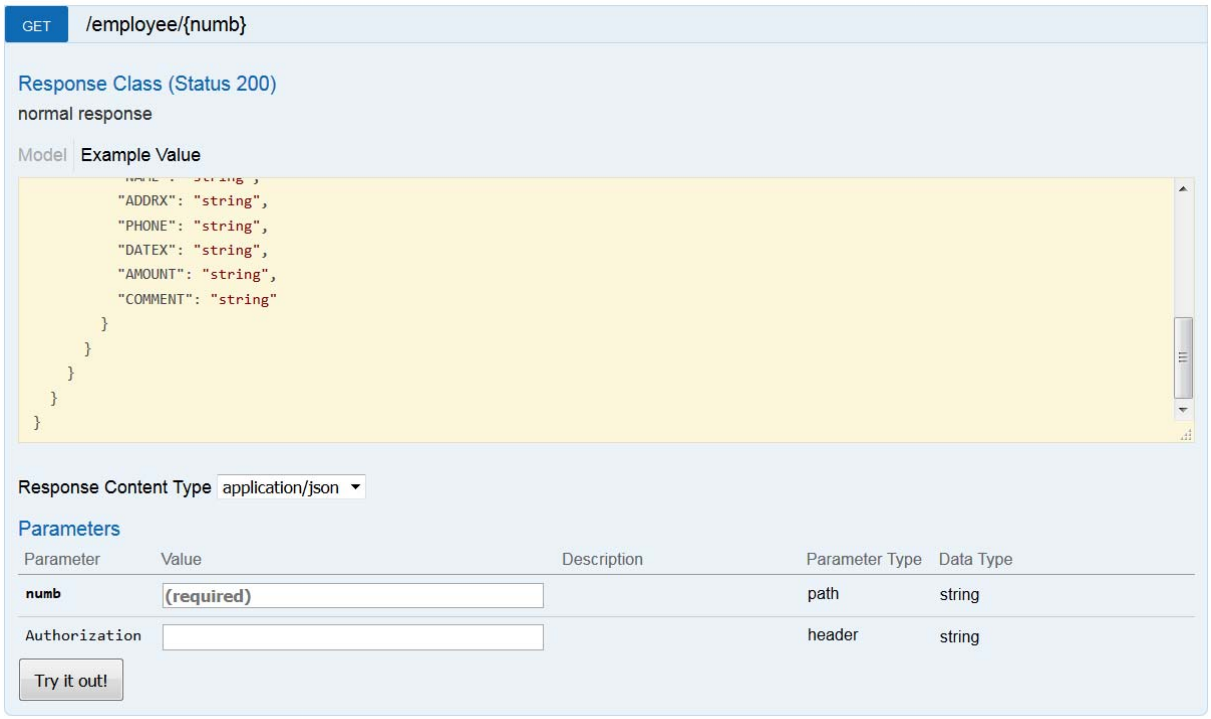
___6. Right mouse button click on *cscvinc* and select *Open in Swagger UI*. Click OK if an informational prompt appears. This will open a Firefox window showing a *Swagger* test client (see below).



___7. Click the *List Operations* and the browser should show a list of the available HTTP methods like this:



___8. Expand the GET operation by clicking on *Get* box and scroll down until the method *Parameters* are displayed as shown below:

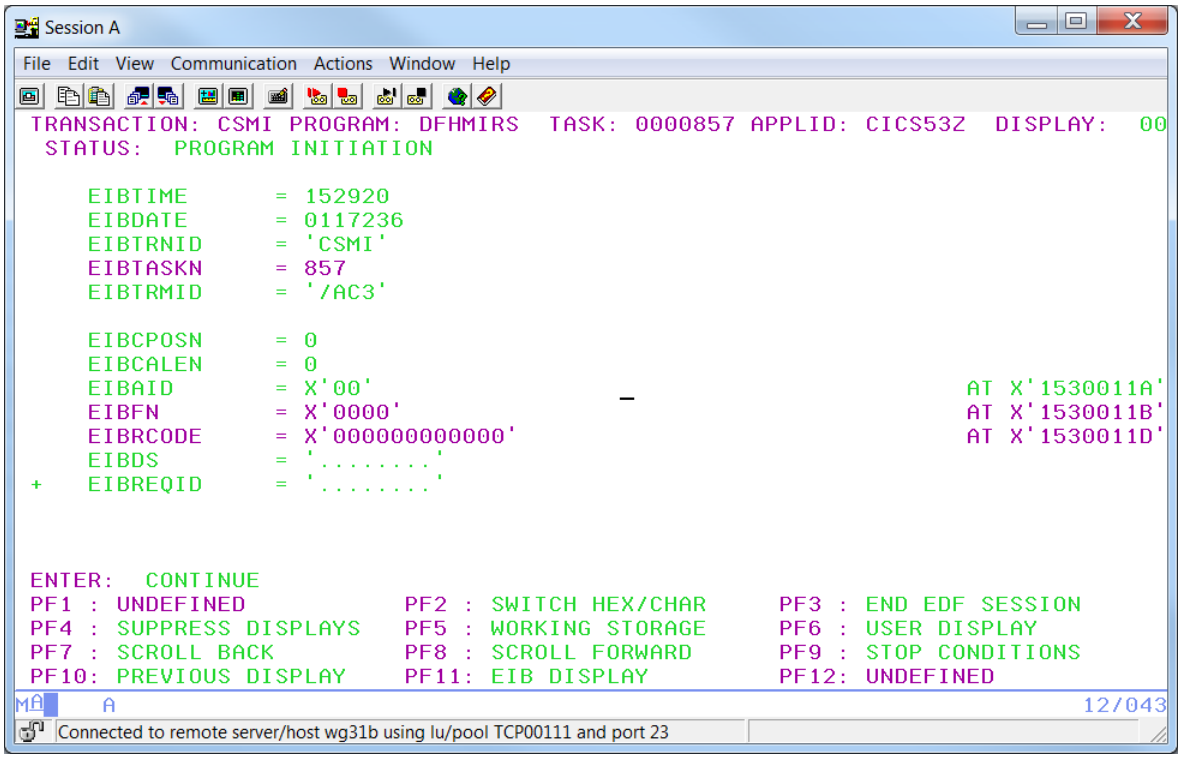


___9. Enter ***000100*** in the area beside numb and click the **Try it Out!** button. Scroll down and you see the response.

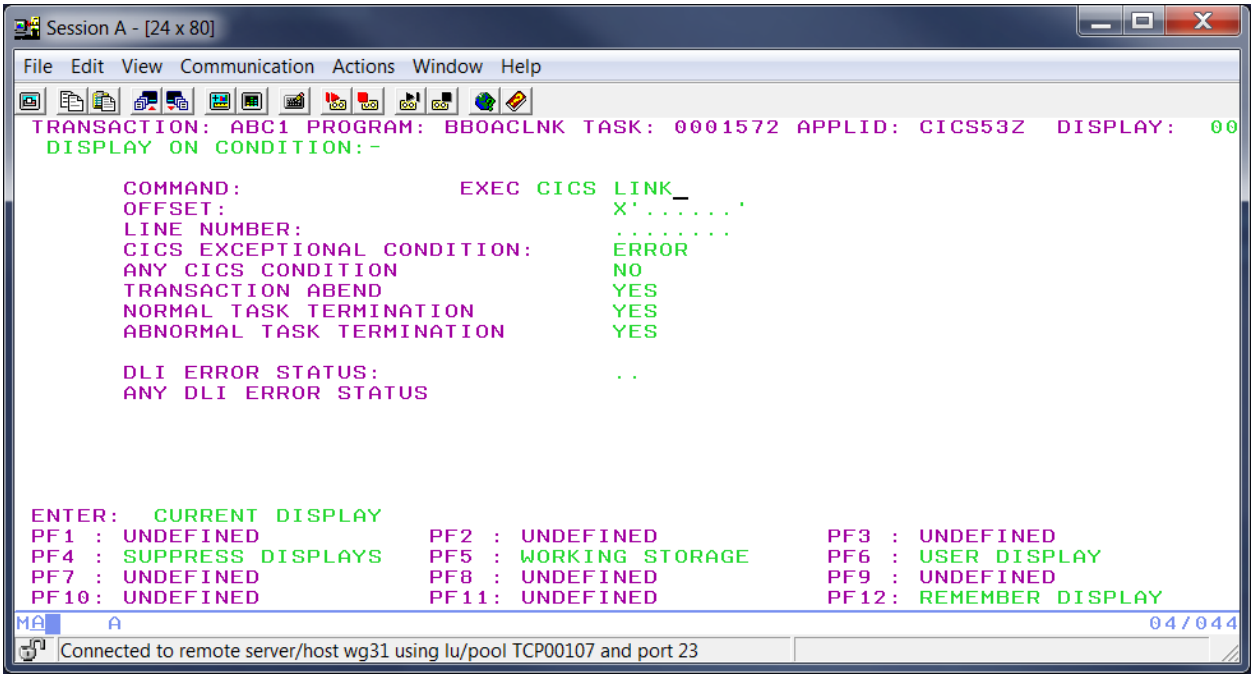
Mitch Johnson (mitchj@us.ibm.com)

___10. In a new or existing 3270 session start a session with CICS, clear the screen and enter CICS transaction ***CEDX CSMI*** to start a CICS execution diagnostic facility (EDF) on transaction CSMI.

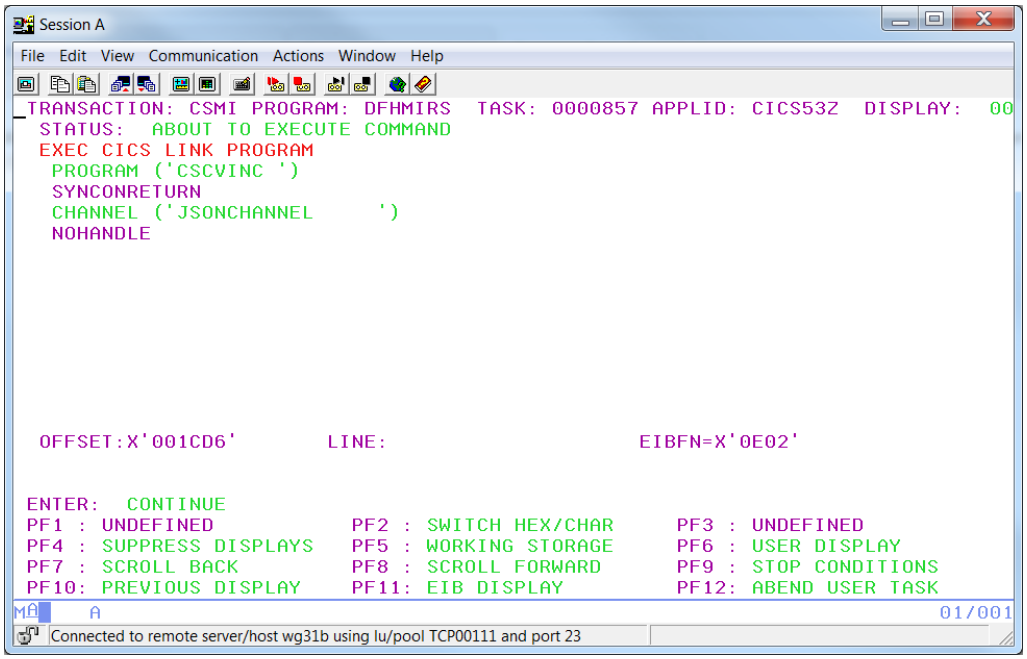
___11. Send another GET API request and you should see the EDF session start in the 3270 session.



___12. Use the **F9** key to set a stop condition. Set a stop on ***EXEC CICS LINK*** commands.

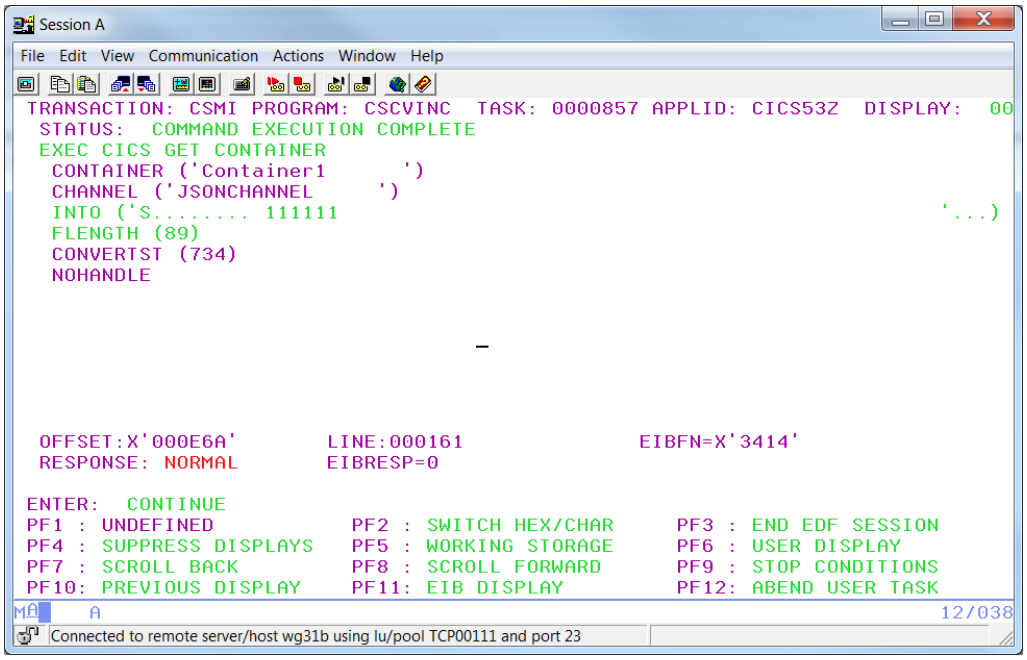


___13. Use the F4 key to suppress the intervening EDF displays. Eventually EDF will stop on an EXEC CICS LINK request.



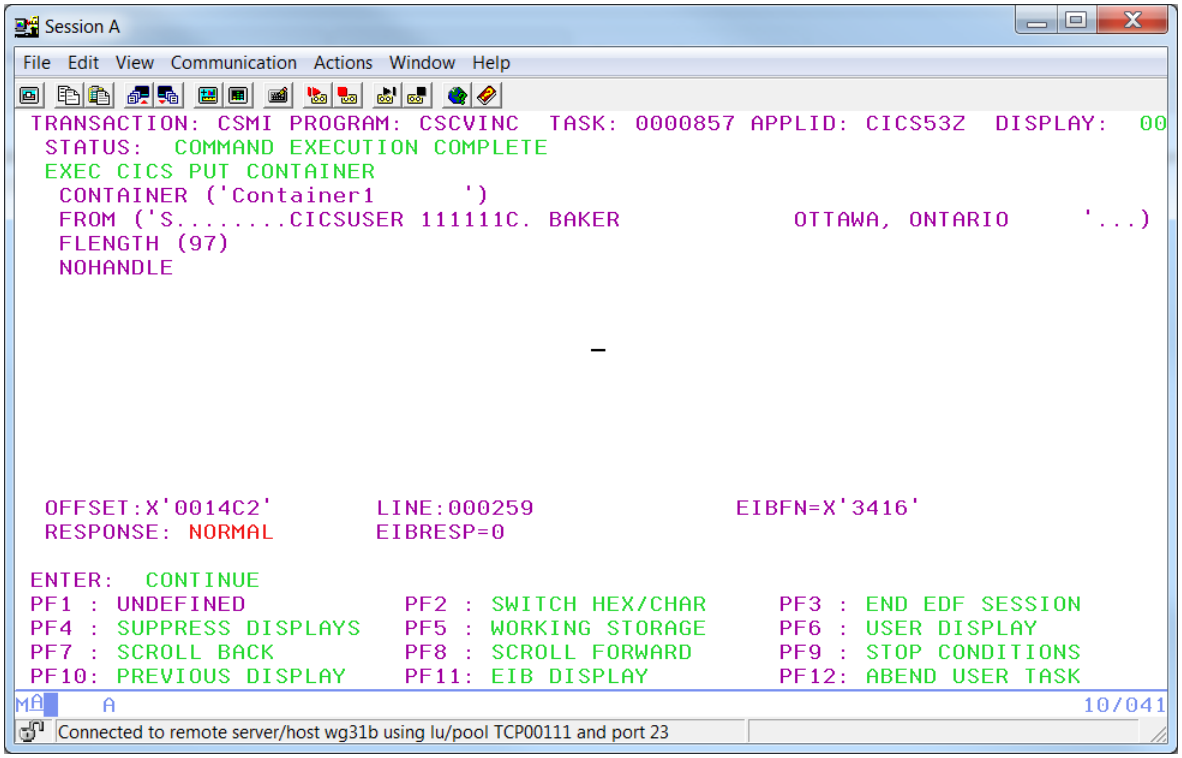
Note that the channel name is not the one specified in the LS2JSCSC input. z/OS Connect will override that channel name and use its own name for the channel and container.

___14. Keep pressing **ENTER** and eventually you will see the target program receive a NORMAL response to an *EXEC CICS GET CONTAINER* request.



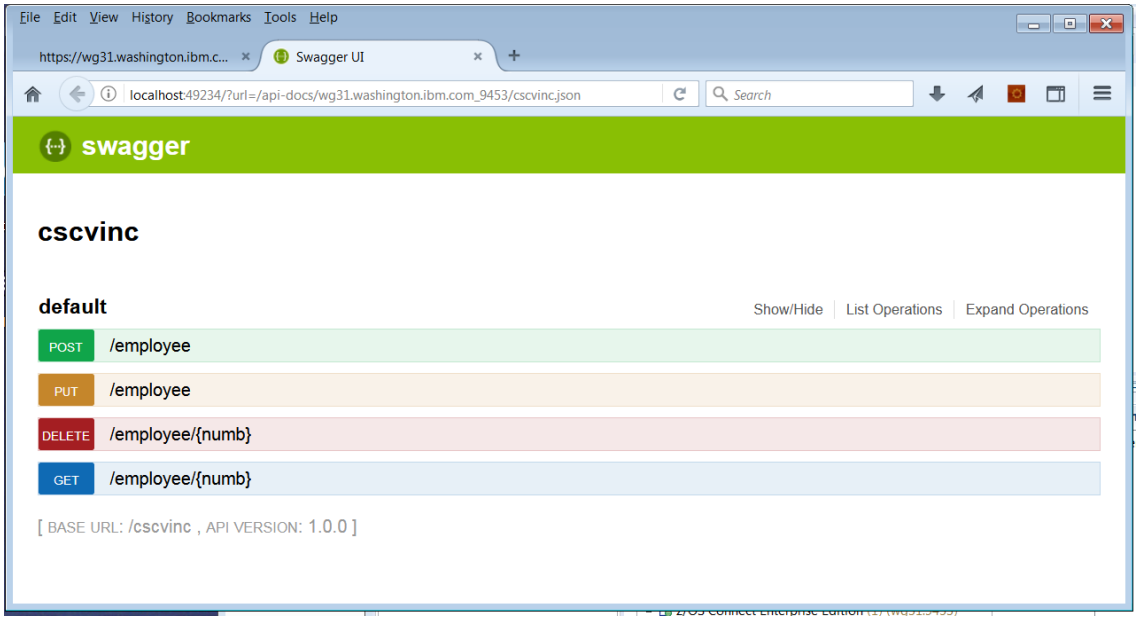
Mitch Johnson (mitchj@us.ibm.com)

___15. Keep pressing ENTER and the EXEC CICS APIs in the target program will be traced in EDF. Eventually there will be EXEC CICS PUT CONTAINER execution which places the results container into the channel for return back to z/OS Connect for conversion to a JSON message.

