

IBM z/OS Connect (OpenAPI 2.0)

Developing RESTful APIs for a CICS Container program



Table of Contents

| | |
|---|----------|
| Overview | 3 |
| Connect the IBM z/OS Explorer to the z/OS Connect Server | 4 |
| z/OS Connect APIs and a CICS program | 7 |
| Create the services..... | 7 |
| Create the “Insert” service..... | 7 |
| Create the “Delete” service..... | 17 |
| Create the “Update” Service | 19 |
| Create the “Select” Service | 22 |
| Export and deploy the Service Archive files..... | 25 |
| Create the CscvincAPI API project..... | 26 |
| Compose the API for the CICS Container Application | 28 |
| Deploy the API to a z/OS Connect Server..... | 38 |
| Test the APIs with a CICS Container Application | 40 |

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.

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 and the API Toolkit. These two products allow the exposure of z/OS resources to JSON clients. For information about scheduling this workshop in your area contact your IBM representative.

If you have completed another the developing APIs exercise for this workshop you can start with section *z/OS Connect APIs and a CICS program* on page 7.

General Exercise Information and Guidelines

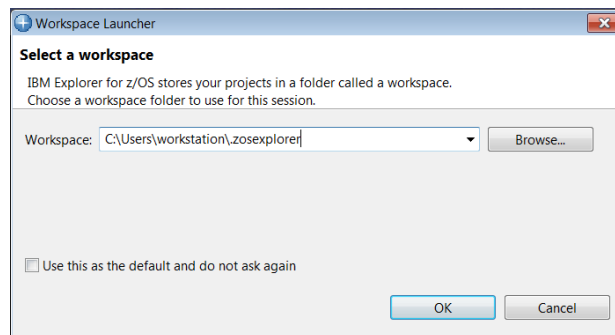
- ✓ This exercise requires using z/OS user identity *USER1*. The RACF password for this user is USER1.
- ✓ Any time you have any questions about the use of IBM z/OS Explorer, 3270 screens, features or tools do not hesitate to ask the instructor for assistance.
- ✓ Text in **bold** and highlighted in **yellow** in this document should be available for copying and pasting in a file named *Development APIs CopyPaste* file on the desktop.
- ✓ Please note that there may be minor differences between the screen shots in this exercise versus what you see when performing this exercise. These differences should not impact the completion of this exercise.
- ✓ For information regarding the use of the Personal Communication 3270 emulator, see the *Personal Communications Tips* PDF in the exercise folder.

Connect the IBM z/OS Explorer to the z/OS Connect 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.

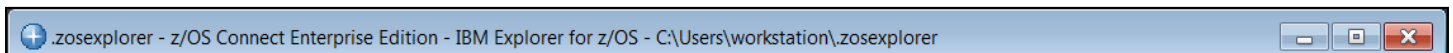
Tech-Tip: Windows desktop tools can be opened either by double clicking the icon or by selecting the icon and right mouse button clicking and then selecting the *Open* option.

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:



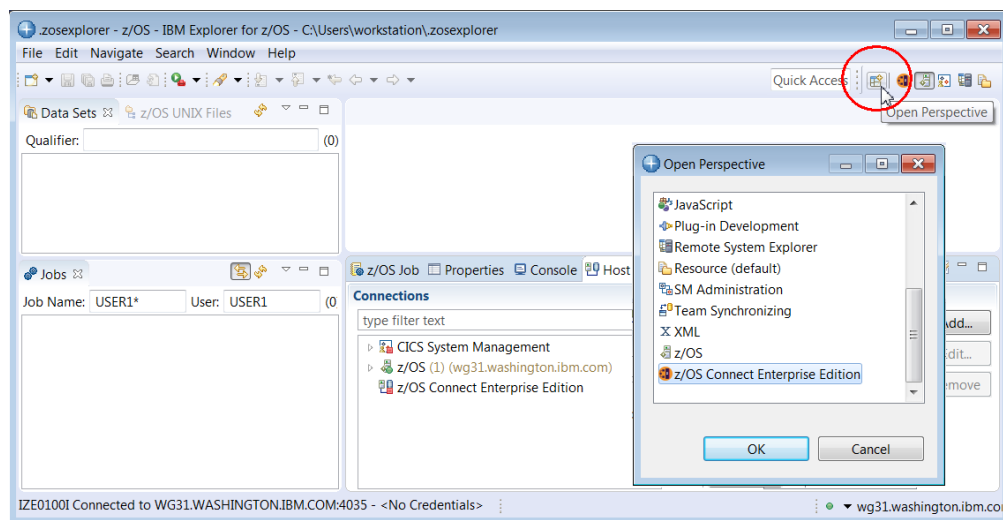
Take the default value by clicking **OK**.

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 connections* tab in the lower view and then click the **Add** button.



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

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

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

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

6. In the pop-up list displayed select *z/OS Connect Enterprise Edition* and on the *Add z/OS Connect Enterprise Edition Connection* 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 APIs and a CICS program

This exercise provides an opportunity to compose and deploy an API that invokes a CICS program. This program happens to interact using CICS Channels and Containers, but the process is fundamentally the same for CICS programs that interacts using COMMAREAs.

The target CICS program, *CSCVINC*, is a simple CICS program receives a container in a cannel. The program accesses a VSAM file based on an *action* field in the container. If the *action* field contains an **I**, the program inserts a new record into the VSAM file. If the *action* field contains a **D**, the record identified by the key field is deleted. If the *action* field contains a **U** the program updates an existing record with new data from the container. Finally, ff the *action* field contains a **S**, the program selects or retrieves a record based on the key field in the container. The name of the request container is reused for the name of the response container. This allows accessing the program without any dependency on a specific container name.

The 4 functions of the CSCVINC program (insert, delete, update and select) will be exposed as 4 services that correspond to the HTTP methods POST, DELETE, PUT and GET respectively.

These 4 services will then be integrated into a RESTful API. Note that the response messages for the *POST*, *DELETE*, and *PUT* methods of this API will return to the client only the HTTP response code.

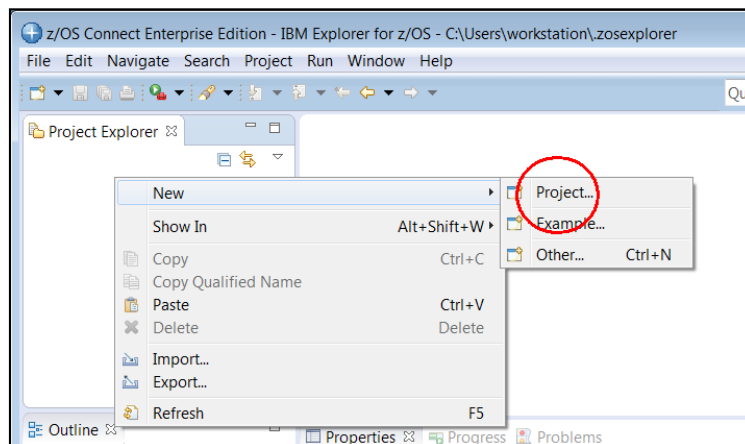
Create the services

The first step is to create the 4 services which describe the interaction with the CICS program CSCVINC. Each service will correspond to the insert, delete, update or select functions described above.

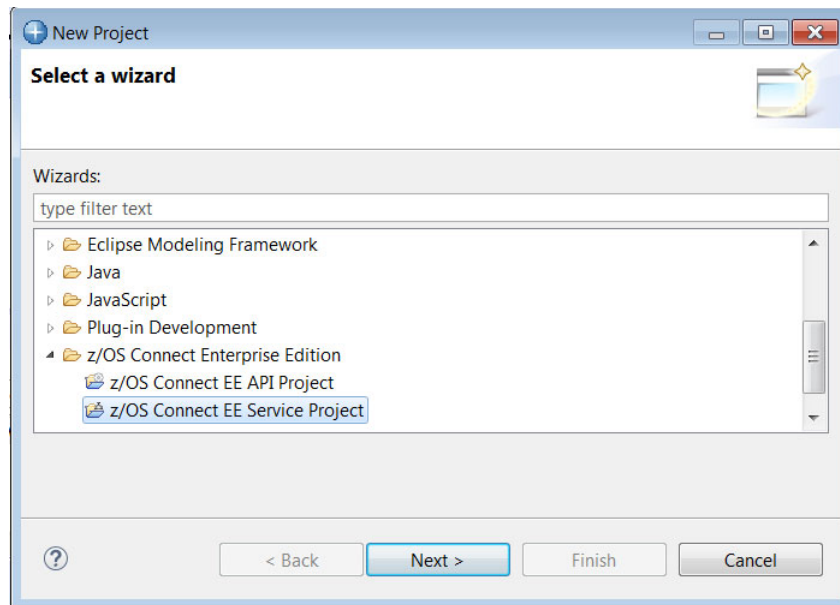
Switch to the *z/OS Connect Enterprise Edition* perspective. Let's start with the *insert* or *POST* service.

Create the “Insert” service

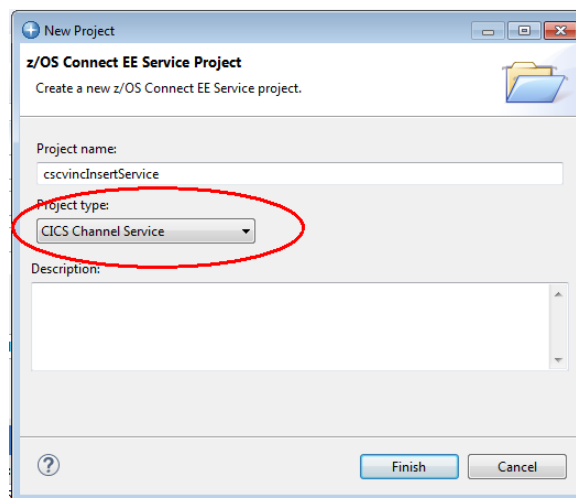
1. In the upper left, position your mouse anywhere in the *Project Explorer* view and right-mouse click, then select *New* → *Project*:



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



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



Tech-Tip: This is really the only difference between creating a service that accesses a CONTAINER enabled CICS program versus creating a service that accesses a COMMAREA enabled CICS program.