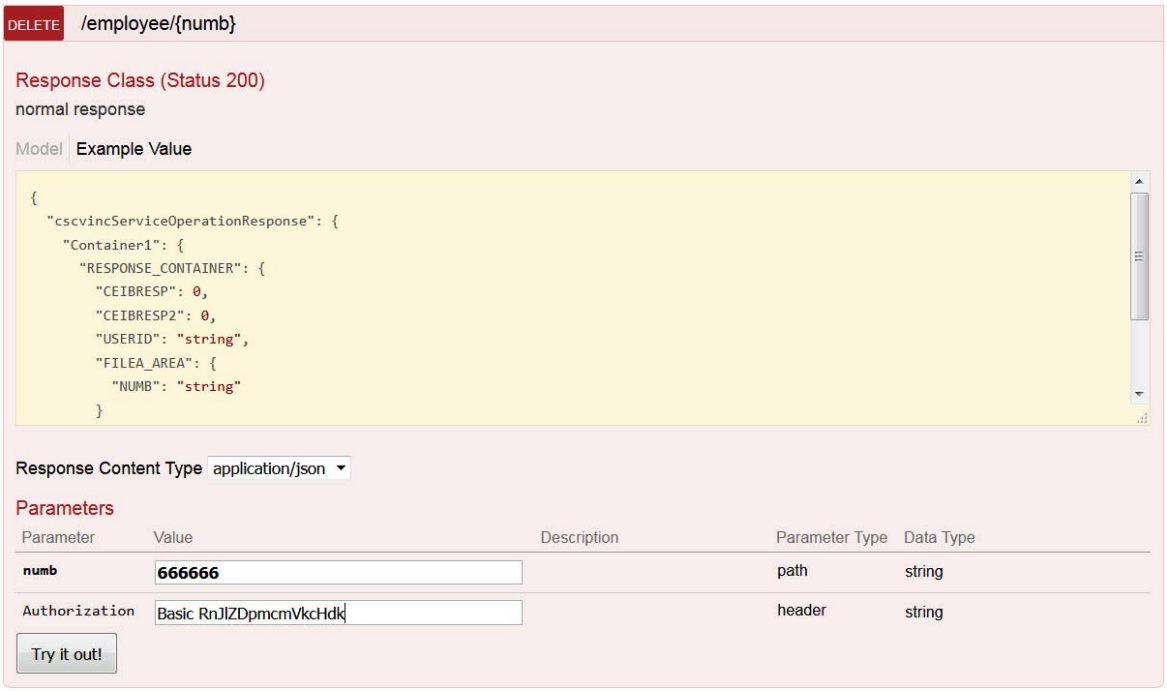


___16. Use the **F3** key to terminate the EDF tracing.

___17. Back in the Swagger-UI browser session scroll up to the list of operations and select the *Delete* operation.



___18. Scroll down to display the details of the DELETE parameters. Enter ***666666*** in the area beside *numb* and ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization*.



___19. Press the **Try it out!** Button and you see the results in the response body. (Both CEIBRESP and CEIBRESP2 set to zero indicates the CICS program was able to delete the record successfully).

___20. Try using the Swagger-UI to test the other operations (PUT and POST) with the other records

1. stat	numb	name	addrx	Phone	datex	amount	comment
Y	000100	S. D. BORMAN	SURREY, ENGLAND	32156778	26 11 81	$0100.11	*********
Y	000102	J. T. CZAYKOWSI	WARWICK, ENGLAND	98356183	26 11 81	$1111.11	*********
Y	000104	M. B. DOMBEY	LONDON,ENGLAND	12846293	26 11 81	$0999.99	*********
Y	000106	A. I. HICKSON	CROYDON, ENGLAND	19485673	26 11 81	$0087.71	*********
Y	000111	ALAN TULIP	SARATOGA,CALIFORNIA	46120753	01 02 74	$0111.11	*********
Y	000762	SUSAN MALAIKA	SAN JOSE,CALIFORNIA	22312121	01 06 74	$0000.00	*********
Y	000983	J. S. TILLING	WASHINGTON, DC	34512120	21 04 75	$9999.99	*********
Y	001222	D.J.VOWLES	BOBLINGEN, GERMANY	70315551	10 04 73	$3349.99	*********
Y	001781	TINA J YOUNG	SINDELFINGEN,GERMANY	70319990	21 06 77	$0009.99	*********
Y	003210	B.A. WALKER	NICE, FRANCE	12345670	26 11 81	$3349.99	*********
N	003214	PHIL CONWAY	SUNNYVALE, CAL.	34112120	00 06 73	$0009.99	*********
N	003890	BRIAN HARDER	NICE FRANCE	00000000	28 05 74	$0009.99	*********
N	004004	JANET FOUCHE	DUBLIN, IRELAND	71112121	02 11 73	$1259.99	*********
N	004445	DR. P. JOHNSON	SOUTH BEND, S.DAK.	61212120	26 11 81	$0009.99	*********
N	004878	ADRIAN JONES	SUNNYVALE, CALIF.	32212120	10 06 73	$5399.99	*********
N	005005	A. E. DALTON	SAN FRANCISCO, CA.	00000001	01 08 73	$0009.99	*********
N	005444	ROS READER	SARATOGA, CALIF.	67712120	20 10 74	$0809.99	*********
N	005581	PETE ROBBINS	BOSTON, MASS.	41312120	11 04 74	$0259.99	*********
Y	006016	SIR MICHAEL ROBERTS	NEW DELHI, INDIA	70331211	21 05 74	$0009.88	*********
N	006670	IAN HALL	NEW YORK, N.Y.	21212120	31 01 75	$3509.88	*********
Y	006968	J.A.L. STAINFORTH	WARWICK, ENGLAND	56713821	26 11 81	$0009.88	*********
N	007007	ANDREW WHARMBY	STUTTGART, GERMANY	70311000	10 10 75	$5009.88	*********
N	007248	M. J. AYRES	REDWOOD CITY, CALF.	33312121	11 10 75	$0009.88	*********
Y	007779	MRS. A. STEWART	SAN JOSE, CALIF.	41512120	03 01 75	$0009.88	*********
Y	009000	P. E. HAVERCAN	WATERLOO, ONTARIO	09876543	21 01 75	$9000.00	*********
Y	100000	M. ADAMS	TORONTO, ONTARIO	03415121	26 11 81	$0010.00	*********
Y	111111	C. BAKER	OTTAWA, ONTARIO	51212003	26 11 81	$0011.00	*********
Y	200000	S. P. RUSSELL	GLASGOW, SCOTLAND	63738290	26 11 81	$0020.00	*********
Y	222222	DR E. GRIFFITHS	FRANKFURT, GERMANY	20034151	26 11 81	$0022.00	*********
Y	300000	V. J. HARRIS	NEW YORK, U.S.	64739801	26 11 81	$0030.00	*********
Y	333333	J.D. HENRY	CARDIFF, WALES	78493020	26 11 81	$0033.00	*********
Y	400000	C. HUNT	MILAN, ITALY	25363738	26 11 81	$0040.00	*********
Y	444444	D. JACOBS	CALGARY, ALBERTA	77889820	26 11 81	$0044.00	*********
Y	500000	P. KINGSLEY	MADRID, SPAIN	44454640	26 11 81	$0000.00	*********
Y	555555	S.J. LAZENBY	KINGSTON, N.Y.	39944420	26 11 81	$0005.00	*********
Y	600000	M.F. MASON	DUBLIN, IRELAND	12398780	26 11 81	$0010.00	*********
Y	666666	R. F. WALLER	LA HULPE, BRUSSELS	42983840	26 11 81	$0016.00	*********
Y	700000	M. BRANDON	DALLAS, TEXAS	57984320	26 11 81	$0002.00	*********
Y	777777	L.A. FARMER	WILLIAMSBURG, VIRG.	91876131	26 11 81	$0027.00	*********
Y	800000	P. LUPTON	WESTEND, LONDON	24233389	26 11 81	$0030.00	*********
Y	888888	P. MUNDY	NORTHAMPTON, ENG.	23691639	26 11 81	$0038.00	*********
Y	900000	D.S. RENSHAW	TAMPA, FLA.	35668120	26 11 81	$0040.00	*********
Y	999999	ANJI STEVENS	RALEIGH, N.Y.	84591639	26 11 81	$0049.00	*********

Summary

You have verified the API. The service layer is what does the data conversion and mapping and the IPIC connection to the backend program. The API layer provides a further level of abstraction and allows a more flexible use of HTTP verbs, and better mapping of data via the API editor function.

z/OS Connect EE APIs and a CICS COMMAREA program

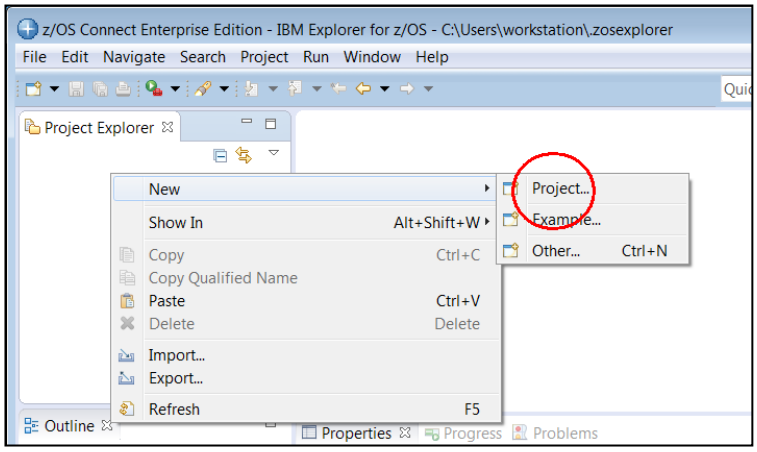
Create the services

The CICS application that will be invoked by this service is CICS sample application DFH0XCMN. This program is provided with the CICS office supply sample application. DFH0XCMN is passed a COMMAREA which contains a request field. This field is used by DFH0XCMN to determine the next which of 3 functions of the sample application will be accessed. The three functions are (1) inquire on a single item in the catalog (INQS), (2) provide a list of items in the catalog starting with a specific item (INQC) or (3) place an order for an item (ORDR). The COMMAREA is based on a COPYBOOK which uses redefines the COMMAREA based on the type of request.

The next step is to create 3 services which correspond to these three functions provided by the sample program DFH0XCMN. These service names must match the service names specified in the server.xml file (see catalog.xml on page 71).

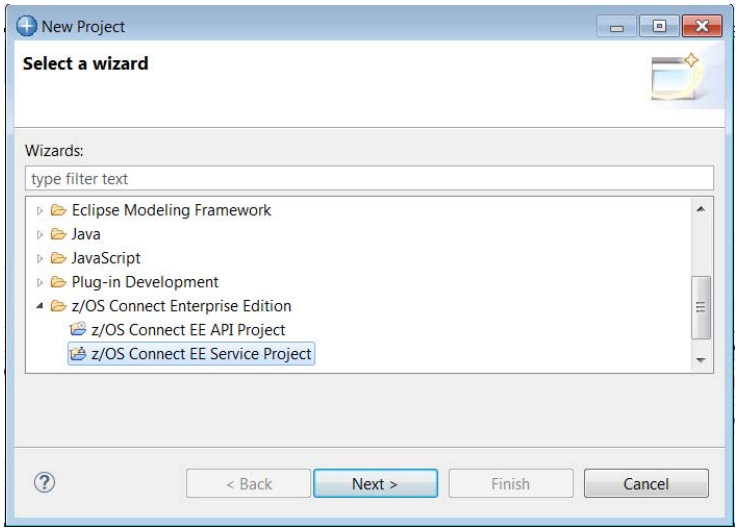
___1. In the z/OS Explorer session switch the perspective back to the *z/OS Connect Enterprise Edition* perspective.

___2. In the upper left, position your mouse anywhere in the *Project Explorer* view and right-mouse click, then select *New* →*Project*:

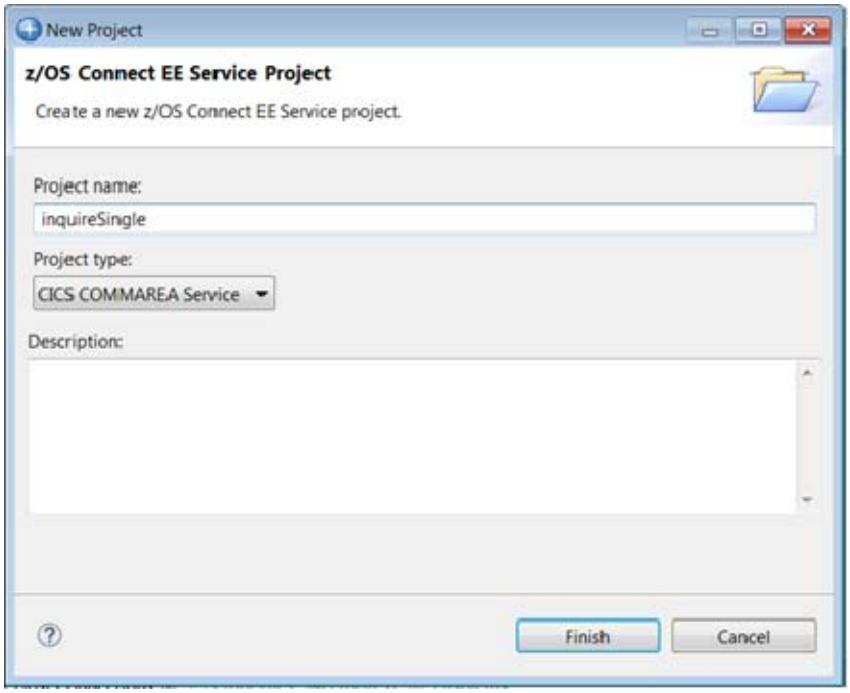


Mitch Johnson (mitchj@us.ibm.com)

___3. In the *New Project* window, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE Service Project* and then click the **Next** button.

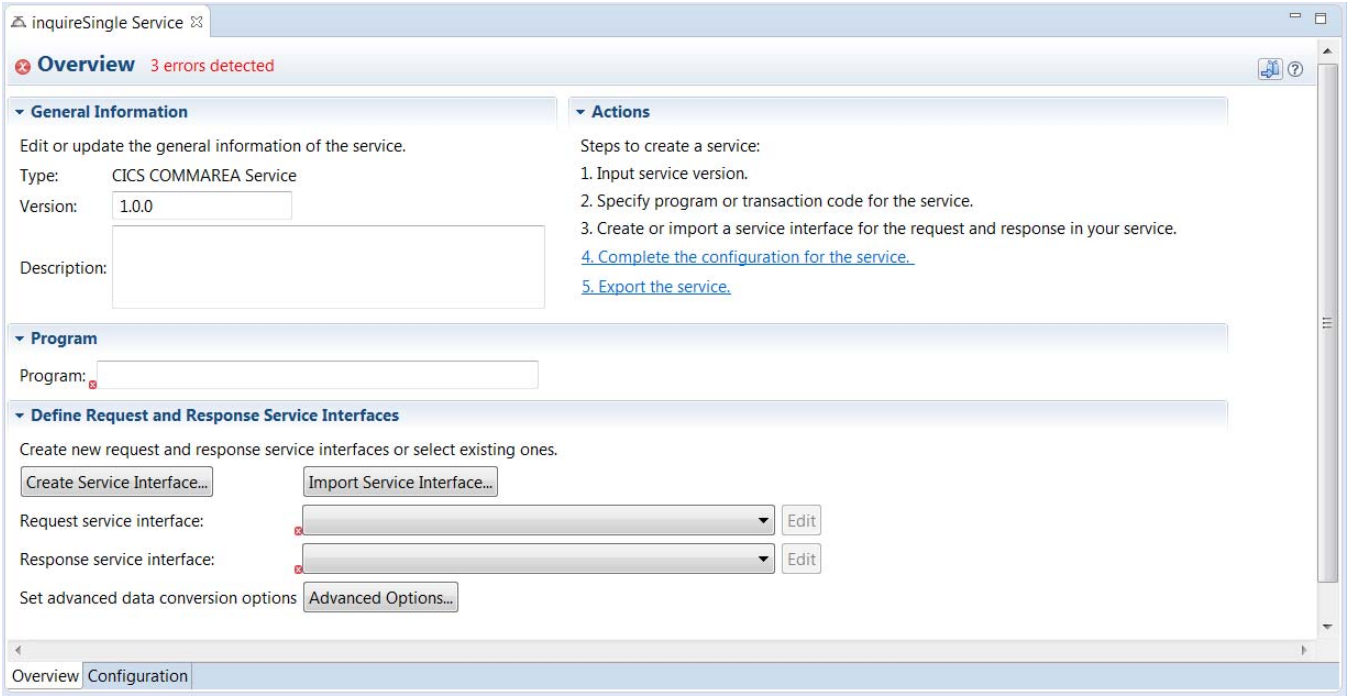


___3. On the new *New Project* window enter ***inquireSingle*** as the *Project name* and use the pull-down arrow to select *CICS COMMAREA Service* as the *Project type*. Click **Finish** to continue



Mitch Johnson (mitchj@us.ibm.com)

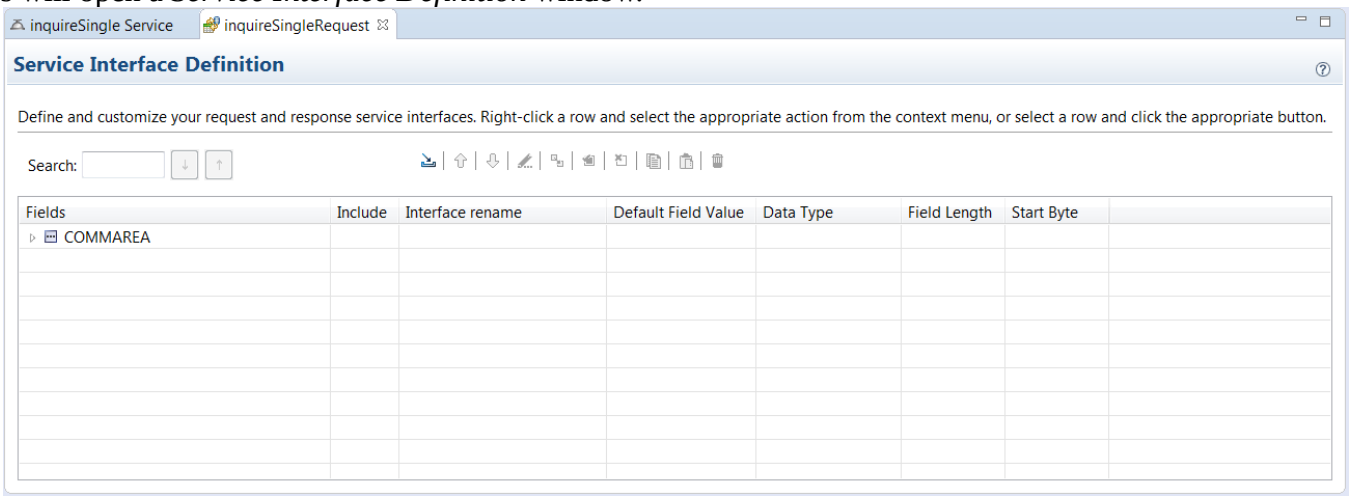
___4. This will open the *Overview* window for the *inquireSingle Service*. For now disregard the message about the 3 errors detected, they will be addressed shortly.