4. This will open the *Overview* window for the *cscvincInsertService*. For now, disregard the message about the 3 errors addressed shortly.

Tech-Tip: If this view is closed it can be reopened by double clicking the *service.properties* file in the service project.

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

Tech-Tip: The CICS program name must be entered in upper case unless the program has been defined in CICS in lower case.

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

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

[illegible]

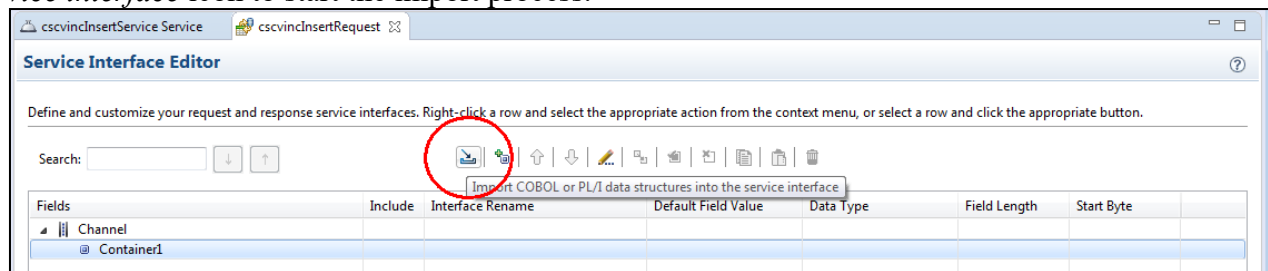
8. The first step is to import the COBOL copy book (see below) that represents the inbound or request CONTAINER.

```

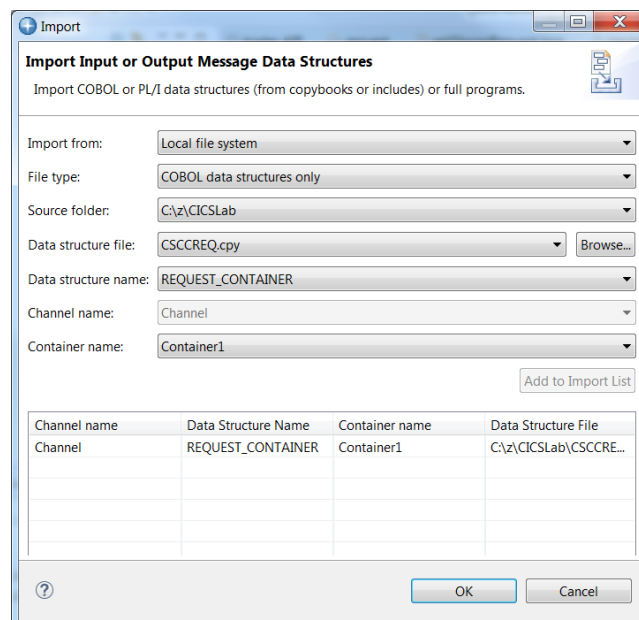
01 Request-Container.
  03 ACTION          PIC X(1) .
  03 USERID          PIC X(8) .
  03 FILEA-AREA.
    05 STAT          PIC X .
    05 NUMB          PIC X(6) .
    05 NAME          PIC X(20) .
    05 ADDR          PIC X(20) .
    05 PHONE         PIC X(8) .
    05 DATEX         PIC X(8) .
    05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

```

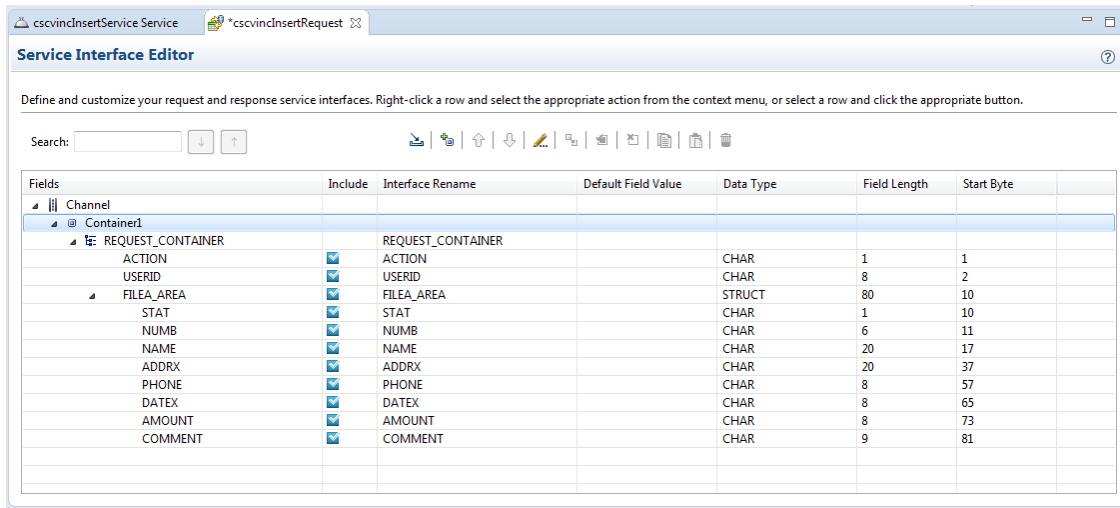
9. On the *Service Interface Definition* window, there is a tool bar near the top. If you hover over an icon its function will be display as below. Select *Container1* and click the *Import COBOL or PL/I data structure into the service interface* icon to start the import process.



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 *CSCCREQ.cpy* and click **Open** to import this file into this project. Click the **Add to Import List** button to continue.

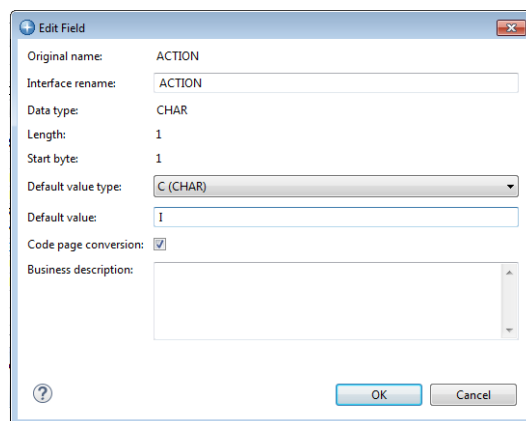


11. Click **OK** and when you expand *Container1*, *REQUEST_CONTAINER* and *FILEA_AREA* you will see the COBOL ‘variables’ that have been imported into the service project.

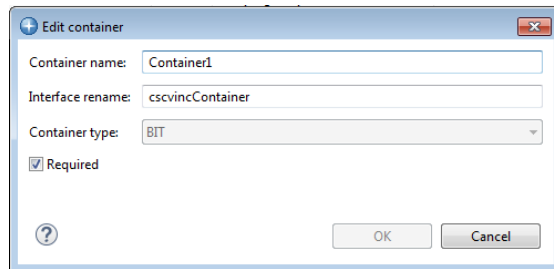


N.B. by default all of the variables/fields are automatically selected to be include in the interface and their default interface names are derived from the COBOL source shown earlier.

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 field(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 set its default value to *I* by using the right mouse button technique or the *Edit selected field* icon (the pencil) in the tool bar.



17. Next remove all of the fields from the service interface by unchecking all of the boxes in the *Include* column.
18. Select the *Container1* field and use the *Edit selected data structure* button to rename the interface name for the container to ***cscvincContainer*** (see below).



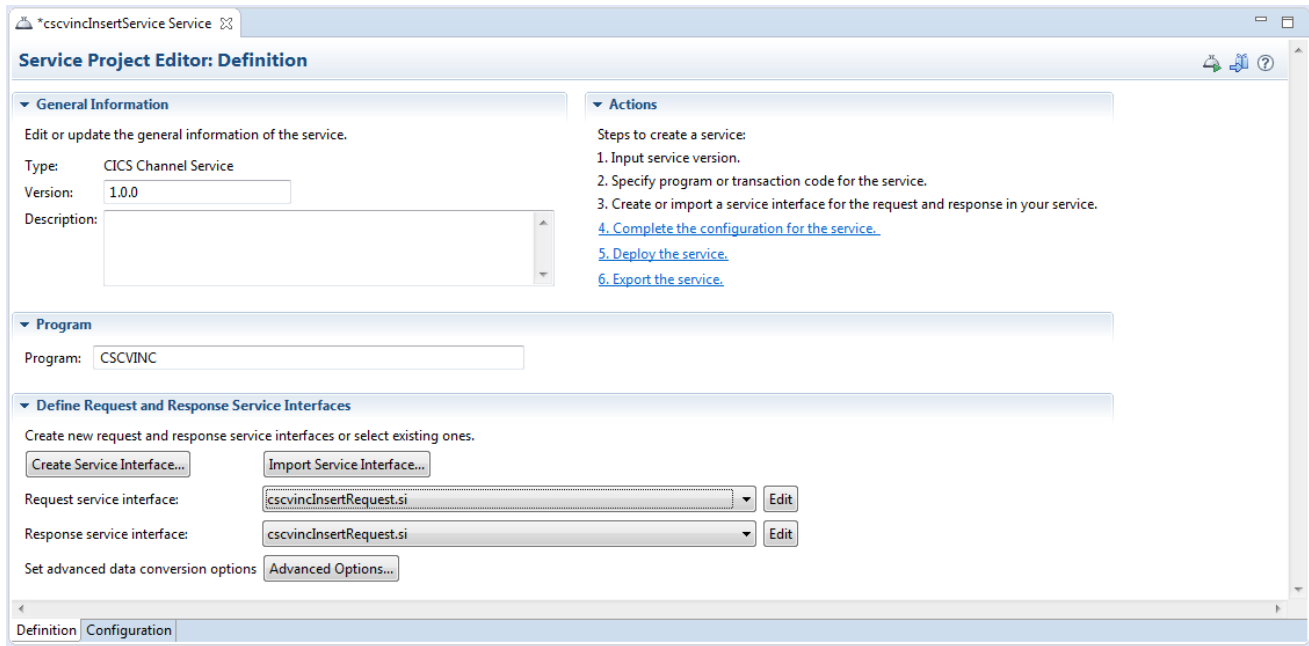
19. Check the box beside the *NUMB*, *NAME*, *ADDRX*, *PHONE*, *DATEX* and *AMOUNT* fields so they will be included in the interface. Note the higher structure (*FILEA_AREA*) will be automatically selected also.
20. Rename the other interface names for the other interface names as shown below:

- *REQUEST_CONTAINER* to ***request***
- *FILEA_AREA* to ***filea***
- *NUMB* to ***employeeNumber***
- *NAME* to ***name***
- *ADDRX* to ***address***
- *PHONE* to ***phoneNumber***
- *DATEX* to ***date***
- *AMOUNT* to ***amount***

N.B. if the interface names are not renamed, the original interface names will appear in subsequent screen shots.
When finished your service definition interface should look like this.

| Service Interface Editor | | | | | | | |
|---|-------------------------------------|------------------|---------------------|-----------|--------------|------------|--|
| Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and click the appropriate button. | | | | | | | |
| Search: | | | | | | | |
| Fields | Include | Interface Rename | Default Field Value | Data Type | Field Length | Start Byte | |
| Channel | | | | | | | |
| Container1 | | cscvincContainer | | | | | |
| REQUEST_CONTAINER | | request | | | | | |
| ACTION | <input type="checkbox"/> | ACTION | I | CHAR | 1 | 1 | |
| USERID | <input type="checkbox"/> | USERID | | CHAR | 8 | 2 | |
| FILEA_AREA | <input checked="" type="checkbox"/> | filea | | STRUCT | 80 | 10 | |
| STAT | <input type="checkbox"/> | STAT | | CHAR | 1 | 10 | |
| NUMB | <input checked="" type="checkbox"/> | employeeNumber | | CHAR | 6 | 11 | |
| NAME | <input checked="" type="checkbox"/> | name | | CHAR | 20 | 17 | |
| ADDRX | <input checked="" type="checkbox"/> | address | | CHAR | 20 | 37 | |
| PHONE | <input checked="" type="checkbox"/> | phoneNumber | | CHAR | 8 | 57 | |
| DATEX | <input checked="" type="checkbox"/> | date | | CHAR | 8 | 65 | |
| AMOUNT | <input checked="" type="checkbox"/> | amount | | CHAR | 8 | 73 | |
| COMMENT | <input type="checkbox"/> | COMMENT | | CHAR | 9 | 81 | |

21. 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 *cscvincSelectServiceRequest.si*. Also note that you can use their respective **Edit** buttons to return to the *Service Interface Definition* window for each interface.
22. The response container from program CSCVINC has a different layout from the request container so the Response service interface needs to be created.



23. Click the **Create Service Interface** button to create the response service interface required by this API and enter a *Service interface name* of *cscvincResponse*. Click **OK** to continue.
24. This will open a *Service Interface Editor* window.
25. The first step is to import the COBOL copy book (see below) that represents the outbound or response CONTAINER.

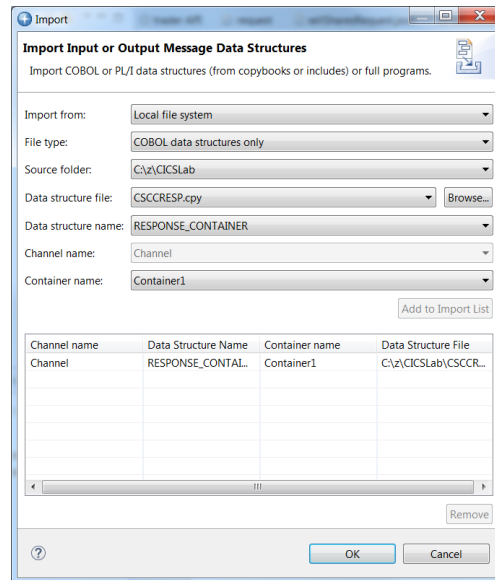
```

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 ADDR        PIC X(20) .
       05 PHONE       PIC X(8) .
       05 DATEX       PIC X(8) .
       05 AMOUNT      PIC X(8) .
       05 COMMENT     PIC X(9) .

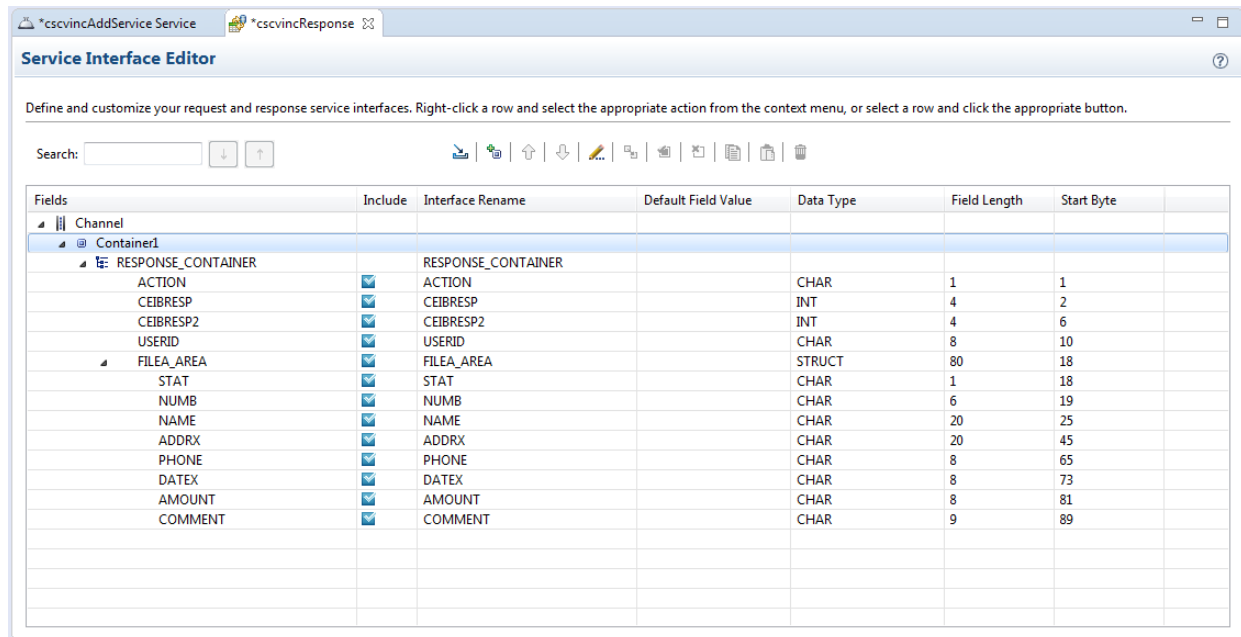
```

26. 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.

27. 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.



28. Click **OK** and when you expand *Container1*, *RESPONSE_CONTAINER* and *FILEA_AREA* you will see the COBOL ‘variables’ that have been imported into the service project.



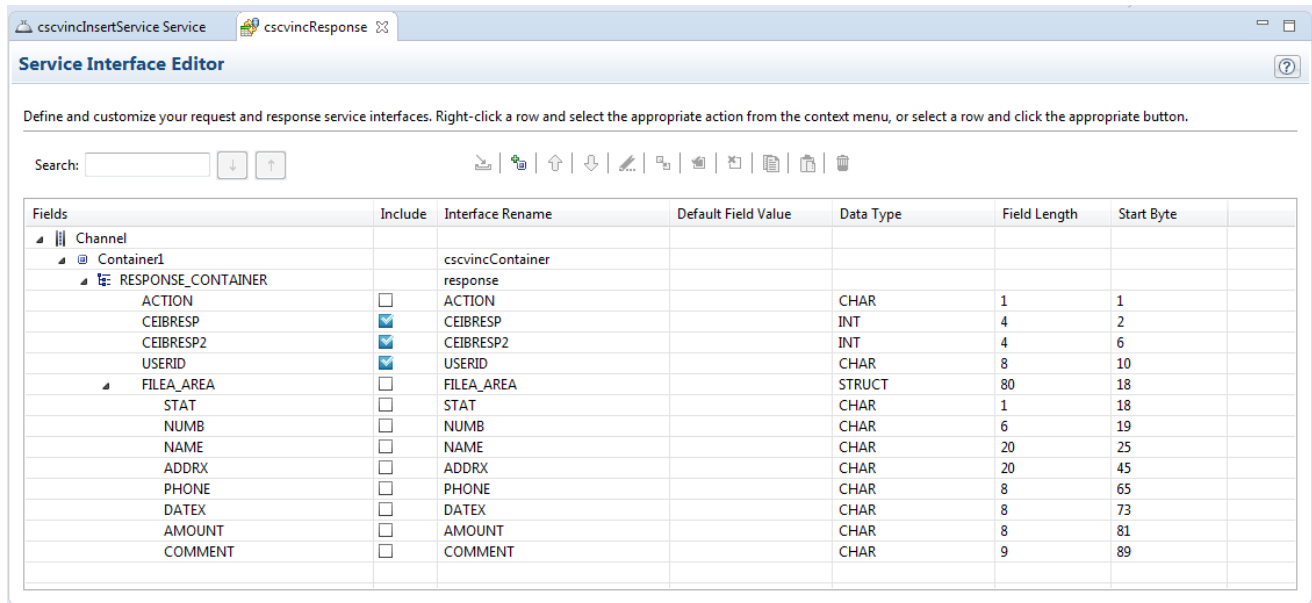
N.B. the interface names were derived from the COBOL source shown earlier

29. Again, in this view we could exclude fields from the interface or rename fields. Exclude all of the fields from the interface with the exception of *CEIBRESP*, *CEIBRESP2* and *USERID*. These fields will be referenced later.