___5. Next enter the target CICS program name ***DFH0XCMN*** in the area beside *Program* (case is important).

___6. Click the **Create Service Interface** button to create the first service required by this API and enter a *Service interface name* of ***inquireSingleRequest***. Click **OK** to continue.

___7. This will open a *Service Interface Definition* window.

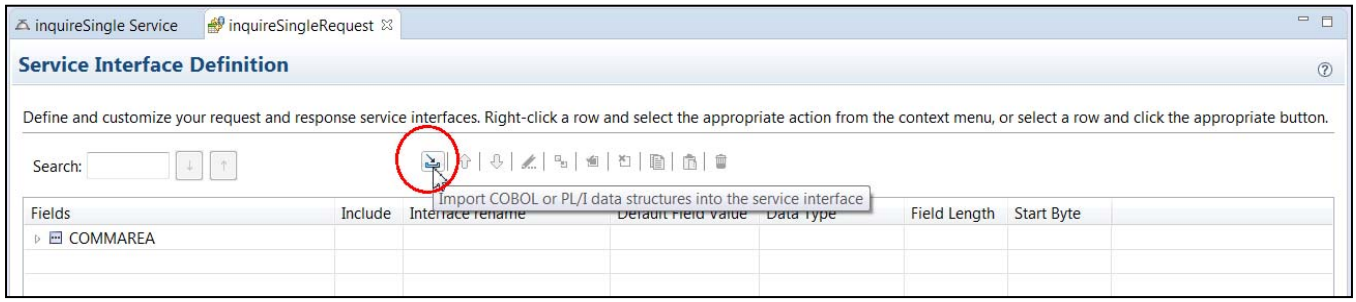


Mitch Johnson (mitchj@us.ibm.com)

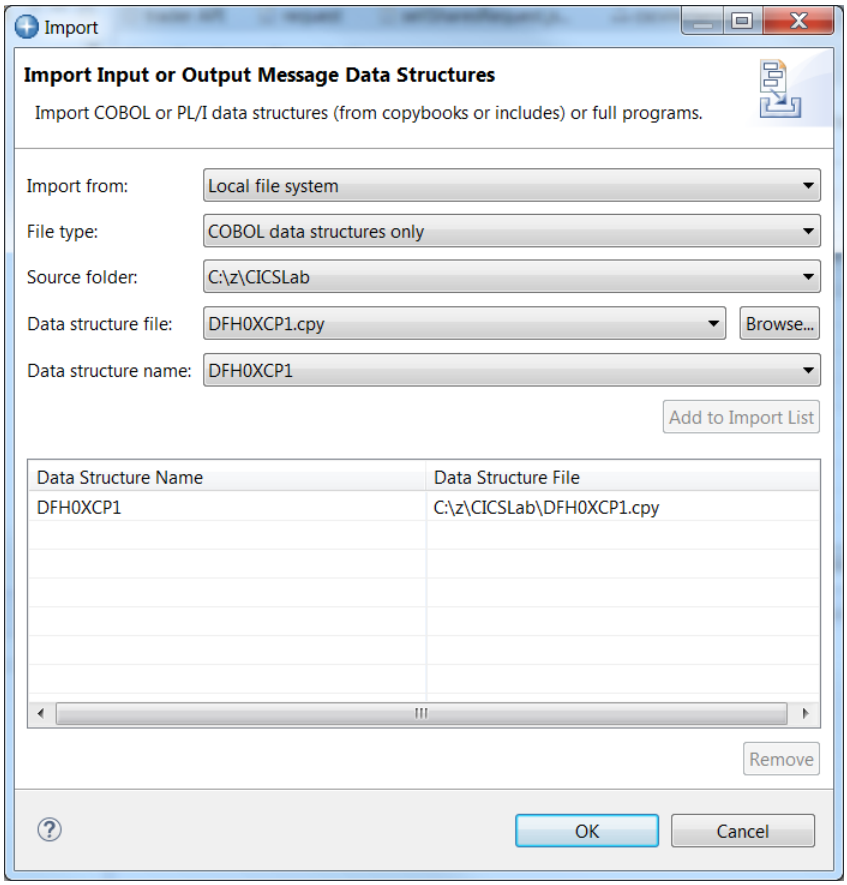
___8. The first step is to import the COBOL copy book that represents the COMMAREA when inquiring on a single item in the Catalog application. This copy book was downloaded from member DFH0XCP1 in the CICS SDFHSAMP target data set.

```
      *    Catalogue COMMAREA structure
           03 CA-REQUEST-ID            PIC X(6).
           03 CA-RETURN-CODE           PIC 9(2).
           03 CA-RESPONSE-MESSAGE      PIC X(79).
           03 CA-REQUEST-SPECIFIC      PIC X(911).
      *    Fields used in Inquire Catalog
           03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
               05 CA-LIST-START-REF        PIC 9(4).
               05 CA-LAST-ITEM-REF         PIC 9(4).
               05 CA-ITEM-COUNT            PIC 9(3).
               05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
               05 CA-CAT-ITEM  REDEFINES CA-INQUIRY-RESPONSE-DATA
                              OCCURS 15 TIMES.
                   07 CA-ITEM-REF          PIC 9(4).
                   07 CA-DESCRIPTION       PIC X(40).
                   07 CA-DEPARTMENT        PIC 9(3).
                   07 CA-COST              PIC X(6).
                   07 IN-STOCK             PIC 9(4).
                   07 ON-ORDER             PIC 9(3).
      *    Fields used in Inquire Single
           03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
               05 CA-ITEM-REF-REQ          PIC 9(4).
               05 FILLER                   PIC 9(4).
               05 FILLER                   PIC 9(3).
               05 CA-SINGLE-ITEM.
                   07 CA-SNGL-ITEM-REF     PIC 9(4).
                   07 CA-SNGL-DESCRIPTION  PIC X(40).
                   07 CA-SNGL-DEPARTMENT   PIC 9(3).
                   07 CA-SNGL-COST         PIC X(6).
                   07 IN-SNGL-STOCK        PIC 9(4).
                   07 ON-SNGL-ORDER        PIC 9(3).
               05 FILLER                   PIC X(840).
      *    Fields used in Place Order
           03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
               05 CA-USERID                PIC X(8).
               05 CA-CHARGE-DEPT           PIC X(8).
               05 CA-ITEM-REF-NUMBER       PIC 9(4).
               05 CA-QUANTITY-REQ          PIC 9(3).
               05 FILLER                   PIC X(888).
```

___9. On the *Service Interface Definition* window there is a tool bar near the top. If you hover over an icon its function will be display as below. Click the *Import COBOL or PL/I data structure into the service interface* icon to start the import process.



___10. This will open the *Import* window. On this window select *Local file system* as source of the import and *COBOL data structure only* as the *File type*. Press the **Browse** button and **Open** directory *C:\z\CICSLAB* and then select file *DFH0XCP1*.cpy and click **Open** to import this file into this project. Click the **Add to Import List** button and the click **OK** to continue.

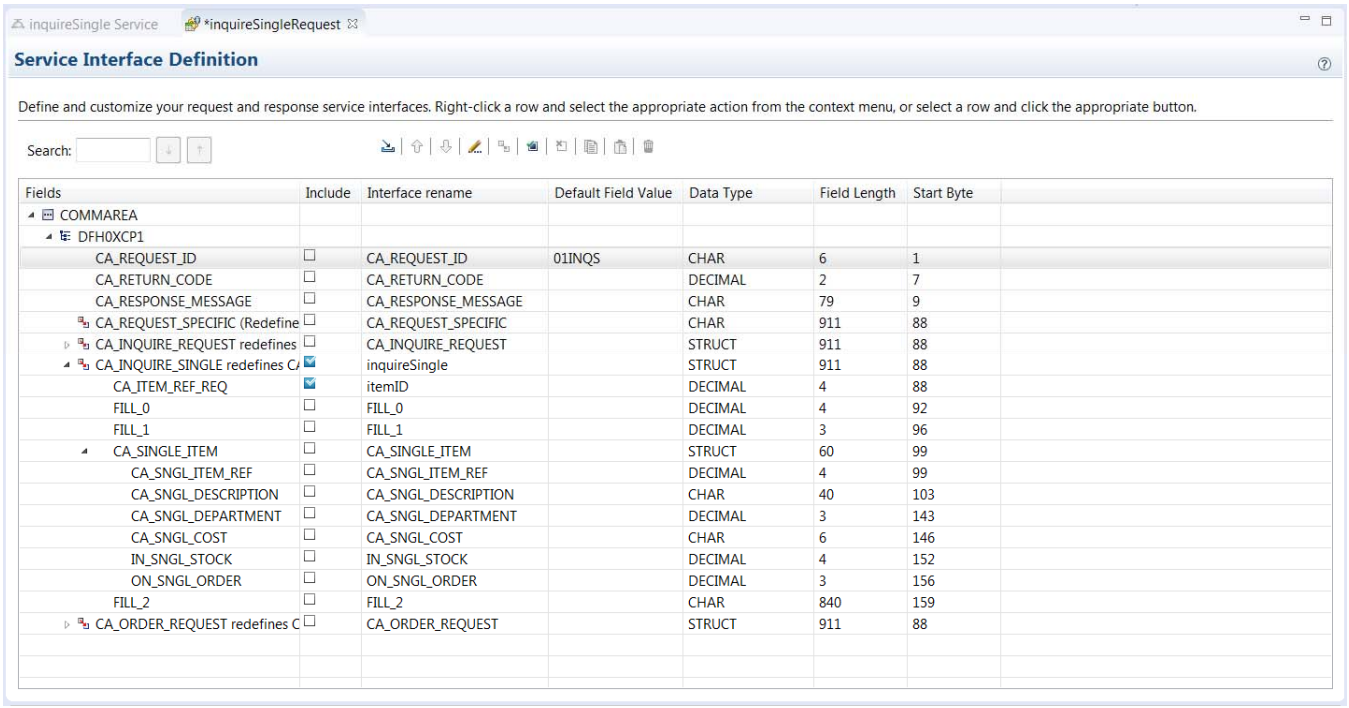


___11. When you expand *COMMAREA* you will see the COBOL 'variables' that have been imported into the service project. In this service we want to return a single item and the only fields that need to be exposed in the request message is the item number.

a. First uncheck the boxes under *Include* column so that only *CA_INQUIRE_SINGLE* and *CA_ITEM_REF_REQ* are checked. These are the fields that will be exposed to the requestor of this API.

b. Double click an entry in the *Interface rename* column to open it for editing. Rename interface fields *CA_INQUIRE_SINGLE* to ***inquireSingle*** and ***CA_ITEM_REF_REQ*** to ***itemID***. Change interface field *CA_REQUEST_ID* by setting its default value to ***01INQS***.

When finished the *Service Interface Definition* should look like the below:

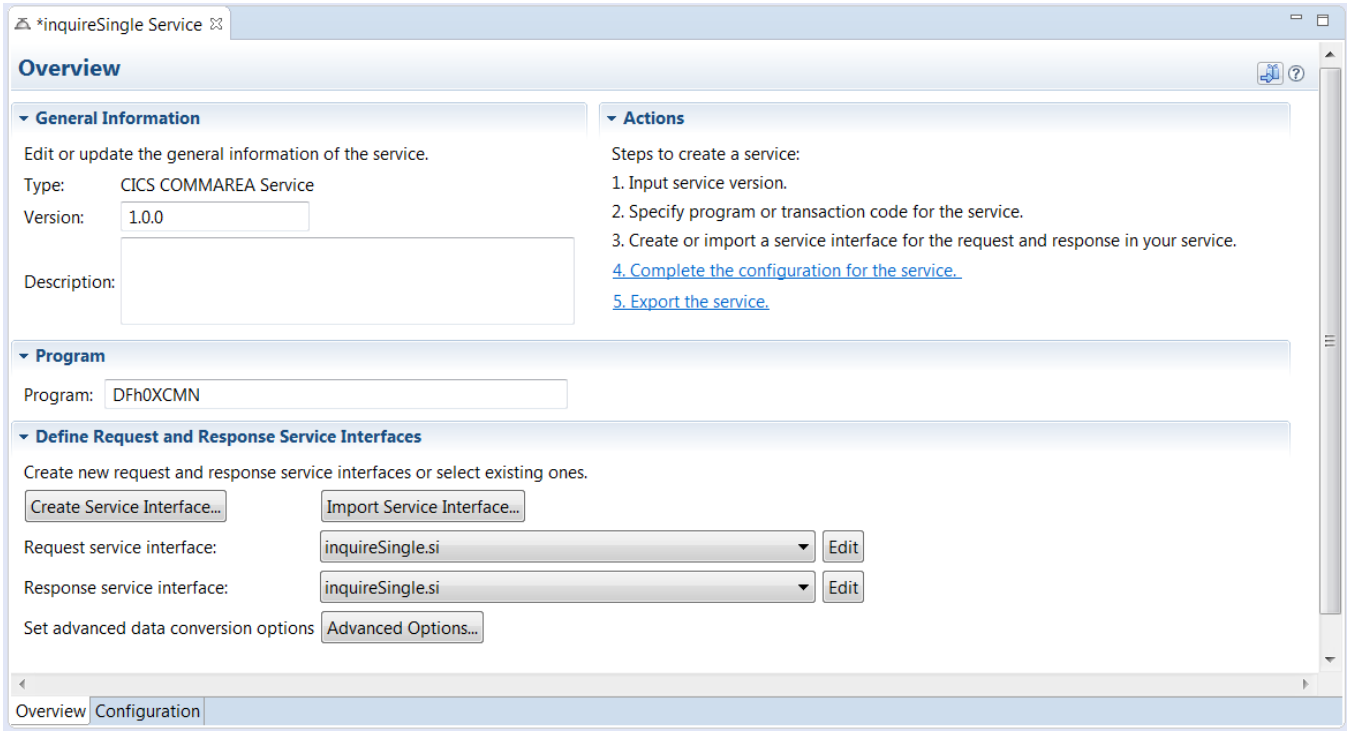


Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID	01INQS	CHAR	6	1
CA_RETURN_CODE	☐	CA_RETURN_CODE		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☐	CA_RESPONSE_MESSAGE		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefine	☐	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines	☐	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines C	☑	inquireSingle		STRUCT	911	88
CA_ITEM_REF_REQ	☑	itemID		DECIMAL	4	88
FILL_0	☐	FILL_0		DECIMAL	4	92
FILL_1	☐	FILL_1		DECIMAL	3	96
CA_SINGLE_ITEM	☐	CA_SINGLE_ITEM		STRUCT	60	99
CA_SNGL_ITEM_REF	☐	CA_SNGL_ITEM_REF		DECIMAL	4	99
CA_SNGL_DESCRIPTION	☐	CA_SNGL_DESCRIPTION		CHAR	40	103
CA_SNGL_DEPARTMENT	☐	CA_SNGL_DEPARTMENT		DECIMAL	3	143
CA_SNGL_COST	☐	CA_SNGL_COST		CHAR	6	146
IN_SNGL_STOCK	☐	IN_SNGL_STOCK		DECIMAL	4	152
ON_SNGL_ORDER	☐	ON_SNGL_ORDER		DECIMAL	3	156
FILL_2	☐	FILL_2		CHAR	840	159
CA_ORDER_REQUEST redefines C	☐	CA_ORDER_REQUEST		STRUCT	911	88

N.B. the original interface names were derived from the COBOL source shown earlier.

Mitch Johnson (mitchj@us.ibm.com)

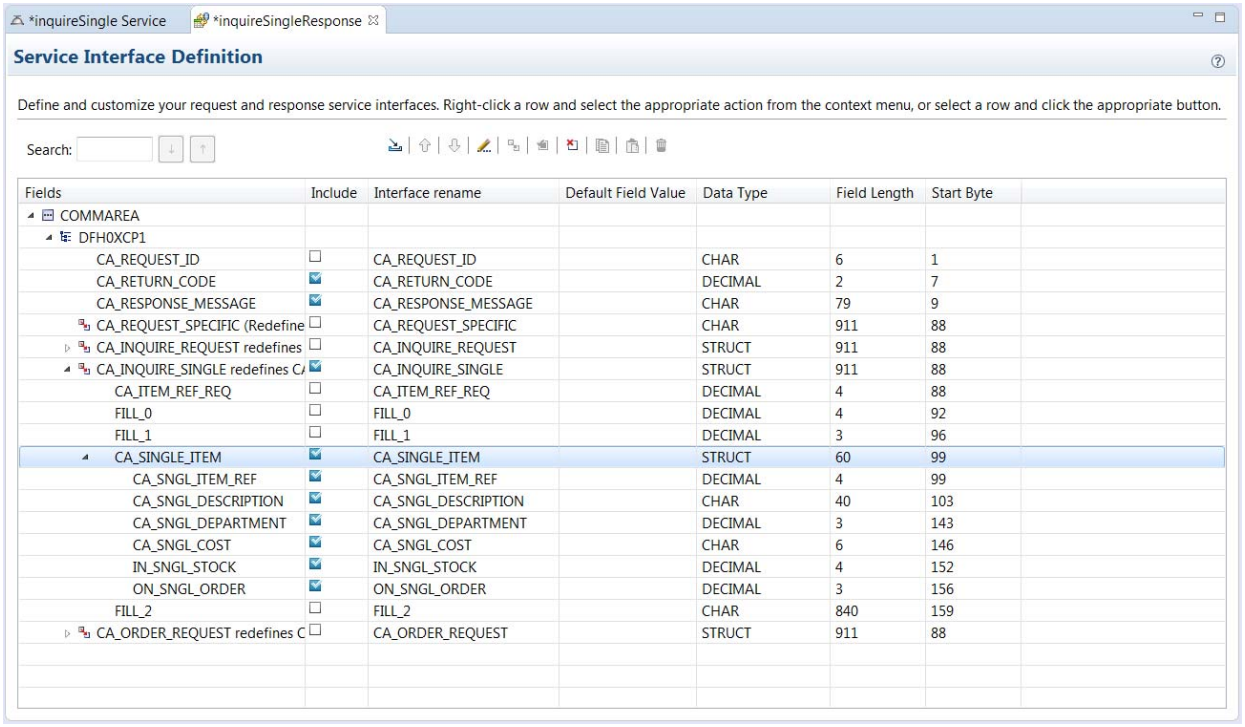
___12. Close the *Service Interface Definition* window by clicking on the white X in the tab being sure to save the changes. Note now that the *Request service interface* and the *Response service interface* areas have now been populated with *inquireSingleRequest.si*. Also note that you can use their respective **Edit** buttons to return to the *Service Interface Definition* window for each interface.



Tech-Tip: Note that even though the program uses a COMMAREA we can easily distinguish which fields are provided in a request and which fields will be in the response by providing different service interfaces. This is particularly useful when using containers.

___13. Repeat steps 6 through 10 to create a service interface for the *inquireSingle* service response message. Enter a service interface name of ***inquireSingleResponse*** rather than ***inquireSingleRequest***.

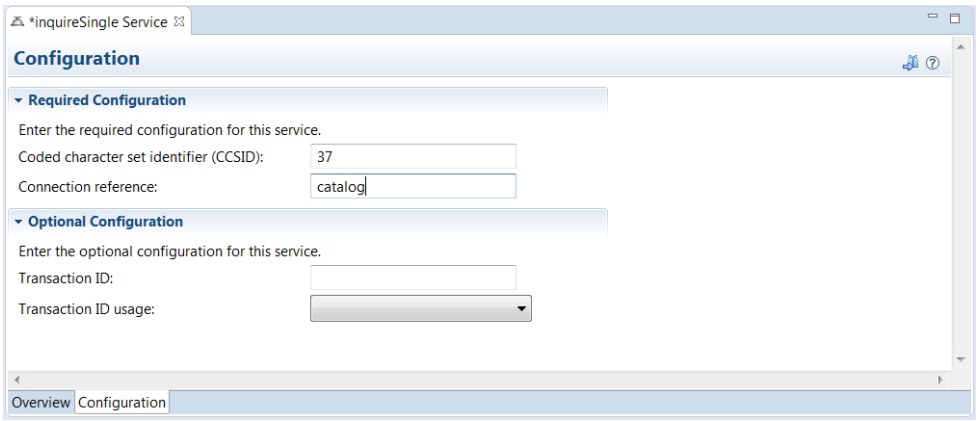
___14. A different set of fields will be returned in the response. Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE, CA_RESPONSE_MESSAGE, CA_INQUIRE_SINGLE* and *CA_SINGLE_ITEM* checked. These are fields that will be exposed to the requestor of this API in the response. You can rename these fields if you choose. If you do some of the later screen shots may be different.



___15. Close the *Service Interface Definition* window.

___16. Set the *Response service interface* to *inquireSingleResponse.si*.

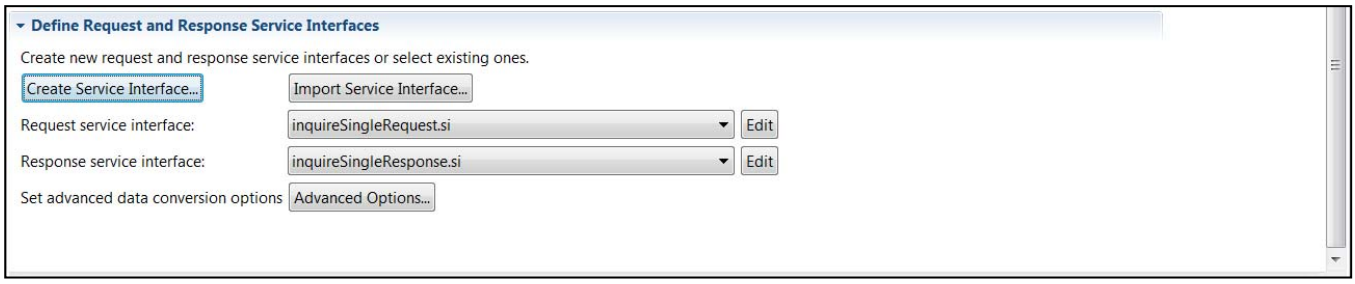
___17. Next, we need to identify a connection reference for which CICS region will be used. Click on the *Configuration* tab at the bottom of the *Overview* window to display the *Configuration* window. Enter ***catalog*** in the area beside *Connection reference*.



Tech-Tip: The *Connection reference* identifies the *cicsIpicConn**ection*** element to be used to access a CICS region in the z/OS Connect EE configuration**,** see catalog.xml on page 71.

Mitch Johnson (mitchj@us.ibm.com)

___19. Verify that the request and response service interfaces are set correctly. If not, set them as below.



___20. Save the *inquireSingle* service by closing all open tabs.

___21. Repeat steps 1 through 5 to create the *inquireCatalog* service.

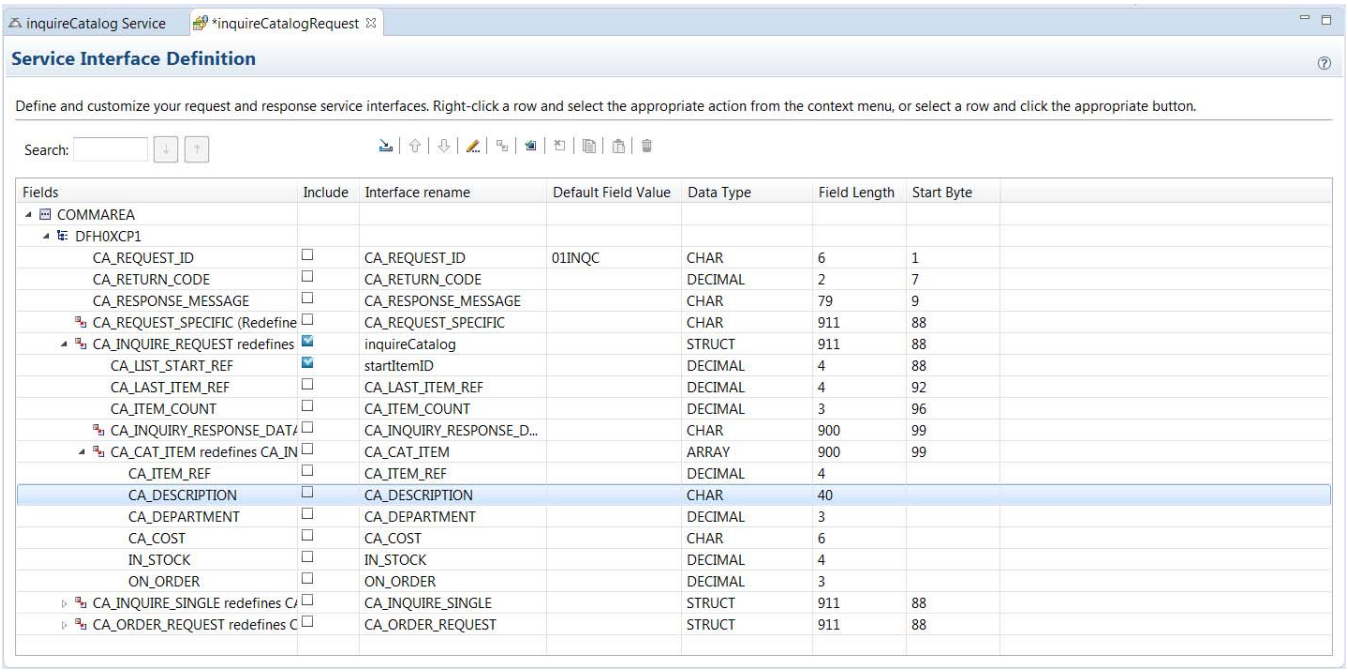
___22. Repeat steps 6 through 10 to create a service interface for ***inquireCatalog*** service request message**.** Enter a name of ***inquireCatalogRequest***.

___23. Expand the COMMAREA and perform the following steps for the *inquireCatalog* request:

a. First uncheck the boxes under *Include* column so that only *CA_INQUIRE_REQUEST* and *CA_LIST_START_REF* are checked. These are the fields that will be exposed to the requestor of this API.

b. Double click an entry in the *Interface rename* column to open it for editing. Rename interface fields *CA_INQUIRE_REQUEST* to ***inquireCatalog*** and ***CA_LIST_START_REF*** to ***startItemID***.

c. Change interface field *CA_REQUEST_ID* by setting its default value to ***01INQC***.

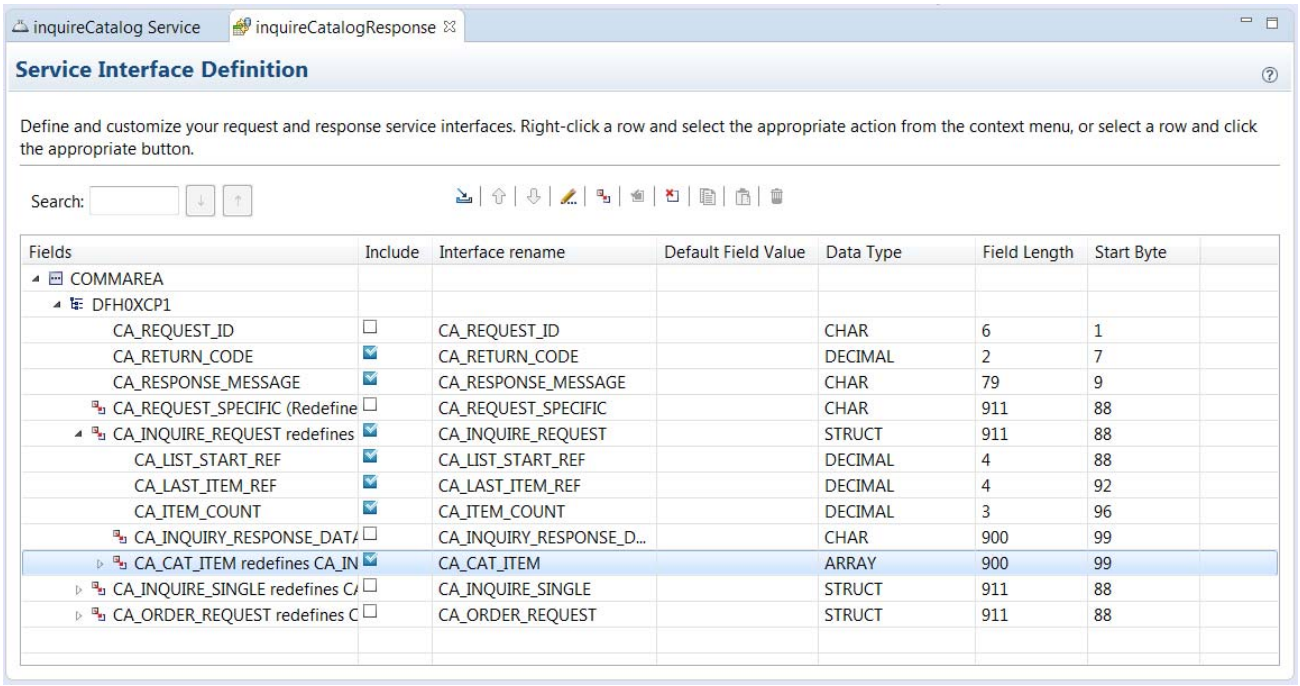


Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID	01INQC	CHAR	6	1
CA_RETURN_CODE	☐	CA_RETURN_CODE		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☐	CA_RESPONSE_MESSAGE		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefine	☐	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines	☑	inquireCatalog		STRUCT	911	88
CA_LIST_START_REF	☑	startItemID		DECIMAL	4	88
CA_LAST_ITEM_REF	☐	CA_LAST_ITEM_REF		DECIMAL	4	92
CA_ITEM_COUNT	☐	CA_ITEM_COUNT		DECIMAL	3	96
CA_INQUIRY_RESPONSE_DAT/	☐	CA_INQUIRY_RESPONSE_D...		CHAR	900	99
CA_CAT_ITEM redefines CA_IN	☐	CA_CAT_ITEM		ARRAY	900	99
CA_ITEM_REF	☐	CA_ITEM_REF		DECIMAL	4	
CA_DESCRIPTION	☐	CA_DESCRIPTION		CHAR	40	
CA_DEPARTMENT	☐	CA_DEPARTMENT		DECIMAL	3	
CA_COST	☐	CA_COST		CHAR	6	
IN_STOCK	☐	IN_STOCK		DECIMAL	4	
ON_ORDER	☐	ON_ORDER		DECIMAL	3	
CA_INQUIRE_SINGLE redefines C/	☐	CA_INQUIRE_SINGLE		STRUCT	911	88
CA_ORDER_REQUEST redefines C	☐	CA_ORDER_REQUEST		STRUCT	911	88

___24. Close the *Service Interface Definition* window.

___25. Repeat steps 6 through 10 to create a service interface for the *inquireCatalog* response message with the *service interface name* of ***inquireCatalogResponse***.

___26. A different set of fields will be returned in the response. Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE, CA_RESPONSE_MESSAGE, CA_INQUIRE_REQUEST, CA_LIST_START_REF, CA_LAST_ITEM_REF, CA_ITEM_COUNT* and *CA_CAT_ITEM* are checked. These are fields that will be exposed to the requestor of this API in the response. Rename these fields if you choose.

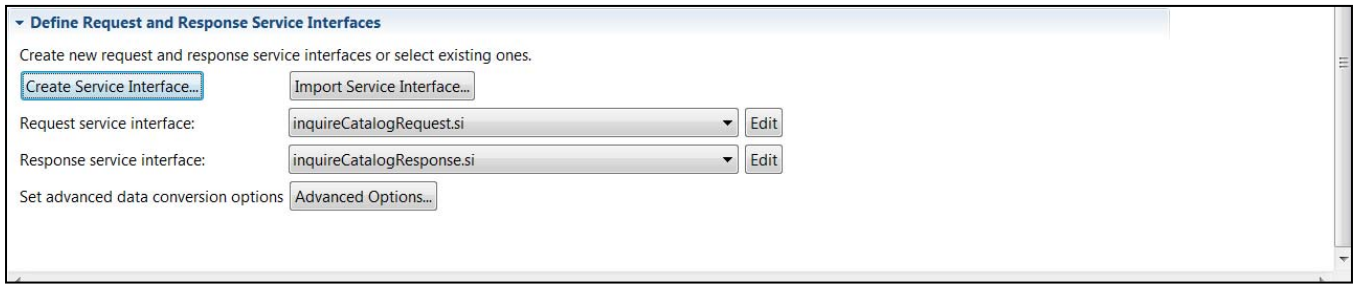


___27. Close the *Service Interface Definition* window.

___28. Set the *Response service interface* to *inquireCatalogResponse.si*.

___29. Identify the connection reference for which CICS region will be used for this service. Click on the Configuration tab at the bottom of the *Overview* window to display the *Configuration* window. Enter ***catalog*** in the area beside *Connection reference*.

___30. Verify that the request and response service interfaces are set correctly, if not set them as below.



___31. Save the *inquireCatalog* service by closing all open tabs.

___32. Create a *placeOrder* service by repeating steps 1 through 5.

___33. Repeat steps 6 through 10 to create *service interface name* ***placeOrderRequest***

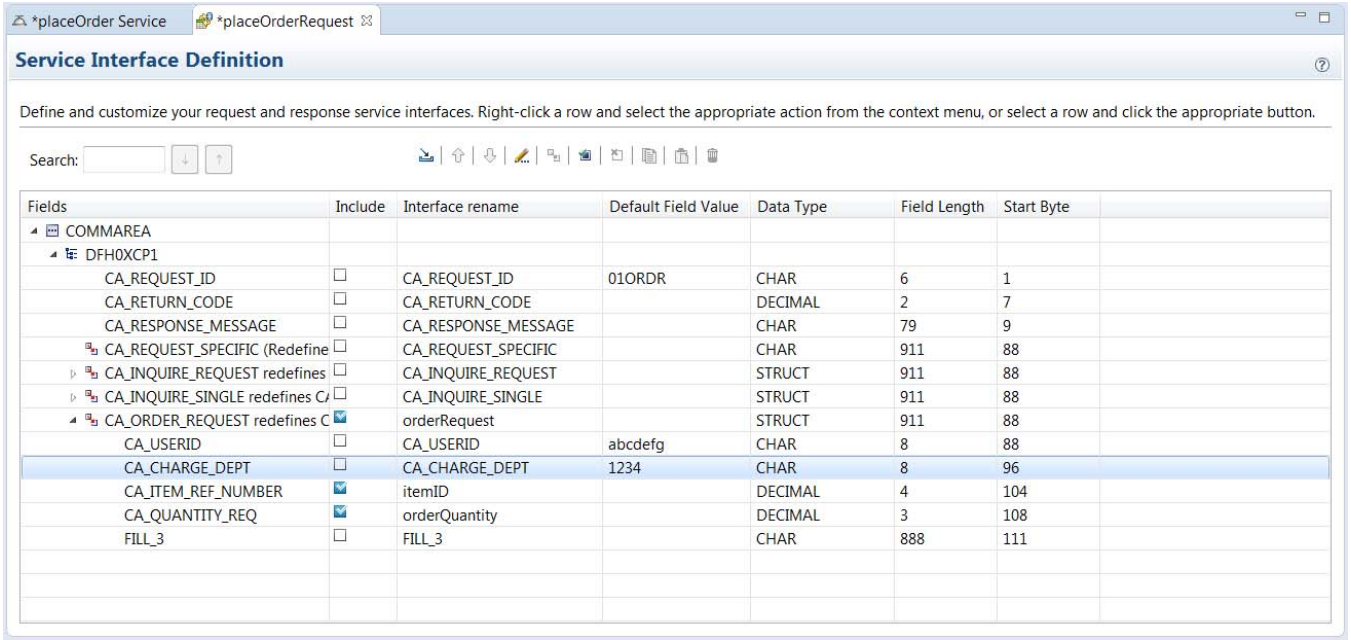
___34. Expand the COMMAREA and perform the following steps for the placeOrderRequest:

a. First uncheck the boxes under *Include* column so that only *CA_ORDER_REQUEST*, *CA_ITEM_REF_NUMBER and CA_QUANTITY_REQ* are checked. These are the fields that will be exposed to the requestor of this API.

b. Double click an entry in the *Interface rename* column to open it for editing. Rename interface fields *CA_ORDER_REQUEST* to ***orderRequest***, ***CA_ITEM_REF_NUMBER*** to ***itemID*** and CQ_QUANTITY_REQ to ***orderQuantity***.

c. Change the default value of field *CA_REQUEST_ID* to a value of ***01ORDR***.

d. Set the default values for *CA_USERID* to ***abcdefgh*** and *CA_CHARGE_DEPT* to ***1234***.