30. Rename the interfaces for the other interface field names as shown below from:

- *Container1* to *cscvincContainer*
- *RESPONSE_CONTAINER* to *response*

31. When finished your service interface should look like this:



Service Interface Editor

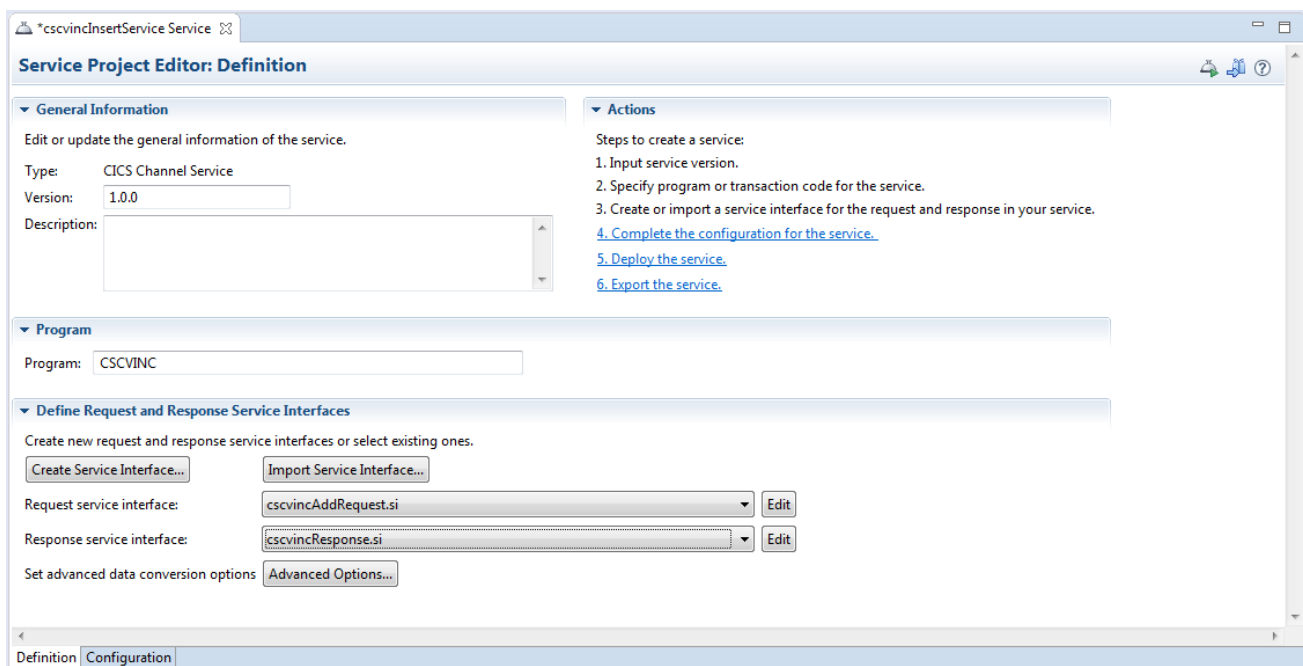
Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and click the appropriate button.

Search:

| Fields | Include | Interface Rename | Default Field Value | Data Type | Field Length | Start Byte |
|--------------------|-------------------------------------|------------------|---------------------|-----------|--------------|------------|
| Channel | | | | | | |
| Container1 | | cscvincContainer | | | | |
| RESPONSE_CONTAINER | | response | | | | |
| ACTION | <input type="checkbox"/> | ACTION | | CHAR | 1 | 1 |
| CEIBRESP | <input checked="" type="checkbox"/> | CEIBRESP | | INT | 4 | 2 |
| CEIBRESP2 | <input checked="" type="checkbox"/> | CEIBRESP2 | | INT | 4 | 6 |
| USERID | <input checked="" type="checkbox"/> | USERID | | CHAR | 8 | 10 |
| FILEA_AREA | <input type="checkbox"/> | FILEA_AREA | | STRUCT | 80 | 18 |
| STAT | <input type="checkbox"/> | STAT | | CHAR | 1 | 18 |
| NUMB | <input type="checkbox"/> | NUMB | | CHAR | 6 | 19 |
| NAME | <input type="checkbox"/> | NAME | | CHAR | 20 | 25 |
| ADDRX | <input type="checkbox"/> | ADDRX | | CHAR | 20 | 45 |
| PHONE | <input type="checkbox"/> | PHONE | | CHAR | 8 | 65 |
| DATEX | <input type="checkbox"/> | DATEX | | CHAR | 8 | 73 |
| AMOUNT | <input type="checkbox"/> | AMOUNT | | CHAR | 8 | 81 |
| COMMENT | <input type="checkbox"/> | COMMENT | | CHAR | 9 | 89 |

32. Close the *cscvincResponse* view by clicking on the white X.

33. Use the pull-down arrow beside *Response service interface* to select *cscvincResponse.si*.



Service Project Editor: Definition

General Information

Edit or update the general information of the service.

Type: CICS Channel Service

Version: 1.0.0

Description:

Actions

Steps to create a service:

1. Input service version.
2. Specify program or transaction code for the service.
3. Create or import a service interface for the request and response in your service.
4. [Complete the configuration for the service.](#)
5. [Deploy the service.](#)
6. [Export the service.](#)

Program

Program: CSCVINC

Define Request and Response Service Interfaces

Create new request and response service interfaces or select existing ones.

Create Service Interface... Import Service Interface...

Request service interface: cscvincAddRequest.si Edit

Response service interface: cscvincResponse.si Edit

Set advanced data conversion options Advanced Options...

Definition Configuration

34. 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*.

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Coded character set identifier (CCSID):

Connection reference:

Optional Configuration

Enter the optional configuration for this service.

Transaction ID:

Transaction ID usage:

Use context containers: ☐

Context containers HTTP headers:

Definition Configuration

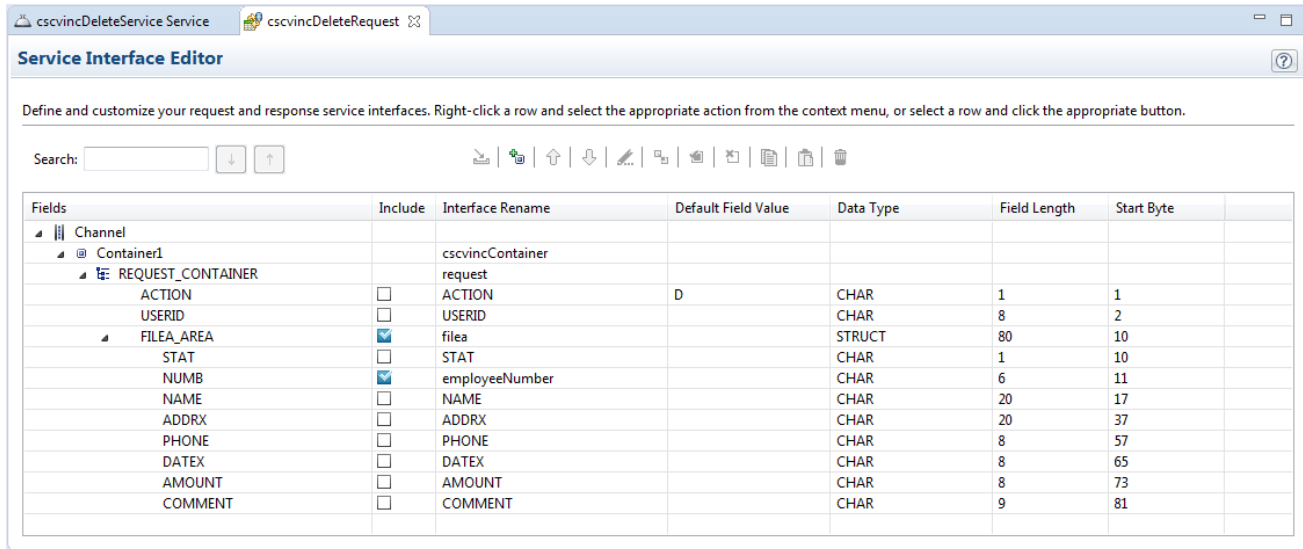
Tech-Tip: The *Connection reference* identifies the *cicsIpicConnection* element or *cicsLocalConnection* element to be used to access a CICS region in the z/OS Connect configuration, see *ipic.xml* on page 38. The case of the value of the *Connection reference* must match the case of the ID of the *cicsIpicConnection* element.