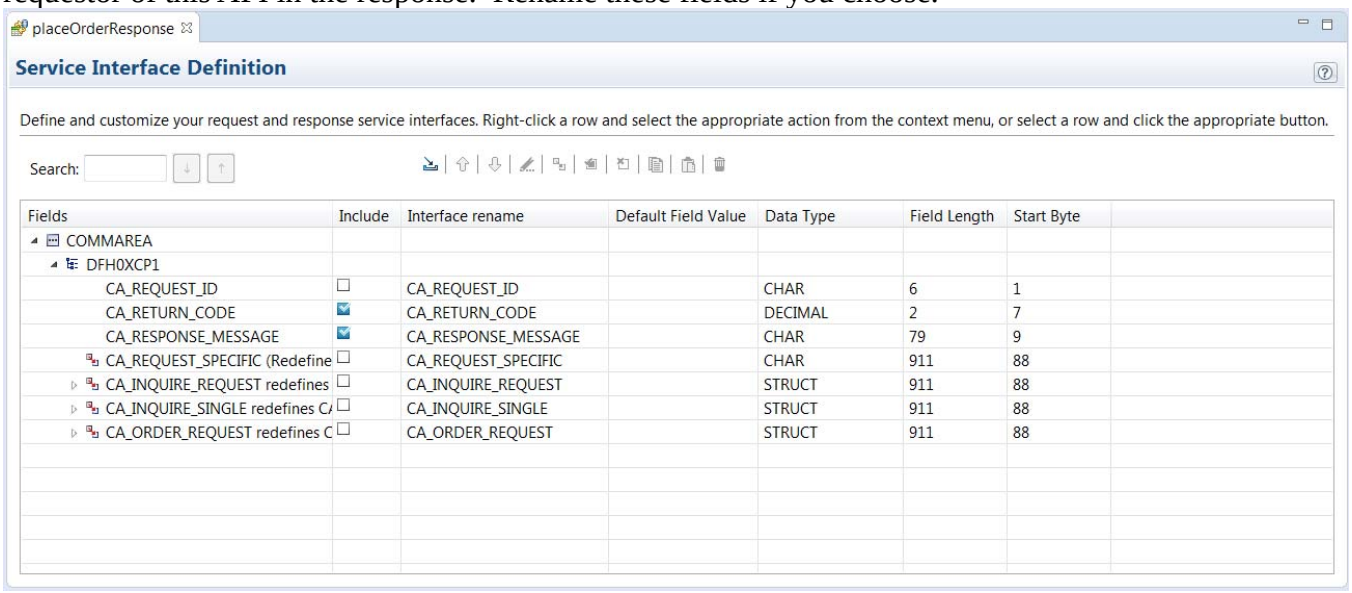


___35. Close the *Service Interface Definition* window.

___36. Repeat steps 6 through 10 to a create service interface ***placeOrderResponse*** response message for the *placeOrder* service.

Mitch Johnson (mitchj@us.ibm.com)

___37. A different set of fields will be returned in the response. Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE* and *CA_RESPONSE_MESSAGE*, are checked. These are fields that will be exposed to the requestor of this API in the response. Rename these fields if you choose.

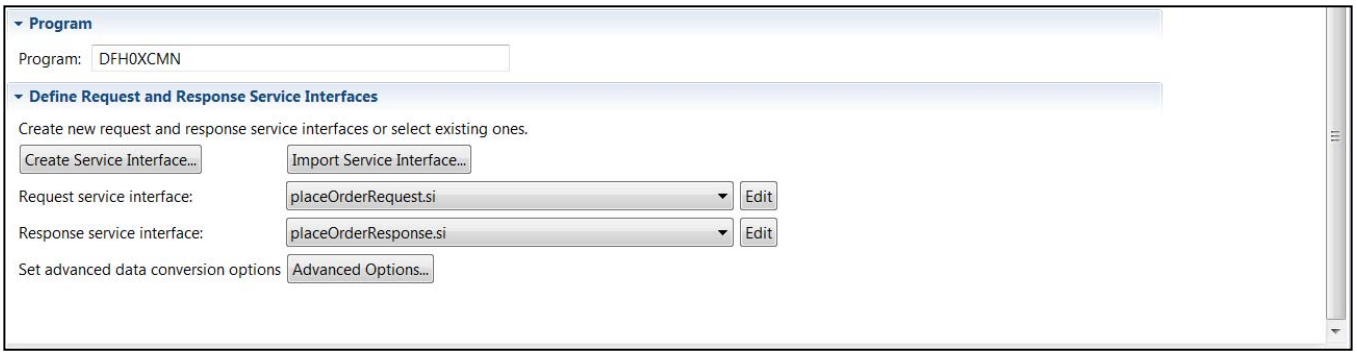


___38. Set the *Response service interface* to *placeOrderResponse.si*.

___39. Close the *Service Interface Definition* window.

___40. Identify the connection reference for which CICS region will be used for this service. Click on the Configuration tab at the bottom of the *Overview* window to display the *Configuration* window. Enter ***catalog*** in the area beside *Connection reference*.

___41. Verify the request and response services interfaces are set correctly, if not set them as below.



___42. Save the *placeOrder* service either by closing all open tabs.

These three services now need to be make available for developing the API and for deployment to the z/OS Connect EE server.

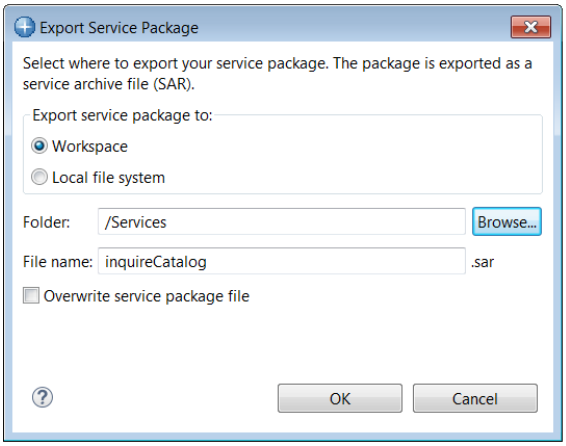
Export and deploy the Service Archive files

Before a service interface can be used it must be exported to create Service Archive (SAR) file and deployed as a SAR file to the server. The exported SAR is used in developing an API in the z/OS Connect EE Toolkit. This section describes the process for exporting and deploying SAR files.

___1. First 'export' them into another project in the z/OS Connect EE Toolkit. Select ***File*** on the tool bar and then on the pop up select **New → Project**. Expand the *General* folder and select *Project* to create a target project for exporting the Service Archive (SAR) files. Click **Next** to continue.

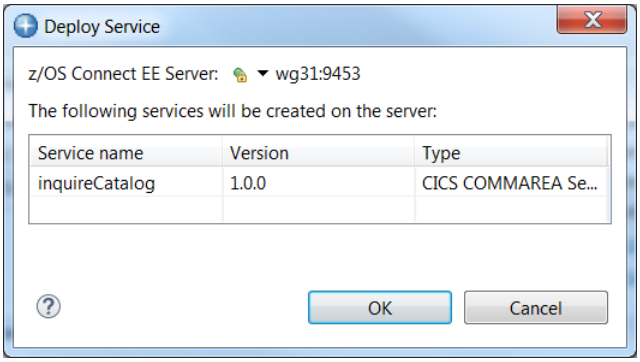
___2. On the *New Project* window enter ***Services*** as the *Project name*. Click **Finish** to continue. This action will add a new project in the *Project Explorer* named *Services*.

___3. Select the *inquireCatalog* service project and right mouse button click. On the pop-up selection select **z/OS Connect EE → Export z/OS Connect EE Service Archive.** On the *Export Services Package* window select the radio button beside *Workspace* and use the **Browse** button to select the *Services* folder. Click **OK** to continue.



Mitch Johnson (mitchj@us.ibm.com)

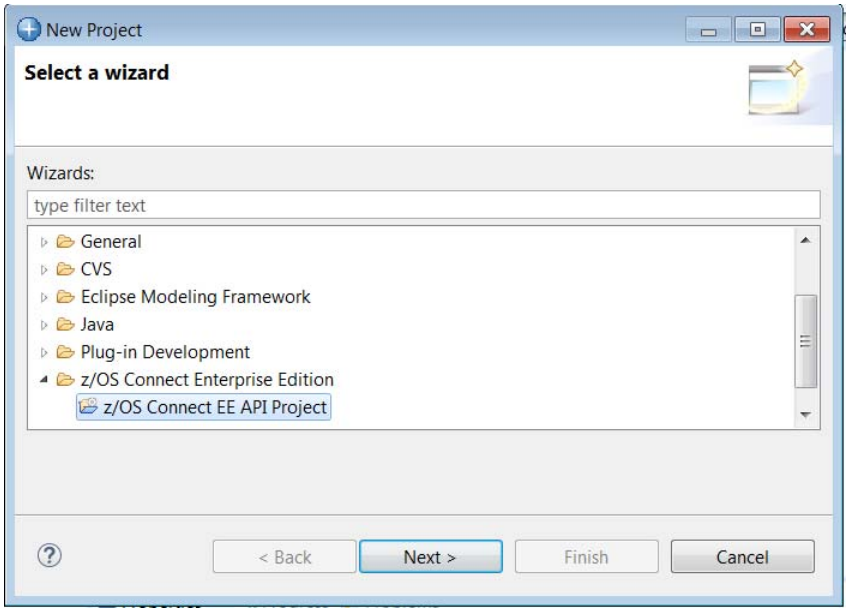
___4. Select the *inquireCatalog* service project again and right mouse button click again and on the pop-up selection select **z/OS Connect EE → Deploy Service to z/OS Connect EE Server**. On the *Deploy Service* window select the target server (*wg31:9453*) and click **OK** twice to continue.



___5. Repeat these two steps to export the *inquireSingle* and *placeOrder* service project to the *Services* project folder and to deploy the services to the z/OS Connect server.

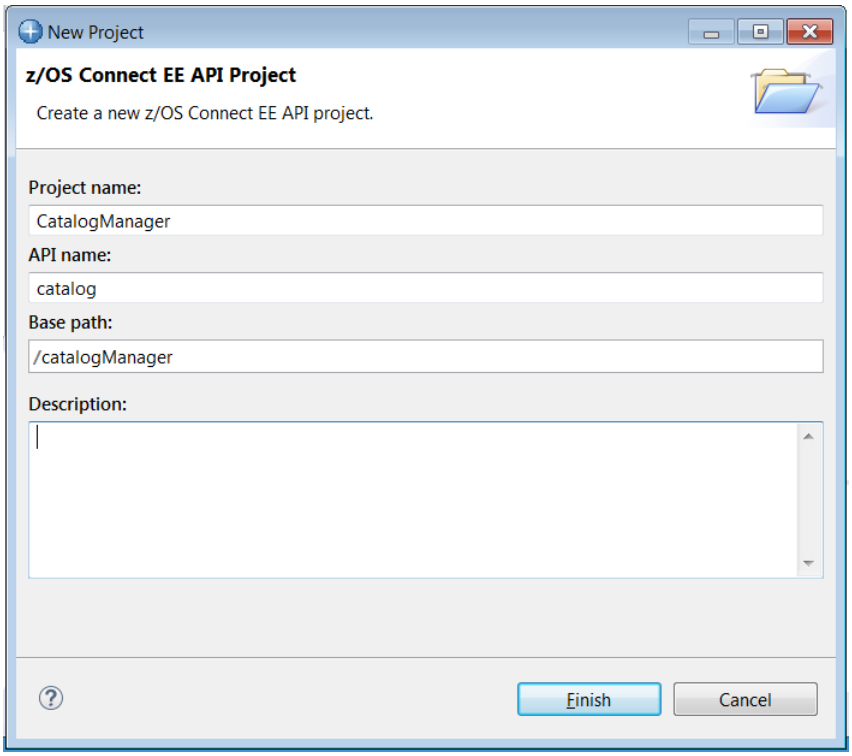
Create the API Project

___1. Back in the z/OS Connect EE tool kit create a new project. 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.

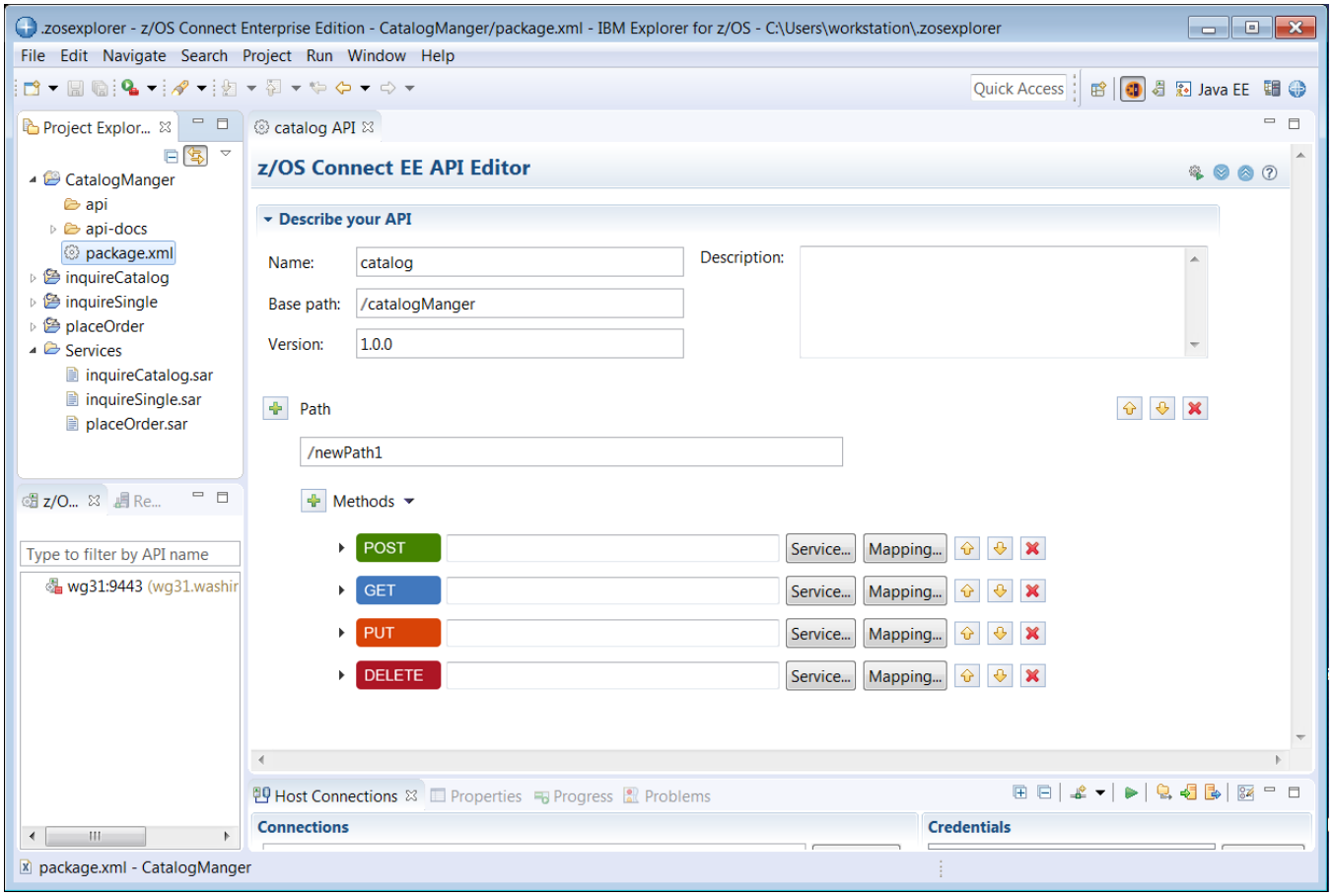


Mitch Johnson (mitchj@us.ibm.com)

___2. Enter ***CatalogManager*** for the *Project name*, ***catalog*** for the *API name* and ***/catalogManager*** for the *Base path* value (case is important). Click **Finish** to continue.



___13. You should now see something like below. The views may need to be adjusted by dragging the view boundary lines.

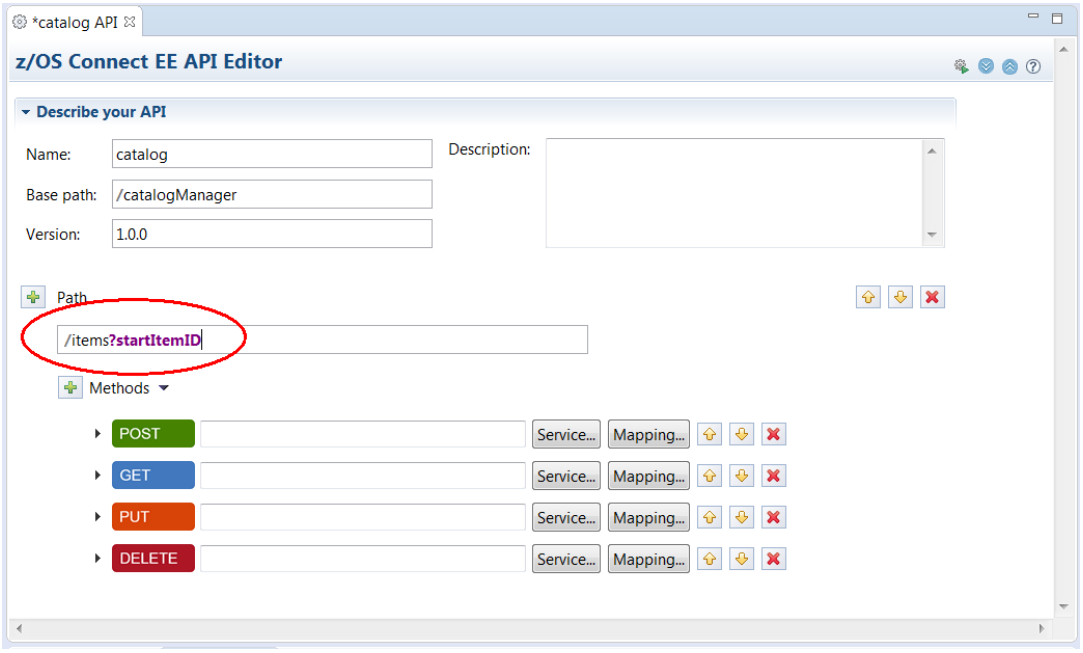


Summary

You created the basic framework for the API project in the API editor.

Compose the API with a CICS COMMAREA application

___1. To compose the *inquireCatalog* service start by entering a *Path* of ***/items?startItemID***.

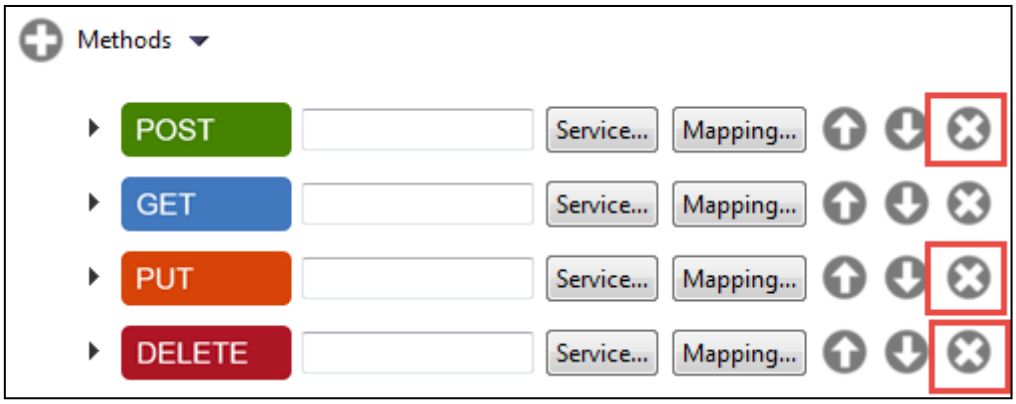


Note: The */items* path element is somewhat arbitrary but use this value so tasks later in the exercise will work unchanged.

The *?startItemID* element tells the editor that value will be supplied as a path parameter in the URL.

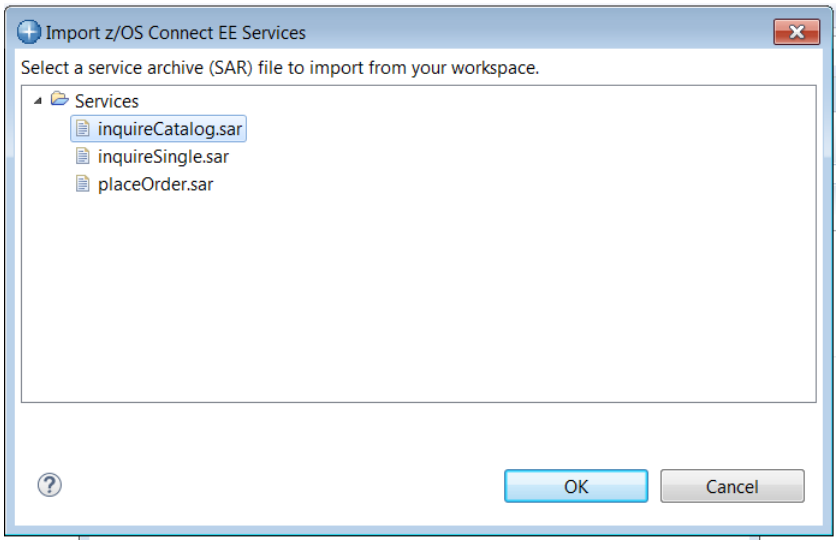
___2. For the *inquireCatalog* service the HTTP method will be **GET**, so we do not need the **POST**, **PUT** and **DELETE** methods. Remove them by clicking the *X* icon to the right of each:

Mitch Johnson (mitchj@us.ibm.com)



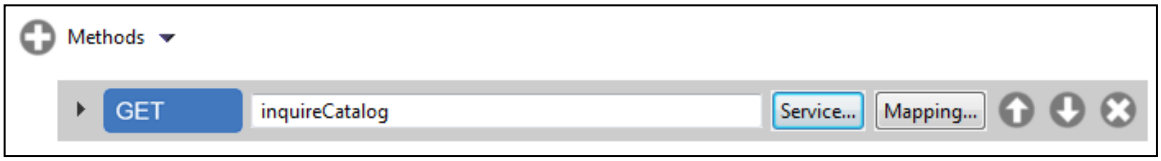
That should leave you with only the **GET** method.

___3. Next import the service required for this method of this *Path*. Click on the **Service** button to the right of the **GET** m*ethod. On the Select a z/OS Connect EE Service window click the* **Workspace** button and expand the *Services* folder on the next window. Select the *inquireCatalog* service from the list of service archive files and click **OK** three times



Mitch Johnson (mitchj@us.ibm.com)

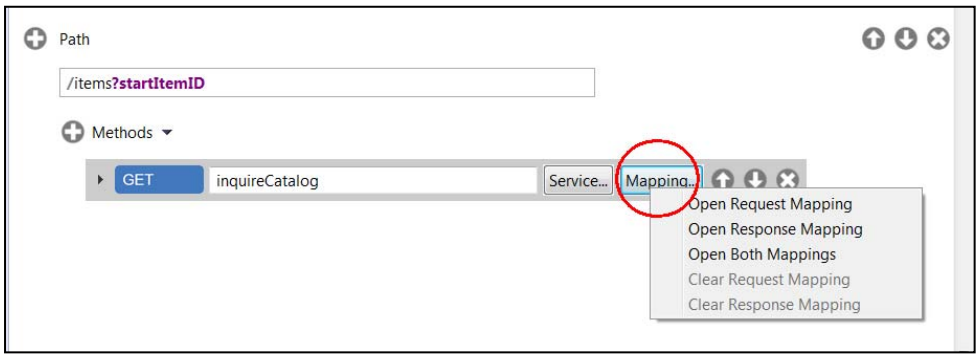
This will populate the field to the right of the method:



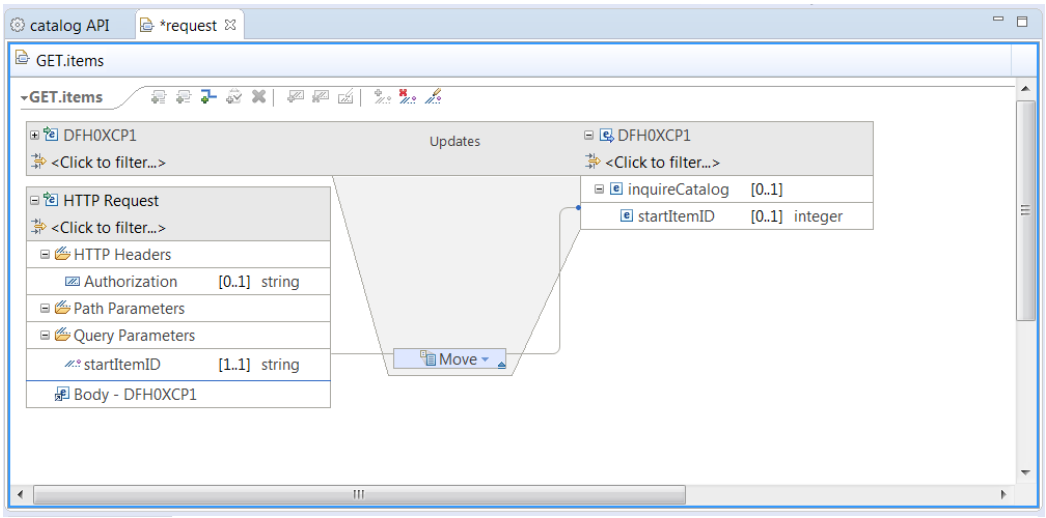
___4. Save the changes so far by using the key sequence **Ctrl-S.**

> **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, then select *Open Request Mapping*:



___6. Expand *inquireCatalog* on the right and next, on the left side hover select *startItemID* and without releasing create a connection between *startItemID* and field *startItemID* on the right hand side. The result is a line that maps the value of *startItemID* from the query parameter to the field *startItemID*. This means the value or contents of *startIemID* query parameter specified in a URL will be moved to the *startItemID* field.



What you have done is reduce the request to a simple GET URI with a single query parameter:

http://<*host*>:<*port*>/catalogManager/items?startItemID=10

The query parameter provides the program the *starting* item number for the listing of items.
Those fields not needed on a request are hidden from the REST client's view. Those fields that have the same static value every time (*ca_request_id*) are assigned a value of 01INQC.

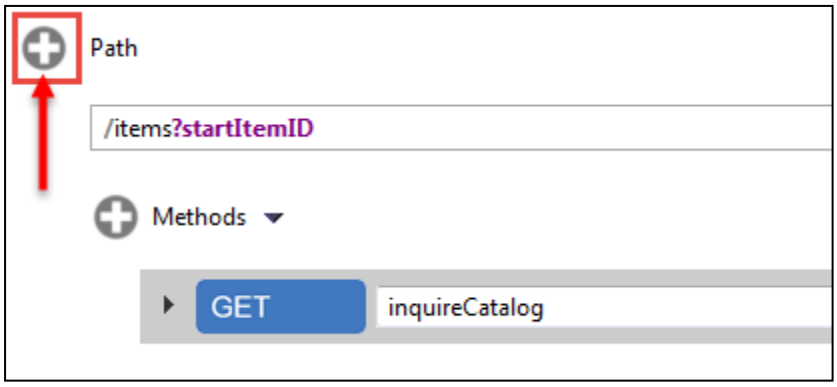
___7. Save the changes using key sequence **Ctrl-S**.

___8. Close the *request* mapping tab by clicking on the *white X* in the tab window.

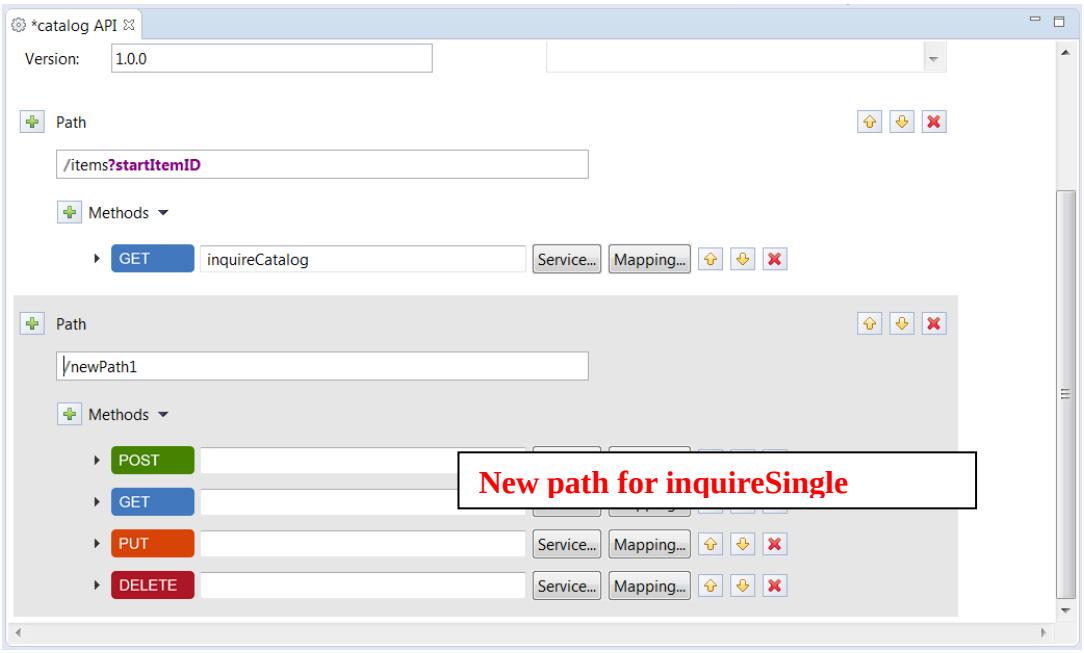
___9. No changes are required for th response mapping.

You're done with the *inquireCatalog* portion of the API. Next up is *inquireSingle*. The steps are very similar. You will find you get better at this the more you do it.

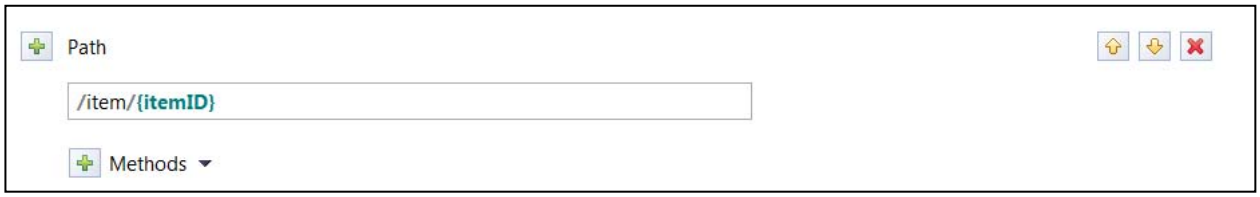
___10. Click the plus icon to add another path to the API.



The result is:



___11. Enter a path value of of ***/item/{itemID}***:

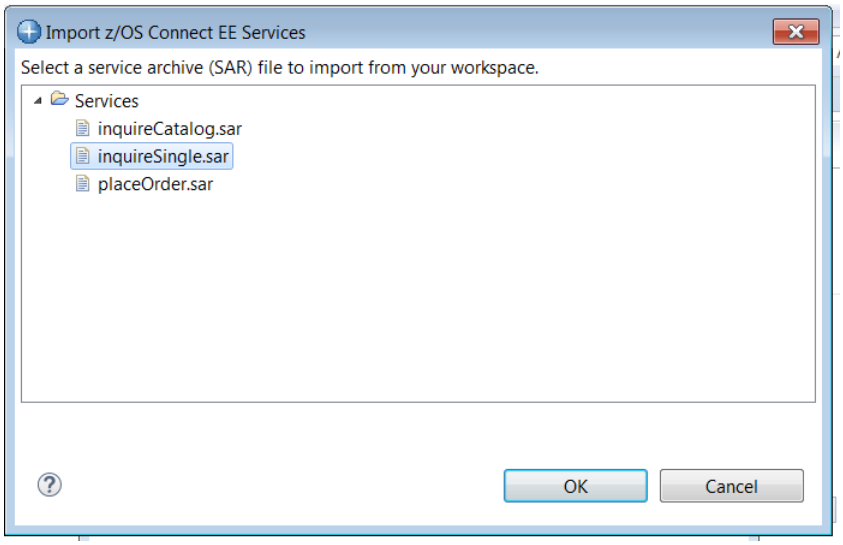


The string *{itemID}* is recognized by the editor as a path parameter.

___12. Once again, the method we will use for this is **GET**, so we do not need **POST**, **PUT** or **DELETE.** Remove those by clicking the *X* to the right of each. The result should be:

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

___14. Next import the service required for this method of this *Path*. Click on the **Service** button to the right of the **GET** m*ethod. On the Select a z/OS Connect EE Service window click the* **Workspace** *button and expand the Services* folder on the next window. Select the *inquireSingle* service from the list of service archive files and click **OK** three times.

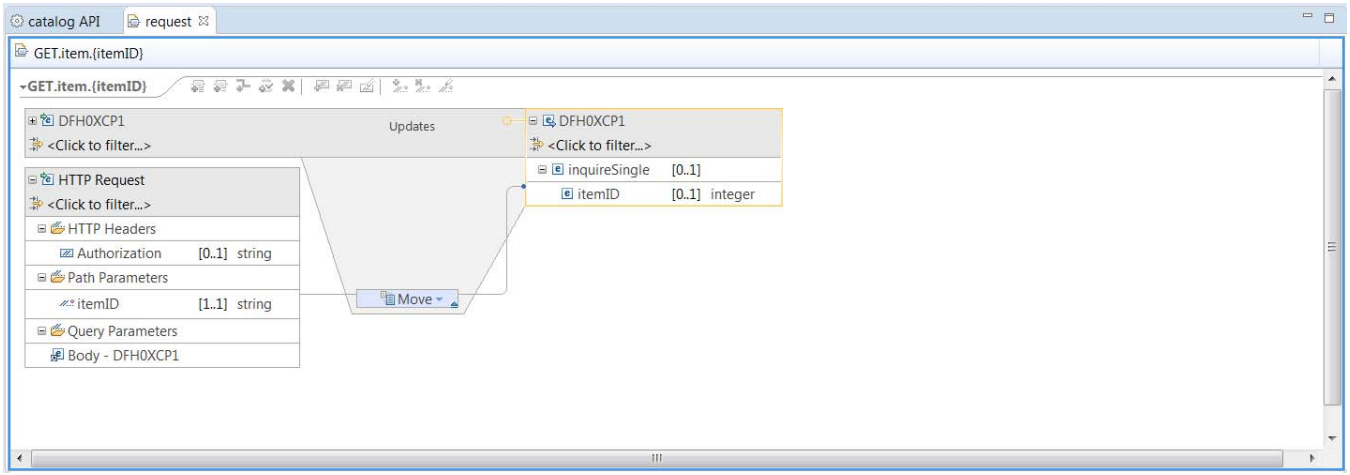


___15. Save the changes by using the key sequence **Ctrl-S**.

___16. Click on *Mapping* → *Open request mapping.*

___17. Expand *inquireSingle* by clicking on the little + sign to the left of the field. Select *itemID* on the left-hand side and drag it over to *itemID* on the right hand side to make a move connection so the value or contents of *itemID* are moved into *itemID* field.

___18. The picture below is the final result. Inspect your mapping and make sure:



What you have done is reduce the request to a simple GET URI with a single path parameter:

http://<host>:<port>/catalogManager/item/10

The path parameter provides the program the item number for the listing of that item..

___19. Save the the changes using the key sequence **Ctrl-S**.

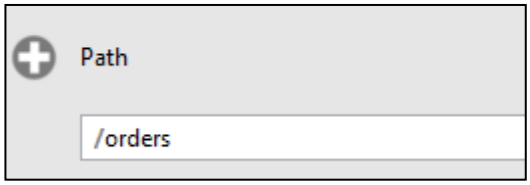
___20. Close the *request* mapping tab.

___21. No changes are required to the *response* mapping.

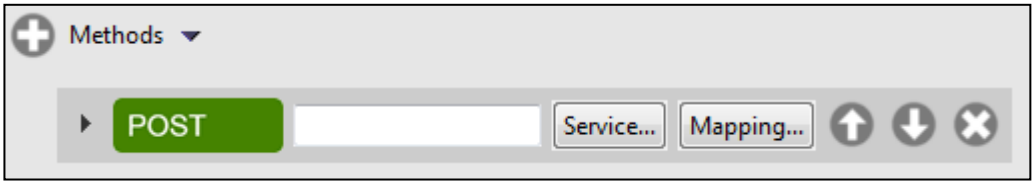
One more service to add – **orderItem**. The process is similar. The steps should be getting familiar to you by this time.

___22. Add another path by clicking on the + circle:

___23. In your new path field, provide a path value of ***/orders***:



___24. The HTTP method for this will be **POST**, so we can get ride of **GET**, **PUT** and **DELETE.** Remove those methods by clicking the **X**. You should then see:



___25. Click on *Mapping* → *Open request mapping* and review. No mappings are required so close the view.

___26. Import the service required for this method of this *Path*. Click on the **Service** button to the right of the **GET** m*ethod. On the Select a z/OS Connect EE Service window click the* **Workspace** *button and expand the Services* folder on the next window. Select the *placeOrder* service from the list of service archive files and click **OK** three times.

___27. Save the changes by using the key sequence **Ctrl-S**.

Summary

You created the API, which consists of three paths and the request and response mapping associated with each. That API will now be deployed into z/OS Connect EE V3.0. The REST interfaces are now simpler than the earlier coarse-grained services. The services are still used, but by z/OS Connect EE V3.0 as it processes the higher-level API requests.