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

Create the “Delete” service

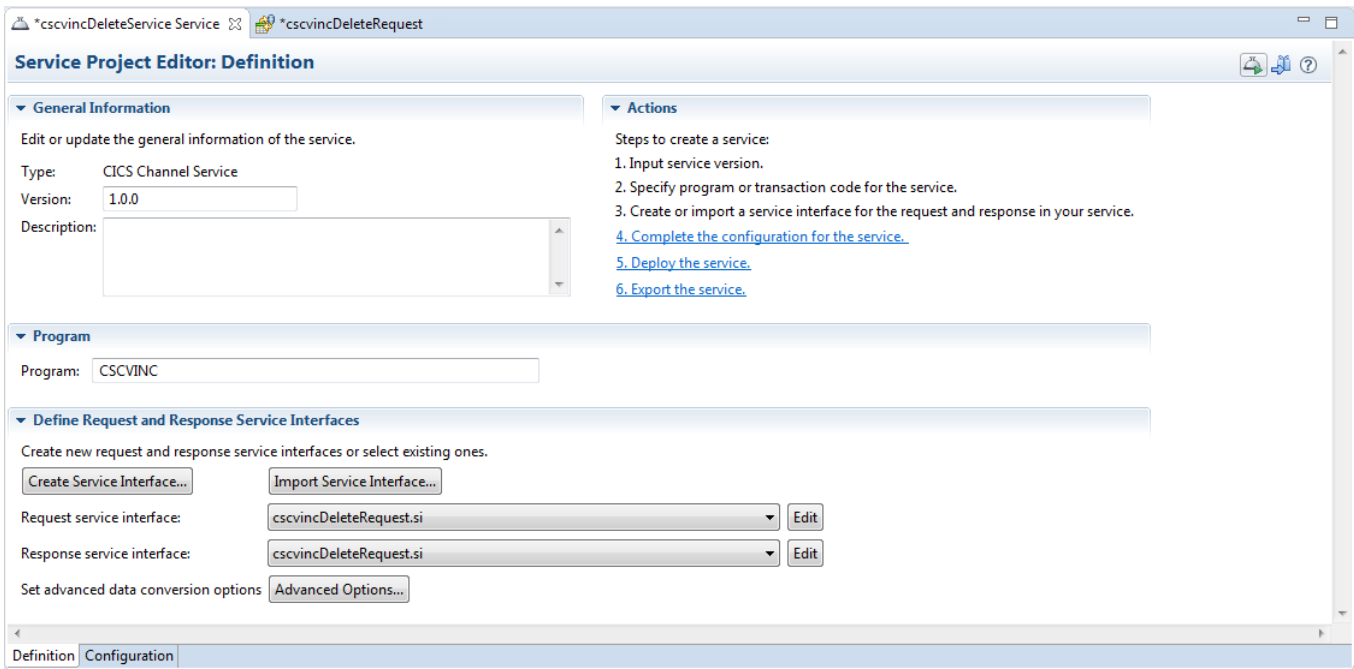
1. Use Steps 1 through 19 in the create insert service section to create service *cscvincDeleteService*. In this case the NUMB field is the only field exposed in the request message as *employeeNumber*. The action is set to *D* with this field omitted from the interface. Finally change the interface names of *Container1* to *cscvincContainer*, field *REQUEST_CONTAINER* to *request* and *FILEA_AREA* to *filea*.

- When finished the *cscvincDeleteRequest* service interface should look like:



| Fields | Include | Interface Rename | Default Field Value | Data Type | Field Length | Start Byte |
|-------------------|-------------------------------------|------------------|---------------------|-----------|--------------|------------|
| Channel | | cscvincContainer | | | | |
| Container1 | | | | | | |
| REQUEST_CONTAINER | | request | | | | |
| ACTION | <input type="checkbox"/> | ACTION | D | CHAR | 1 | 1 |
| USERID | <input type="checkbox"/> | USERID | | CHAR | 8 | 2 |
| FILEA_AREA | <input checked="" type="checkbox"/> | filea | | STRUCT | 80 | 10 |
| STAT | <input type="checkbox"/> | STAT | | CHAR | 1 | 10 |
| NUMB | <input checked="" type="checkbox"/> | employeeNumber | | CHAR | 6 | 11 |
| NAME | <input type="checkbox"/> | NAME | | CHAR | 20 | 17 |
| ADDRX | <input type="checkbox"/> | ADDRX | | CHAR | 20 | 37 |
| PHONE | <input type="checkbox"/> | PHONE | | CHAR | 8 | 57 |
| DATEX | <input type="checkbox"/> | DATEX | | CHAR | 8 | 65 |
| AMOUNT | <input type="checkbox"/> | AMOUNT | | CHAR | 8 | 73 |
| COMMENT | <input type="checkbox"/> | COMMENT | | CHAR | 9 | 81 |

- And the service definition for the *cscvincDeleteService* should look like:



Service Project Editor: Definition

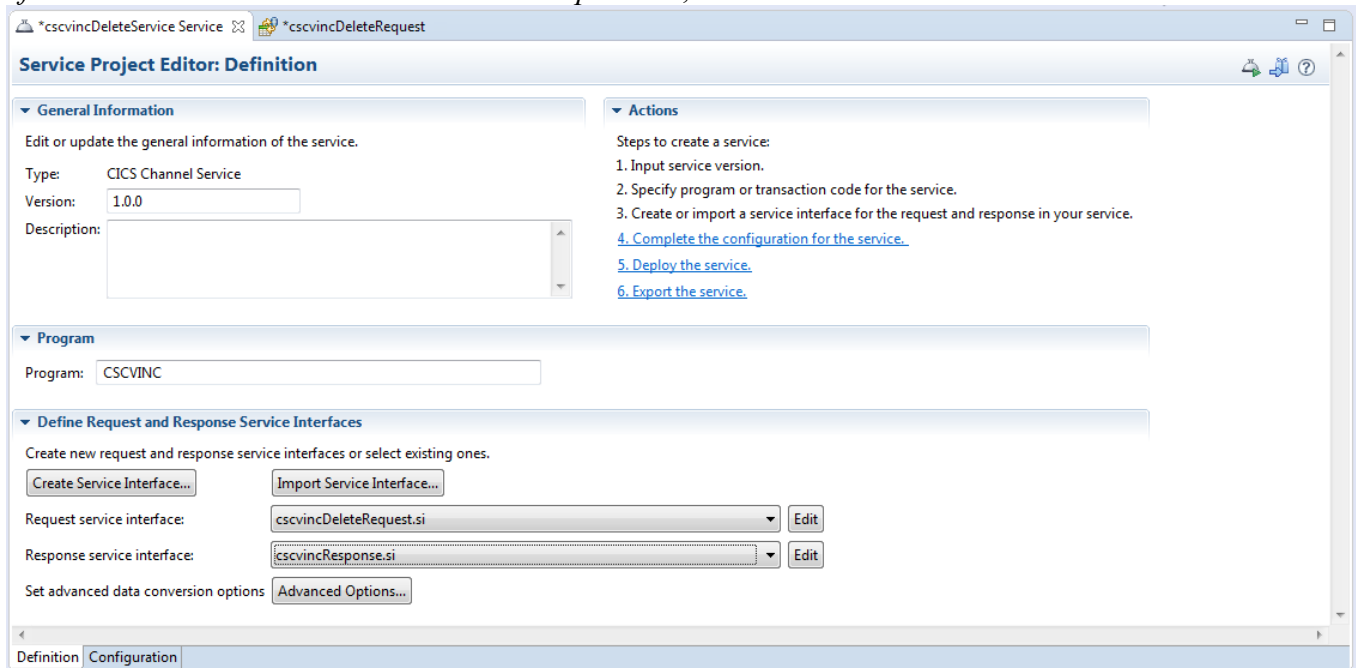
General Information
 Edit or update the general information of the service.
 Type: CICS Channel Service
 Version: 1.0.0
 Description:

Program
 Program: CSCVINC

Define Request and Response Service Interfaces
 Create new request and response service interfaces or select existing ones.
 Create Service Interface... Import Service Interface...
 Request service interface: cscvincDeleteRequest.si Edit
 Response service interface: cscvincDeleteRequest.si Edit
 Set advanced data conversion options Advanced Options...

Actions
 Steps to create a service:
 1. Input service version.
 2. Specify program or transaction code for the service.
 3. Create or import a service interface for the request and response in your service.
 4. Complete the configuration for the service.
 5. Deploy the service.
 6. Export the service.

2. On the *Service Project Editor: Definition* view click the **Import Service Interface** button. Click the **Workspace** button on the *Import a service interface* screen. On the *Import Service Interface – Import Service Interface* window expand the *cscvincInsertService* project and then expand *service-interfaces* and then select *cscvincResponse.si*. Click **OK** to import this service interface into this service project.
3. On the *Service Project Editor: Definition* view use the pull-down arrow beside the area for *Response service interface* and select service interface *cscvincResponse.si*, see below:



Tech-Tip: This has just reused the response service interface from the *cscvincInsertService* service.

4. Next, we need to identify a connection profile and interaction properties profile that 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*.

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

Create the “Update” Service

1. Use Steps 1 through 19 in the create insert service section to create service *cscvincUpdateService*. The action field is set to *U* and this field is omitted from the interface. Change the interface names of *Container1* to *cscvincContainer* and *REQUEST_CONTAINER* to *request*.

2. Rename the other interface names for the other interface names as shown below:

- *FILEA_AREA* to *filea*
- *NUMB* to *employeeNumber*
- *Name* to *name*
- *ADDRX* to *address*
- *PHONE* to *phoneNumber*
- *DATEX* to *date*
- *AMOUNT* to *amount*

N.B. if the interface names are not renamed, the original interface names will appear in subsequent screen shots.

When finished your service definition interface should look like this.

IBM z/OS Connect(OpenAPI 2.0)

Service Interface Editor

Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and click the appropriate button.

Search:

| Fields | Include | Interface Rename | Default Field Value | Data Type | Field Length | Start Byte |
|-------------------|-------------------------------------|------------------|---------------------|-----------|--------------|------------|
| Channel | | | | | | |
| Container1 | | cscvincContainer | | | | |
| REQUEST_CONTAINER | | request | | | | |
| ACTION | <input checked="" type="checkbox"/> | ACTION | U | CHAR | 1 | 1 |
| USERID | <input type="checkbox"/> | USERID | | CHAR | 8 | 2 |
| FILE_AREA | <input checked="" type="checkbox"/> | filea | | STRUCT | 80 | 10 |
| STAT | <input type="checkbox"/> | STAT | | CHAR | 1 | 10 |
| NUMB | <input checked="" type="checkbox"/> | employeeNumber | | CHAR | 6 | 11 |
| NAME | <input checked="" type="checkbox"/> | name | | CHAR | 20 | 17 |
| ADDRX | <input checked="" type="checkbox"/> | address | | CHAR | 20 | 37 |
| PHONE | <input checked="" type="checkbox"/> | phoneNumber | | CHAR | 8 | 57 |
| DATEX | <input checked="" type="checkbox"/> | date | | CHAR | 8 | 65 |
| AMOUNT | <input checked="" type="checkbox"/> | amount | | CHAR | 8 | 73 |
| COMMENT | <input type="checkbox"/> | COMMENT | | CHAR | 9 | 81 |

- When finished the service definition for the *cscvincUpdateService* should look like this:

***cscvincUpdateService Service**

General Information

Edit or update the general information of the service.