Deploy the API to a z/OS Connect EE Server

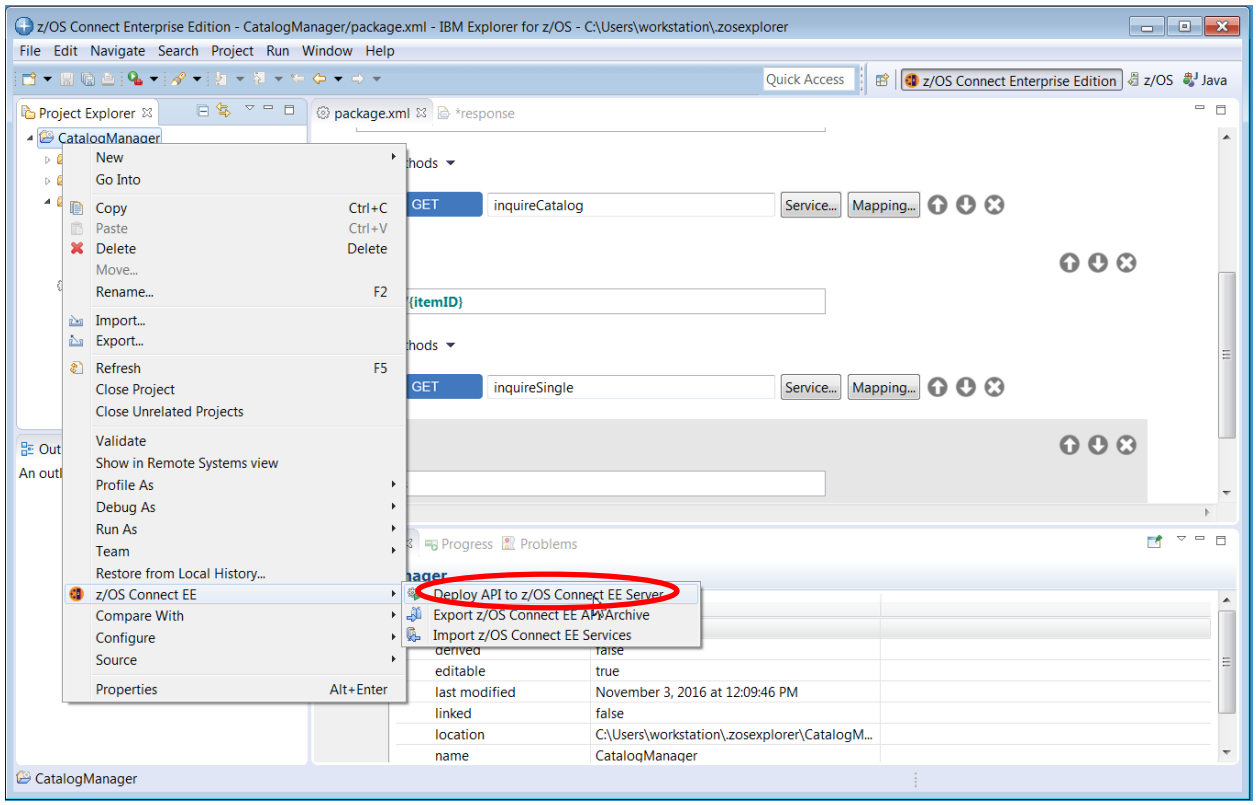
___1. The ***catalog*** API *and* ***inquireCatalog, inquireSingle and placeOrder*** services were defined by the inclusion of the *catalog.xml* file in the *server.xml*.

```
<server description="CICS Catalog">

    <featureManager>
      <feature>zosconnect:cicsService-1.0</feature>
    </featureManager>

    <zosconnect_cicsIpicConnection id="catalog"
      host="wg31.washington.ibm.com"
      port="1491"/>

</server>                                                    "
```

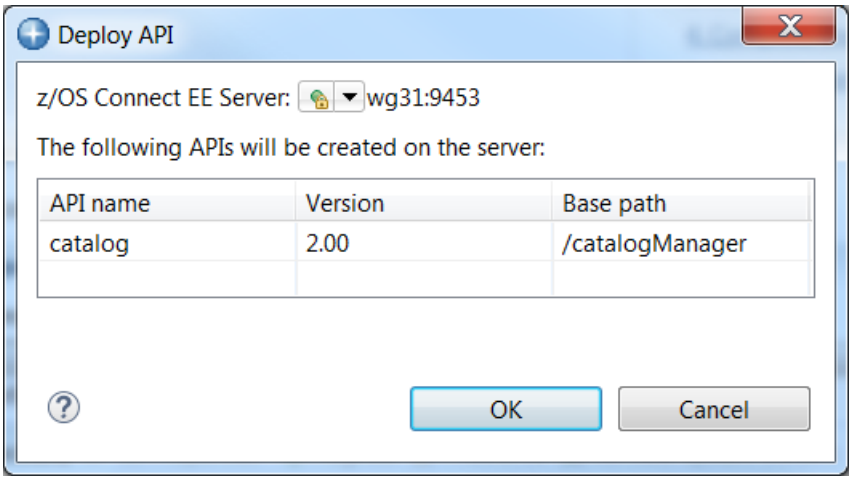
Figure 2 - catalog.xml

The *zosconnect_cicsIpicConnection* element provides the CICS IPIC information that will be used for communications with the CICS region,

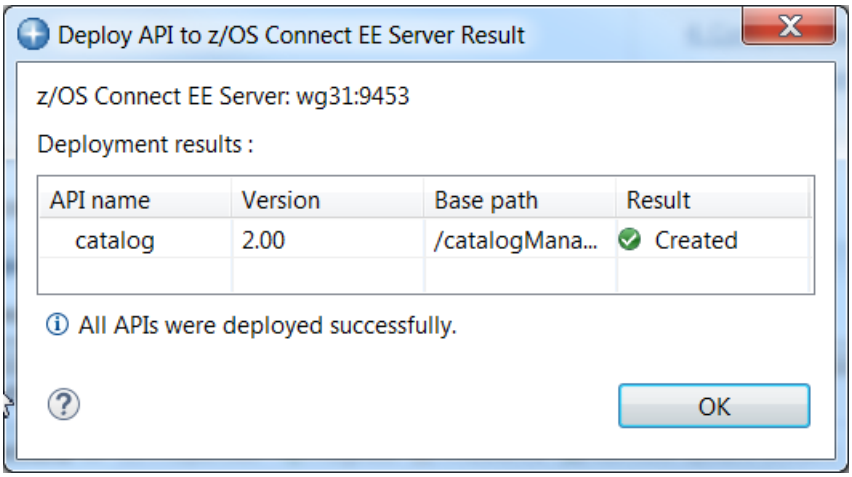
___2. In the *Project Explorer* view (upper left), right-mouse click on the *CatalogManager* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.



___3. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.



___4. The API artifacts will be transferred to z/OS and copied into the */var/zosconnect/servers/zceesrv1/resources/zosconnect/apis* directory.



Summary

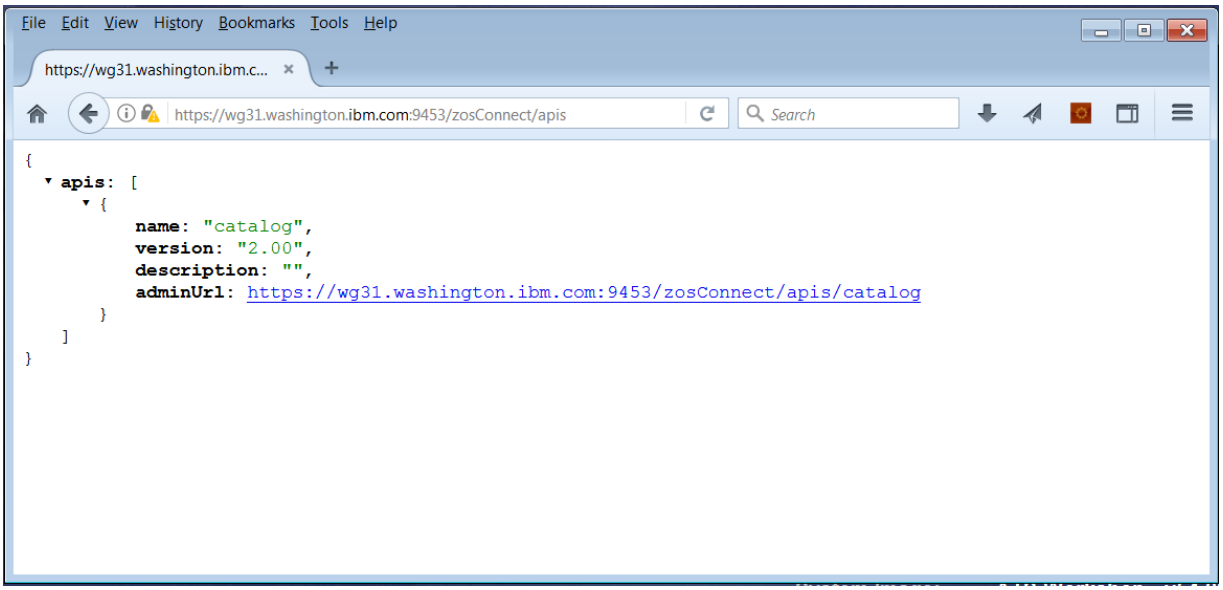
The API was contained in the AAR file, which you uploaded to z/OS and deployed. The update of the server.xml told z/OS Connect EE V3.0 about the presence of the API.

Test the API

Important: The Swagger UI in the API Toolkit has been configured to use an external browser. This simplifies the management of digital certificates.

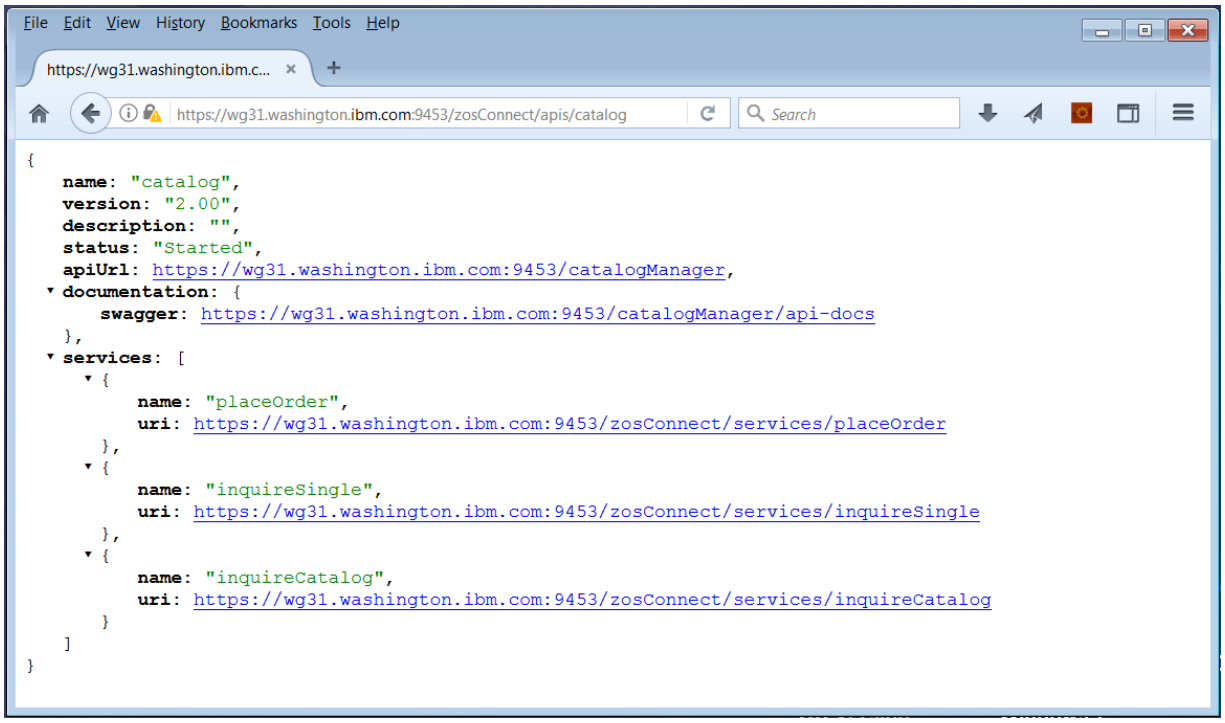
___1. Next enter URL ***https://wg31.washington.ibm.com:9453/zosConnect/apis*** in the Firefox browser and you should see the window below. The *catalog* API now shows as being available.\

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 you for a userid and password. If you do see the prompt, enter the username ***Fred*** and password **fredpwd** (case matters) and click **OK**. Remember we are using basic security and this is the user identity and password defined in the server.xml file.

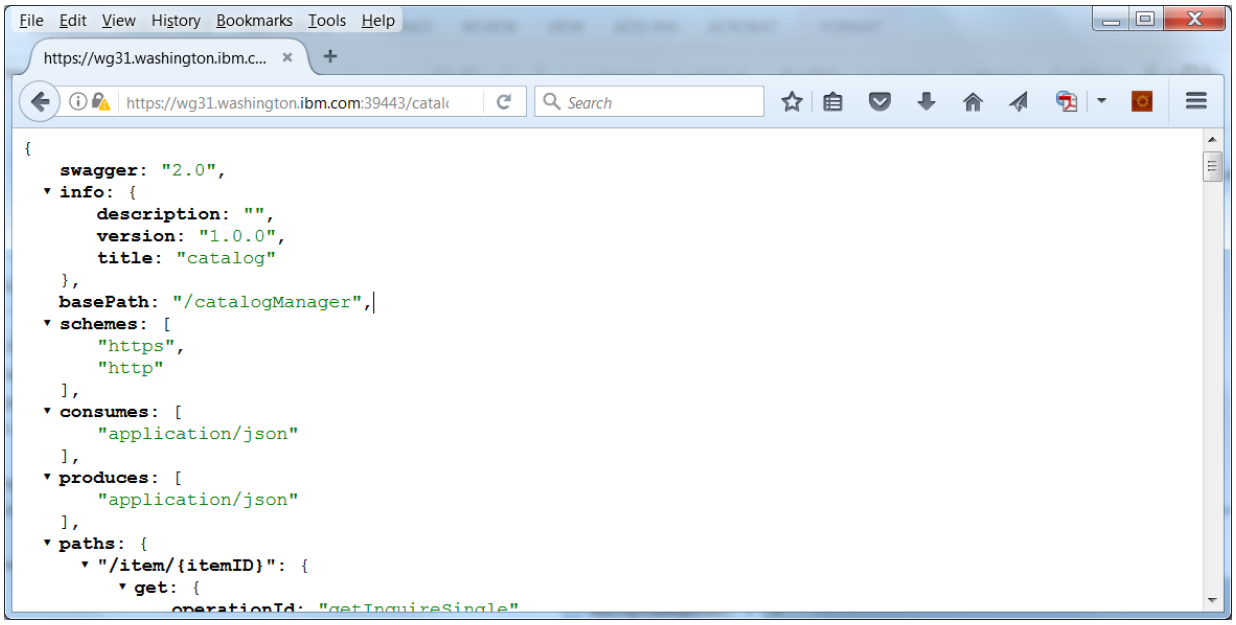


Tech Tip: It is very important to access the z/OS Connect server from a browser prior to any testing using the Swagger UI. Accessing a z/OS Connect URL from a browser starts an SSL handshake between the browser and the server. If this handshake has not performed prior to performing any test the test 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 during this exercise.

___2. If you click on *adminUrl* URL the window below should be displayed:

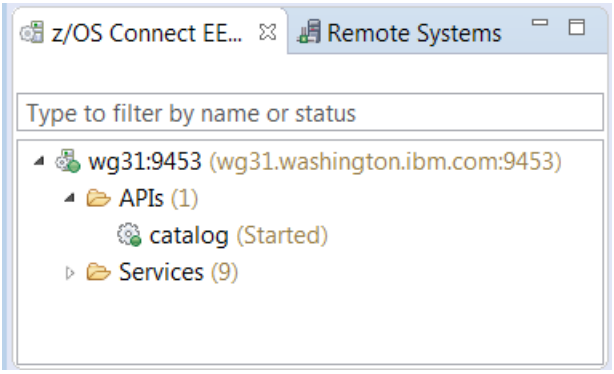


___2. Next click on the *swagger* URL and you should see the Swagger document associated with this API.

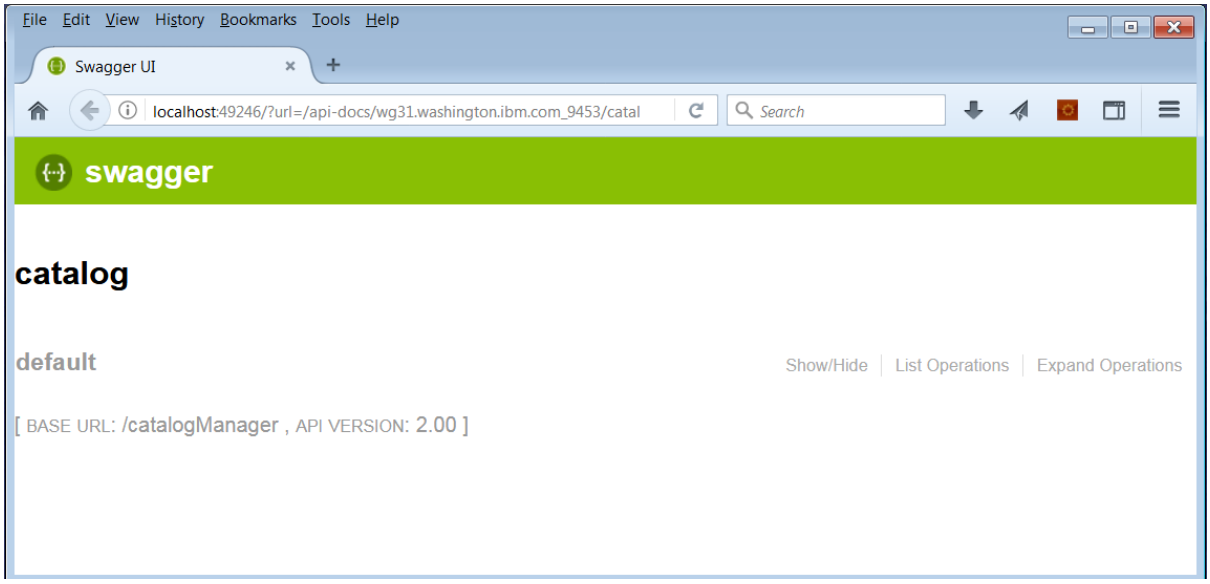


Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This Swagger document can be used by a developer or other tooling to develop REST clients for this specific API.

___3. 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 the expand the *APIs* folder. You should see a list of the APIs installed in the server.

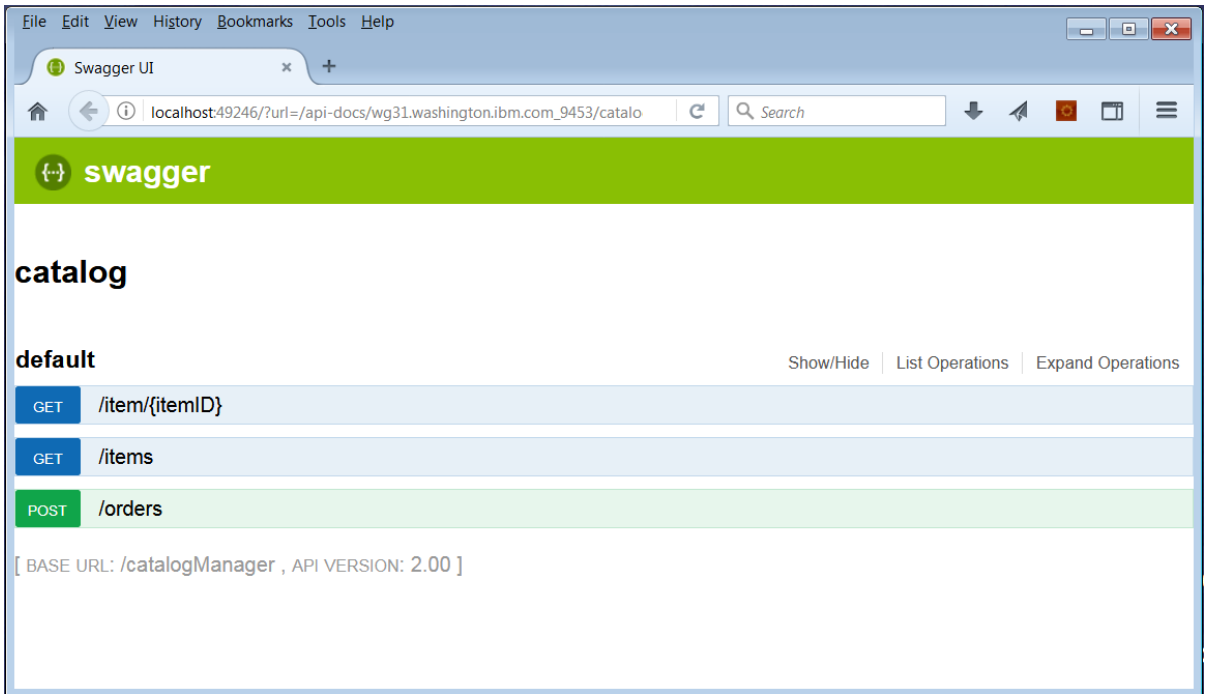


___4. Right mouse button click on *catalog* and select *Open in Swagger UI*. Click **OK** if an informational prompt about certificates appears (see Tech Tip for explanation). This will open a new view showing a *Swagger* test client (see below).

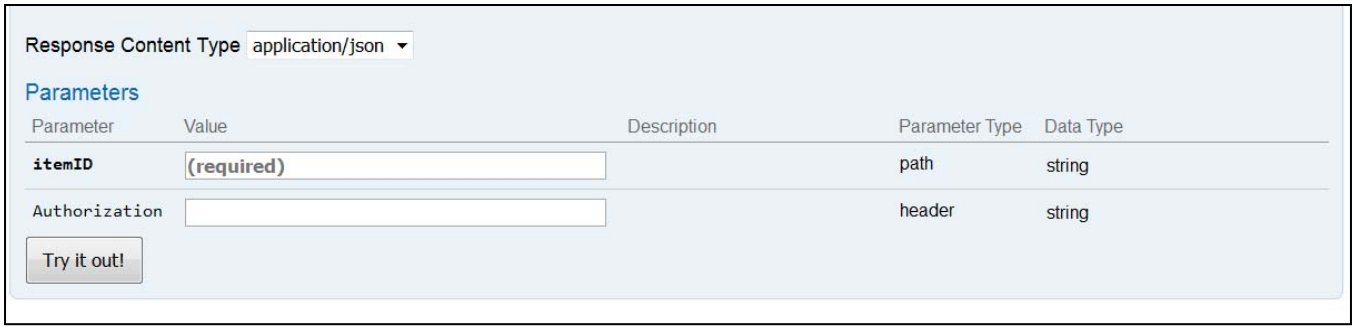


Tech Tip: Swagger UI will not try to automatically download the self-signed certificate from the z/OS Connect EE server. The server's self-signed certificate must be install using other means. In this exercise, the self-signed certificate was installed earlier when displaying the APIs or services installed in the z/OS Connect EE server. Note that this must be done for every server that will be accessed using the Swagger UI.

___5. Click on *List Operations* option in this view and this will display a list of available HTTP methods in this API.



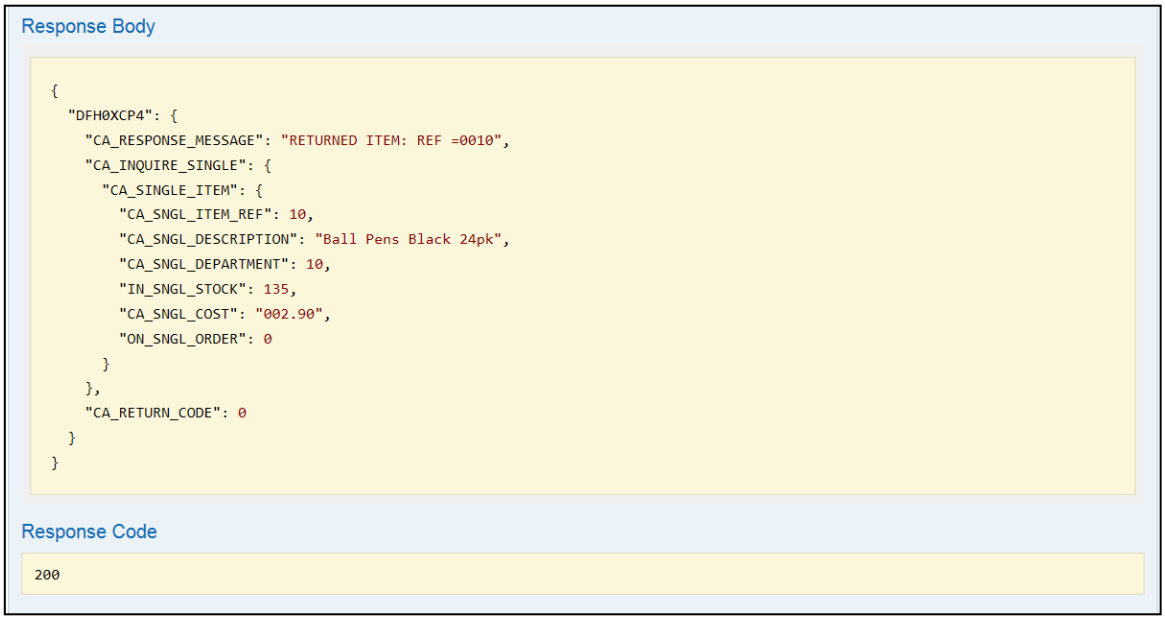
___6. Select the method for selecting a single item from the catalog by item number table by clicking on the *GET* box. This action will expand this method in this view and provides a Swagger UI test client (you may have to use the slider bar and adjust the perspective to see the entire client).



___7. Enter ***10*** in the box beside *itemID* and ***Basic RnJlZDpmcmVkcHdk*** in the area 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:** Since the z/OS Connect EE server require security the Swagger UI must include the authorization string in the Request Header. The format of this authorization string is the type of authorization (e.g. *Basic*) and the base 64 encoded string for *Fred:fredpwd ,e.g RnJlZDpmcmVkcHdk,* see URL https://www.base64encode.org. and/or the *z/OS Connect EE V3 Getting Started Guide*, TechDoc WP102724. Be sure there are no extra spaces at the end of the string.

___8. Scroll down the view and you should see the *Response Body* which contains the results of the GET method (see below). Note that the request field removed from the interface in an earlier steps is not present.



Mitch Johnson (mitchj@us.ibm.com)

___9. Repeat this process with other items and observe the results. For example using a value of ***30*** would return.

Response Body

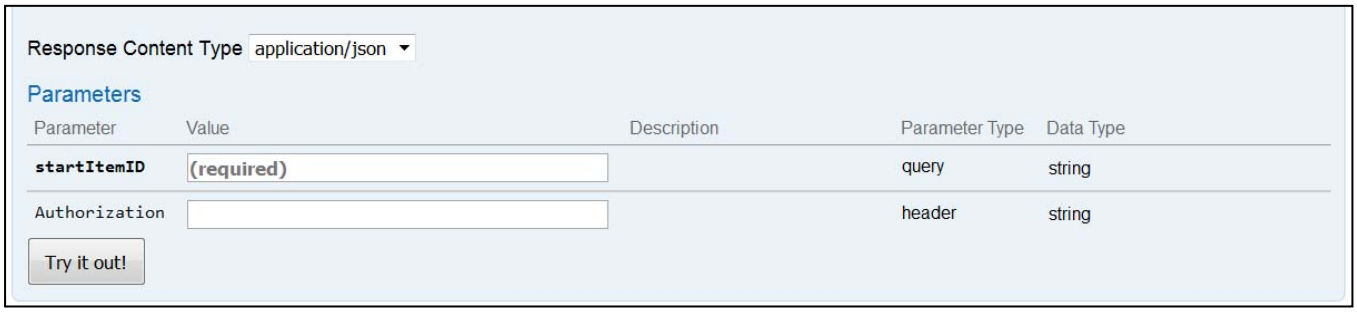

```
{
  "DFH0XCP4": {
    "CA_RESPONSE_MESSAGE": "RETURNED ITEM: REF =0030",
    "CA_INQUIRE_SINGLE": {
      "CA_SINGLE_ITEM": {
        "CA_SNGL_ITEM_REF": 30,
        "CA_SNGL_DESCRIPTION": "Ball Pens Red 24pk",
        "CA_SNGL_DEPARTMENT": 10,
        "IN_SNGL_STOCK": 105,
        "CA_SNGL_COST": "002.90",
        "ON_SNGL_ORDER": 0
      }
    },
    "CA_RETURN_CODE": 0
  }
}
```


The available catalog items are listed below.

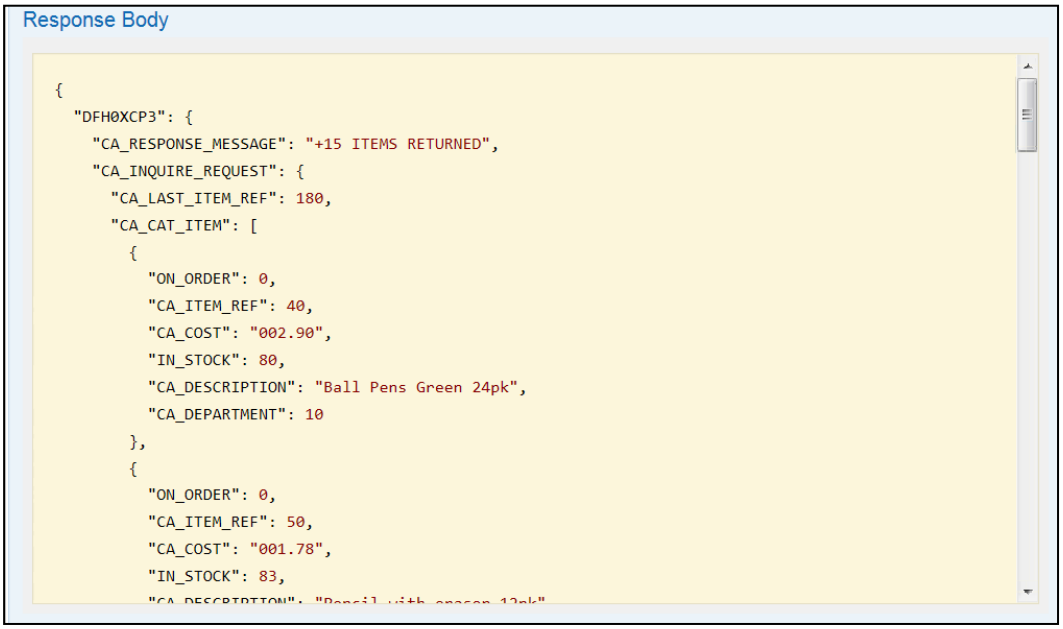
Item#	Description	Dept	Cost	In Stock	On Order
0010	Ball Pens Black 24pk	010	002.90	0135	000
0020	Ball Pens Blue 24pk	010	002.90	0006	050
0030	Ball Pens Red 24pk	010	002.90	0106	000
0040	Ball Pens Green 24pk	010	002.90	0080	000
0050	Pencil with eraser 12pk	010	001.78	0083	000
0060	Highlighters Assorted 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

___10. Collapse the Swagger UI test area for the **GET** single item method for the clicking on the blue *GET* box beside */item/{itemID}*.

___11. Click on the blue *GET* box beside */items* to access the **GET** method to open its Swagger Test user interface for the inquire catalog service (e.g. *inquireCatalog*) and scroll down to the *Response Content Type* area.

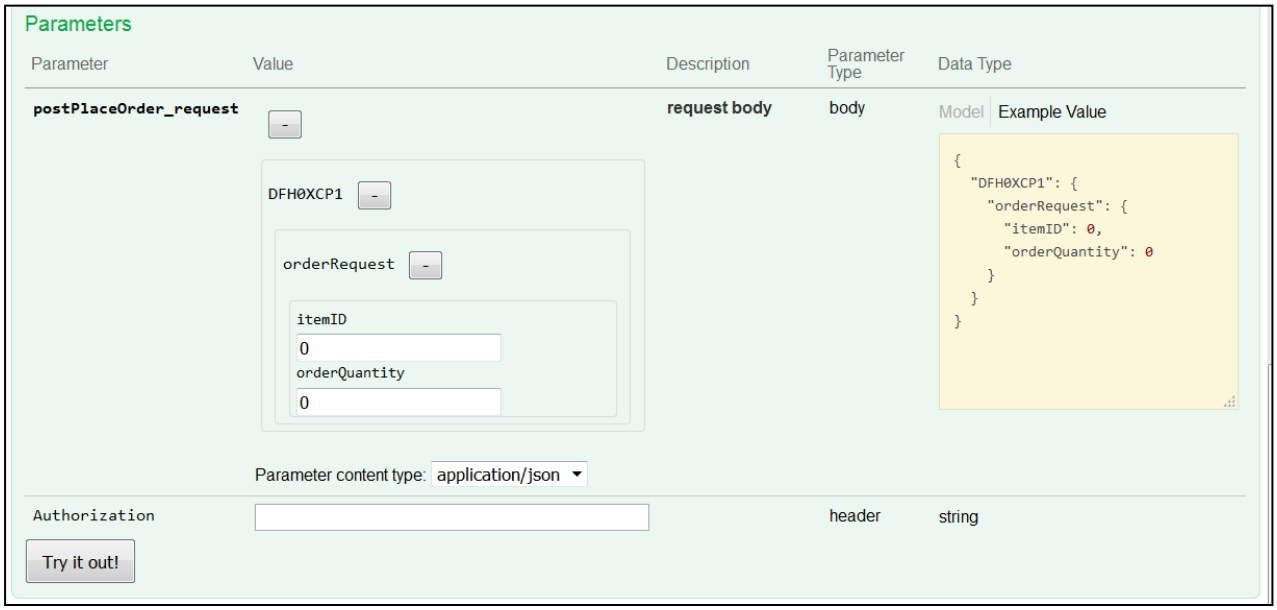


___12. Enter ***40*** in the box beside *startItemID* and ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and press the **Try it out!** button. Scroll down and you should see the following information in the *Response Body*.



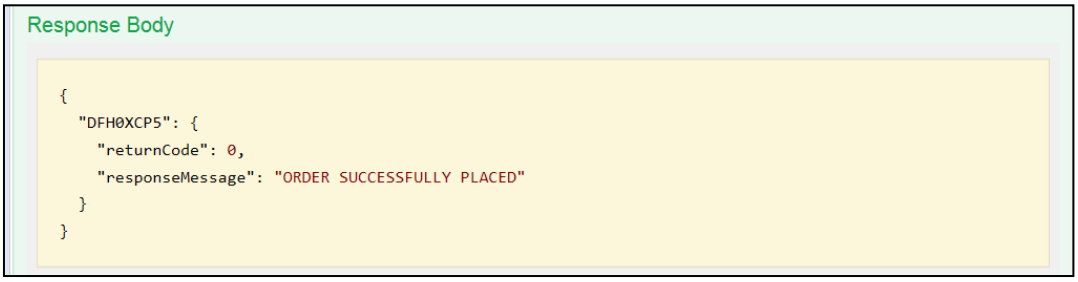
Try a few other values for *startItemID* and compare the resultse.

___13. Click on the green POST box to access the **POST** method to open its Swagger Test user interface for the place order service (e.g. *placeOrder*) and scroll down to the *Response Content Type* area.



___14. Enter ***40*** in the area below *itemID* and ***1*** in the area under *orderQuantity* and ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and press the **Try it out!** button

Scroll down and you should see the following information in the *Response Body*.

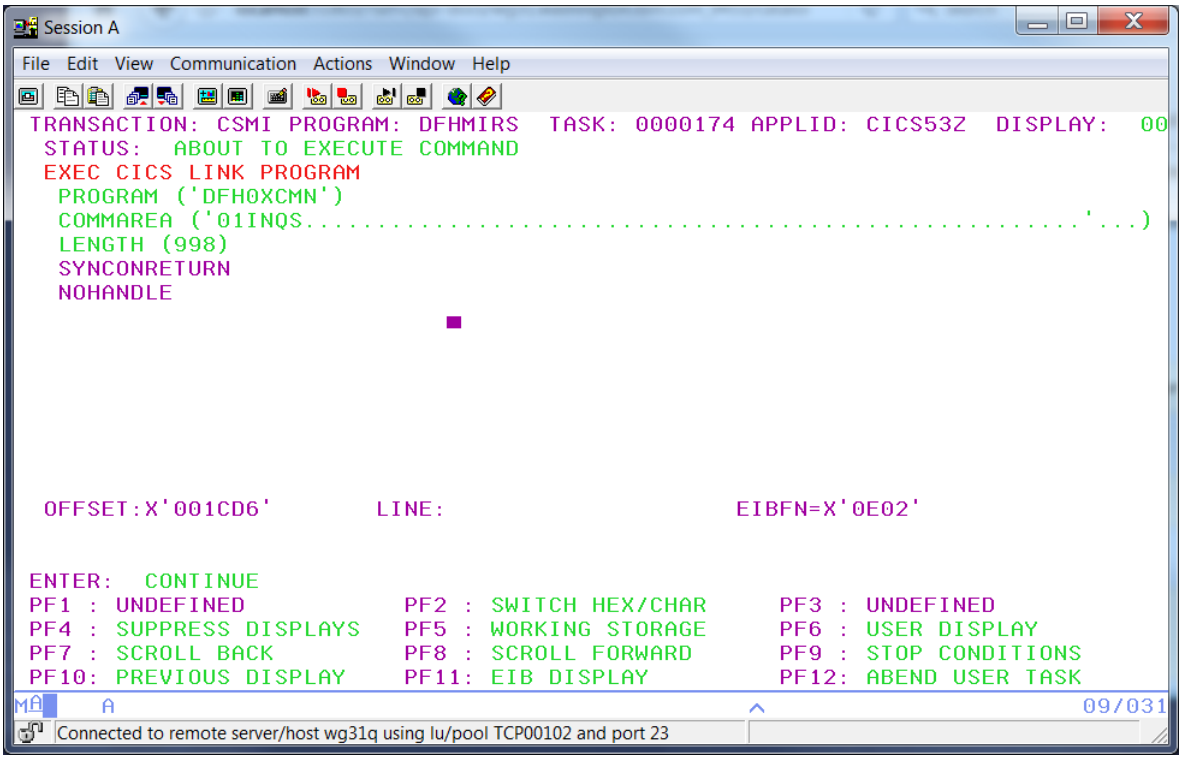


Summary

You have verified the API. The API layer operates above the service layer you defined and tested earlier. The API layer provides a further level of abstraction and allows a more flexible use of HTTP verbs, and better mapping of data via the API editor function.

Optional

Start a 3270 session with CICS and enter CICS transaction ***CEDX CSMI***. When you repeat of any of the above test you should be able to trace the flow of the request through CICS.



Summary

You have verified the API. The API layer operates above the service layer you defined. The API layer provides a further level of abstraction and allows a more flexible use of HTTP verbs, and better mapping of data via the API editor function.