Type: CICS Channel Service

Version: 1.0.0

Description:

Program

Program: CSCVINC

Define Request and Response Service Interfaces

Create new request and response service interfaces or select existing ones.

Create Service Interface... Import Service Interface...

Request service interface: cscvincUpdateService.si Edit

Response service interface: cscvincUpdateService.si Edit

Set advanced data conversion options Advanced Options...

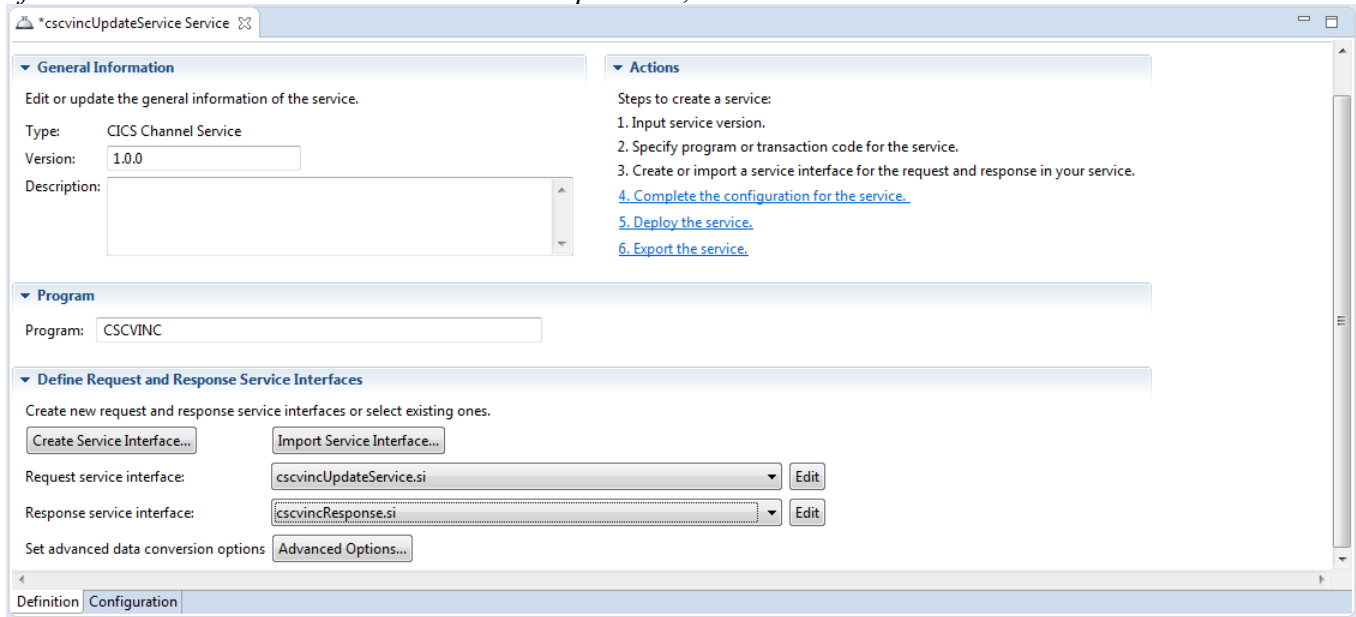
Definition Configuration

Actions

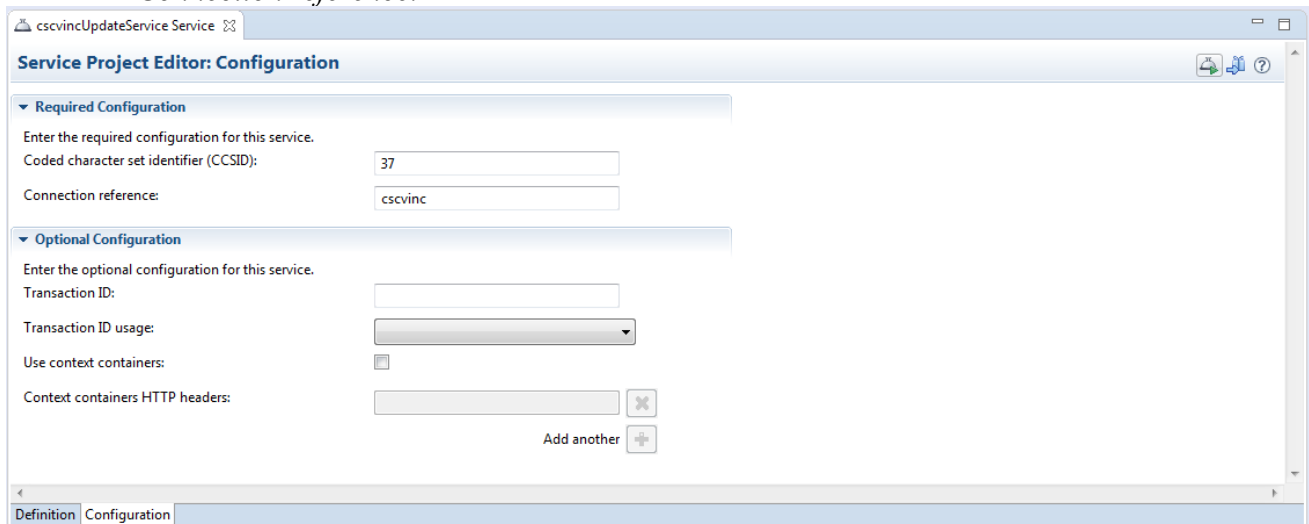
Steps to create a service:

1. Input service version.
2. Specify program or transaction code for the service.
3. Create or import a service interface for the request and response in your service.
4. [Complete the configuration for the service.](#)
5. [Deploy the service.](#)
6. [Export the service.](#)

3. On the *Service Project Editor: Definition* view click the **Import Service Interface** button. Click the **Workspace** button on the *Import a service interface* screen. On the *Import Service Interface – Import Service Interface* window expand the *cscvincInsertService* project and then expand *service-interfaces* and then select *cscvincResponse.si*. Click **OK** to import this service interface into this service project.
4. On the *Service Project Editor: Definition* view use the pull-down arrow beside the area for *Response service interface* and select service interface *cscvincResponse.si*, see below:



5. Next, we need to identify a connection profile and interaction properties profile that 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*.

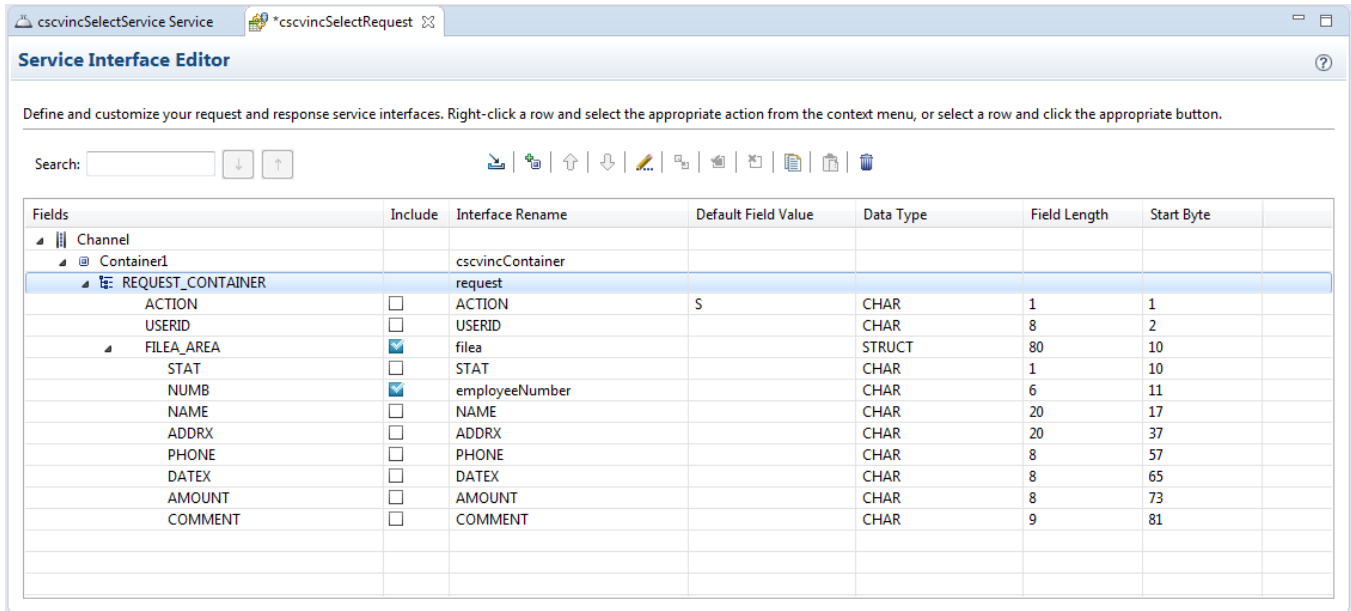


6. Save the *cscvincUpdateService* service either by closing the tab or using the **Ctrl-S** key sequence.

Create the “Select” Service

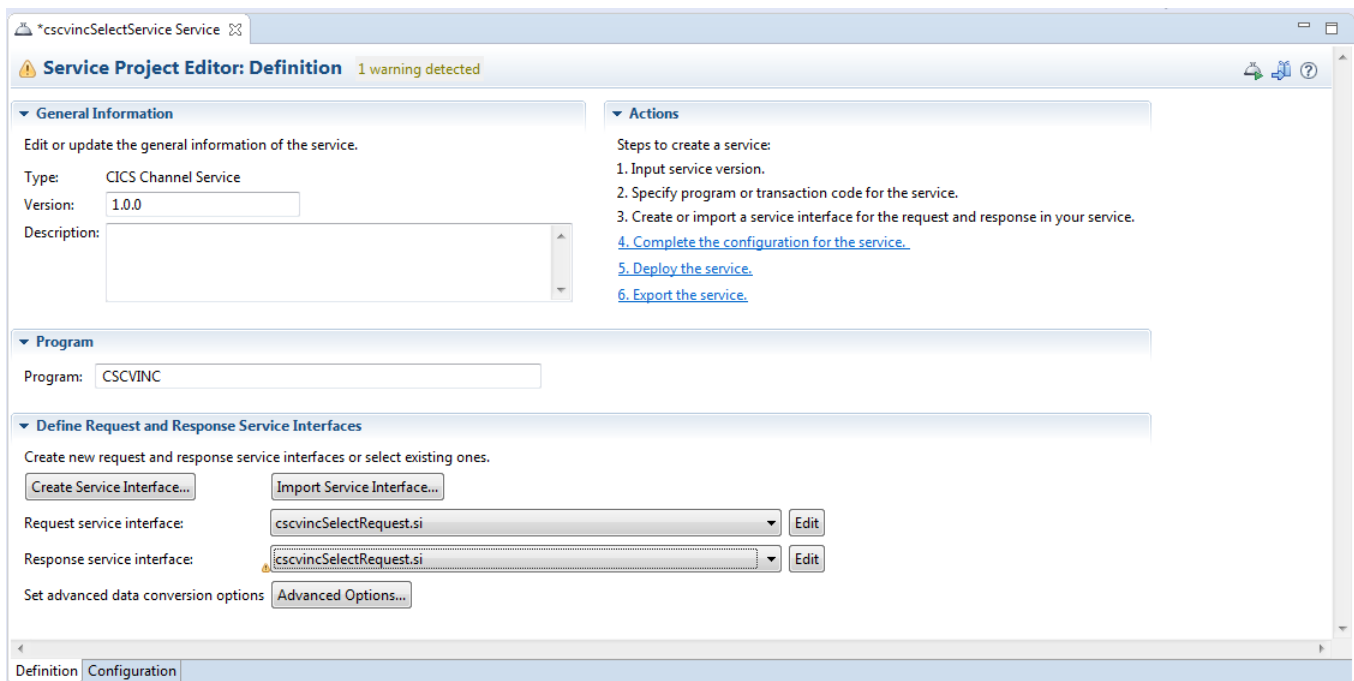
7. Use Steps 1 through 19 to create service *cscvincSelectService*. In the case NUMB will be the only field exposed in the request message. The action is set to *S* with this field omitted from the interface. Finally change the interface name of *Container1* to *cscvincContainer*, *REQUEST_CONTAINER* to *request*, *FILEA_AREA* to *filea* and *NUMB* to *employeeNumber*.

- When finished the *cscvincSelectRequest* service interface should look like this:



| Fields | Include | Interface Rename | Default Field Value | Data Type | Field Length | Start Byte |
|-------------------|-------------------------------------|------------------|---------------------|-----------|--------------|------------|
| Channel | | | | | | |
| Container1 | | cscvincContainer | | | | |
| REQUEST_CONTAINER | | request | | | | |
| ACTION | <input type="checkbox"/> | ACTION | S | CHAR | 1 | 1 |
| USERID | <input type="checkbox"/> | USERID | | CHAR | 8 | 2 |
| FILEA_AREA | <input checked="" type="checkbox"/> | filea | | STRUCT | 80 | 10 |
| STAT | <input type="checkbox"/> | STAT | | CHAR | 1 | 10 |
| NUMB | <input checked="" type="checkbox"/> | employeeNumber | | CHAR | 6 | 11 |
| NAME | <input type="checkbox"/> | NAME | | CHAR | 20 | 17 |
| ADDRX | <input type="checkbox"/> | ADDRX | | CHAR | 20 | 37 |
| PHONE | <input type="checkbox"/> | PHONE | | CHAR | 8 | 57 |
| DATEX | <input type="checkbox"/> | DATEX | | CHAR | 8 | 65 |
| AMOUNT | <input type="checkbox"/> | AMOUNT | | CHAR | 8 | 73 |
| COMMENT | <input type="checkbox"/> | COMMENT | | CHAR | 9 | 81 |

- When finished the service definition for the *cscvincSelectService* should look like this:



Service Project Editor: Definition 1 warning detected

General Information
 Edit or update the general information of the service.
 Type: CICS Channel Service
 Version: 1.0.0
 Description:

Actions
 Steps to create a service:
 1. Input service version.
 2. Specify program or transaction code for the service.
 3. Create or import a service interface for the request and response in your service.
[4. Complete the configuration for the service.](#)
[5. Deploy the service.](#)
[6. Export the service.](#)

Program
 Program: CSCVINC

Define Request and Response Service Interfaces
 Create new request and response service interfaces or select existing ones.
 Create Service Interface... Import Service Interface...
 Request service interface: cscvincSelectRequest.si Edit
 Response service interface: cscvincSelectRequest.si Edit
 Set advanced data conversion options Advanced Options...

Definition Configuration

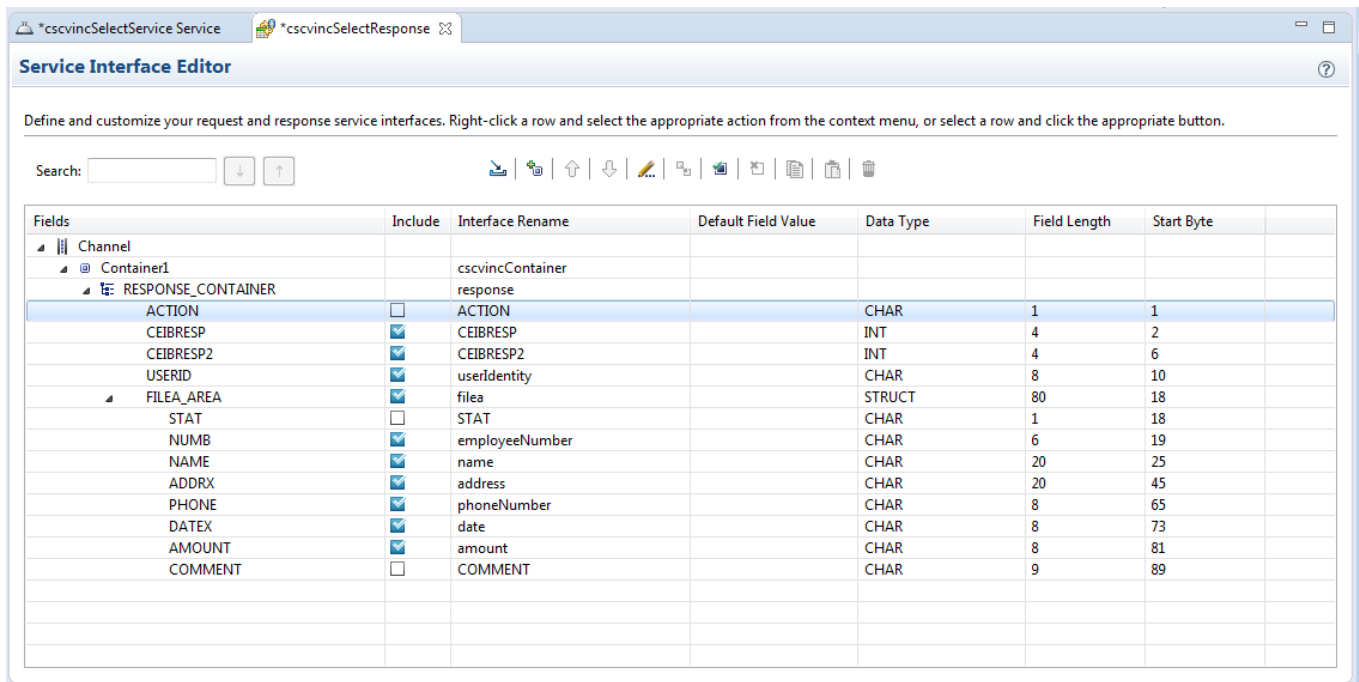
8. Again, the default response service interface is not we want so we need to create another service interface named *cscvincSelectResponse* that will include the contact information in the response message. Use the *Create Service Interface* button and create a new service interfaced named *cscvincSelectResponse*.

9. Change the interface names of *Container1* to *cscvincContainer* and field *REQUEST_CONTAINER* to *request*.

10. Rename the other interface names for the other interface names as shown below:

- *FILEA_AREA* to *filea*
- *NAME* to *name*
- *NUMB* to *employeeNumber*
- *ADDRX* to *address*
- *PHONE* to *phoneNumber*
- *DATEX* to *date*
- *AMOUNT* to *amount*

N.B. if the interface names are not renamed, the original interface names will appear in subsequent screen shots.



| Fields | Include | Interface Rename | Default Field Value | Data Type | Field Length | Start Byte |
|--------------------|-------------------------------------|------------------|---------------------|-----------|--------------|------------|
| Channel | | | | | | |
| Container1 | | cscvincContainer | | | | |
| RESPONSE_CONTAINER | | response | | | | |
| ACTION | <input type="checkbox"/> | ACTION | | CHAR | 1 | 1 |
| CEIBRESP | <input checked="" type="checkbox"/> | CEIBRESP | | INT | 4 | 2 |
| CEIBRESP2 | <input checked="" type="checkbox"/> | CEIBRESP2 | | INT | 4 | 6 |
| USERID | <input checked="" type="checkbox"/> | userIdentity | | CHAR | 8 | 10 |
| FILEA_AREA | <input checked="" type="checkbox"/> | filea | | STRUCT | 80 | 18 |
| STAT | <input type="checkbox"/> | STAT | | CHAR | 1 | 18 |
| NUMB | <input checked="" type="checkbox"/> | employeeNumber | | CHAR | 6 | 19 |
| NAME | <input checked="" type="checkbox"/> | name | | CHAR | 20 | 25 |
| ADDRX | <input checked="" type="checkbox"/> | address | | CHAR | 20 | 45 |
| PHONE | <input checked="" type="checkbox"/> | phoneNumber | | CHAR | 8 | 65 |
| DATEX | <input checked="" type="checkbox"/> | date | | CHAR | 8 | 73 |
| AMOUNT | <input checked="" type="checkbox"/> | amount | | CHAR | 8 | 81 |
| COMMENT | <input type="checkbox"/> | COMMENT | | CHAR | 9 | 89 |

11. Close the *Service Interface Definition* window.

12. Set the *Response service interface* to *cscvincSelectResponse.si* as shown below.

Service Project Editor: Definition

General Information
 Edit or update the general information of the service.
 Type: CICS Channel Service
 Version: 1.0.0
 Description:

Program
 Program: CSCVINC

Define Request and Response Service Interfaces
 Create new request and response service interfaces or select existing ones.
 Create Service Interface... Import Service Interface...
 Request service interface: cscvincSelectRequest.si Edit
 Response service interface: cscvincSelectResponse.si Edit
 Set advanced data conversion options Advanced Options...

Actions
 Steps to create a service:
 1. Input service version.
 2. Specify program or transaction code for the service.
 3. Create or import a service interface for the request and response in your service.
 4. Complete the configuration for the service.
 5. Deploy the service.
 6. Export the service.

Definition Configuration

13. Next, we need to identify a connection profile and interaction properties profile that 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*.

Service Project Editor: Configuration

Required Configuration
 Enter the required configuration for this service.
 Coded character set identifier (CCSID): 37
 Connection reference: cscvinc

Optional Configuration
 Enter the optional configuration for this service.
 Transaction ID:
 Transaction ID usage:
 Use context containers:
 Context containers HTTP headers:
 Add another

Definition Configuration

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

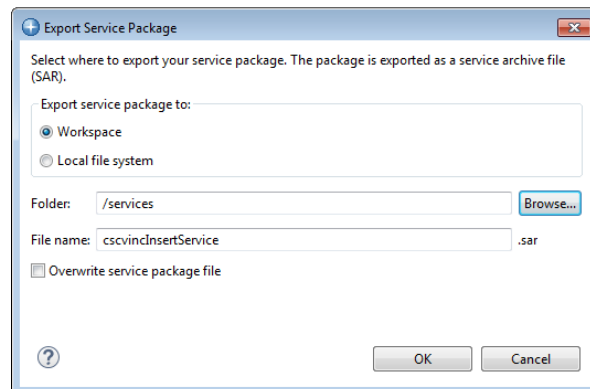
This services now needs to be made available for developing the API and for deployment to the z/OS Connect server.

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

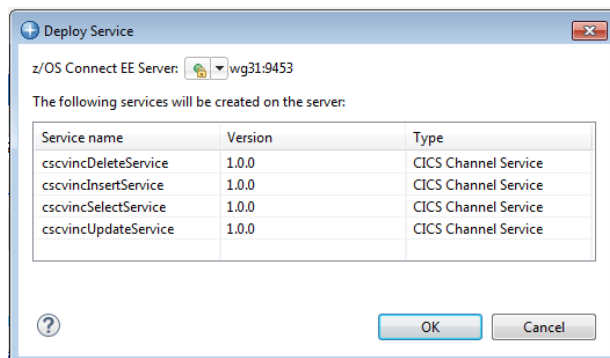
Export and deploy the Service Archive files

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 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 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 *cscvincInsertService* service project and right mouse button click. On the pop-up selection select **z/OS Connect → Export z/OS Connect 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.

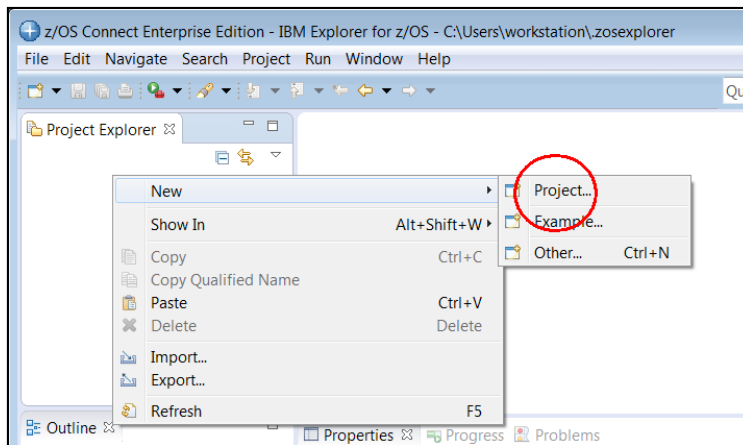


4. Repeat this step for the other 3 services.
5. Select all 4 service projects and right mouse button click again and on the pop-up selection select **z/OS Connect → Deploy Service to z/OS Connect Server**. On the *Deploy Service* window select the target server (*wg31:9453*) and click **OK** twice to continue.

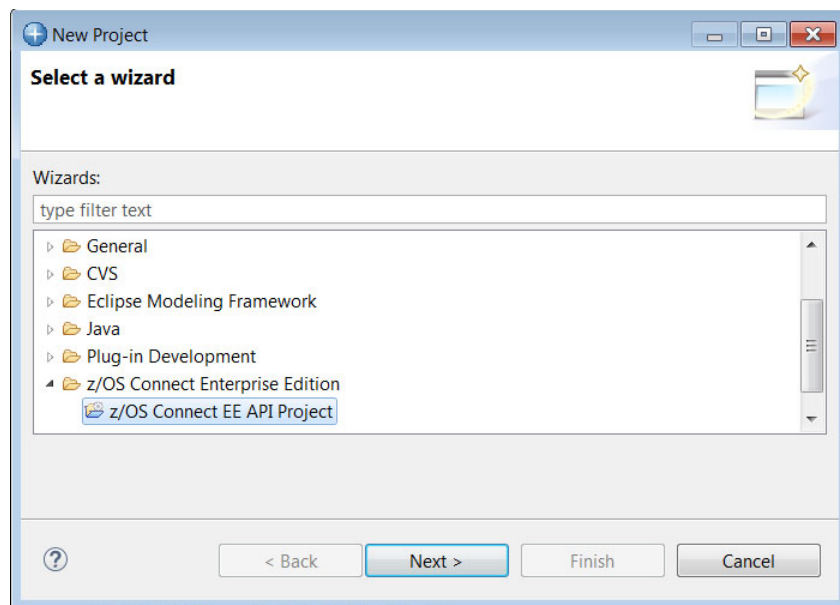


Create the CscvincAPI API project

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



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

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

Project name: cscvincAPI
API name: cscvincAPI
Base path: /cscvinc
Description:

< Back Next > Finish Cancel

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.

API Editor

Describe your API

Name: cscvincAPI Description:

Base path: /cscvinc

Version: 1.0.0

Path

/newPath1

Methods (4)

| Method | Service... | Mapping... |
|--------|------------|------------|
| POST | | |
| GET | | |
| PUT | | |
| DELETE | | |

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

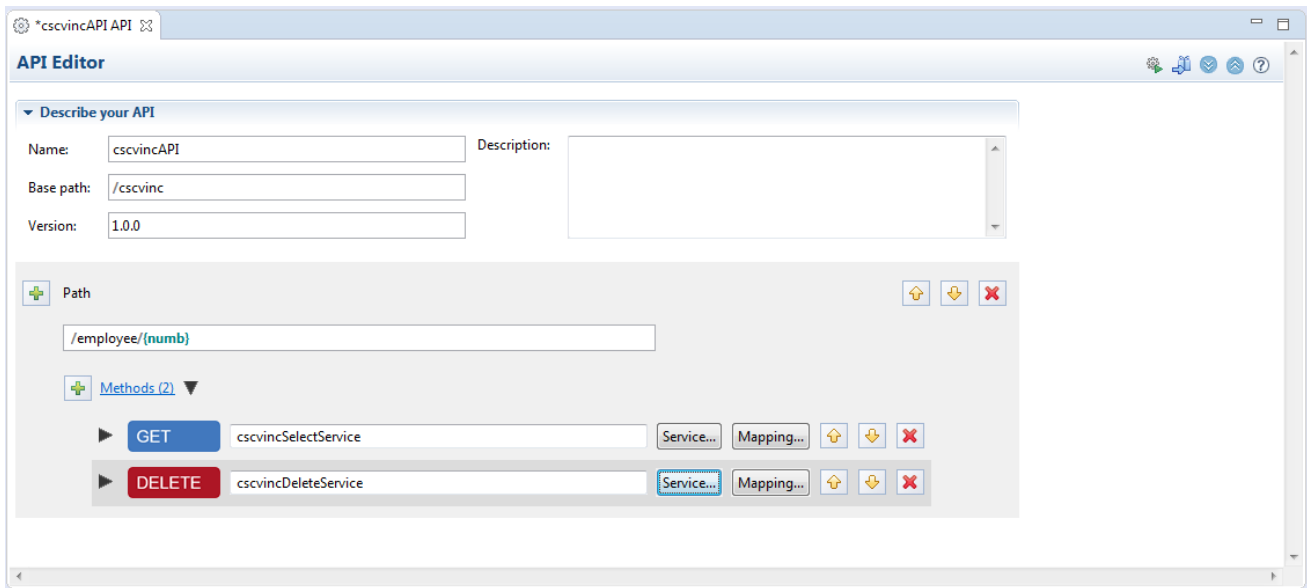
Summary

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

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.

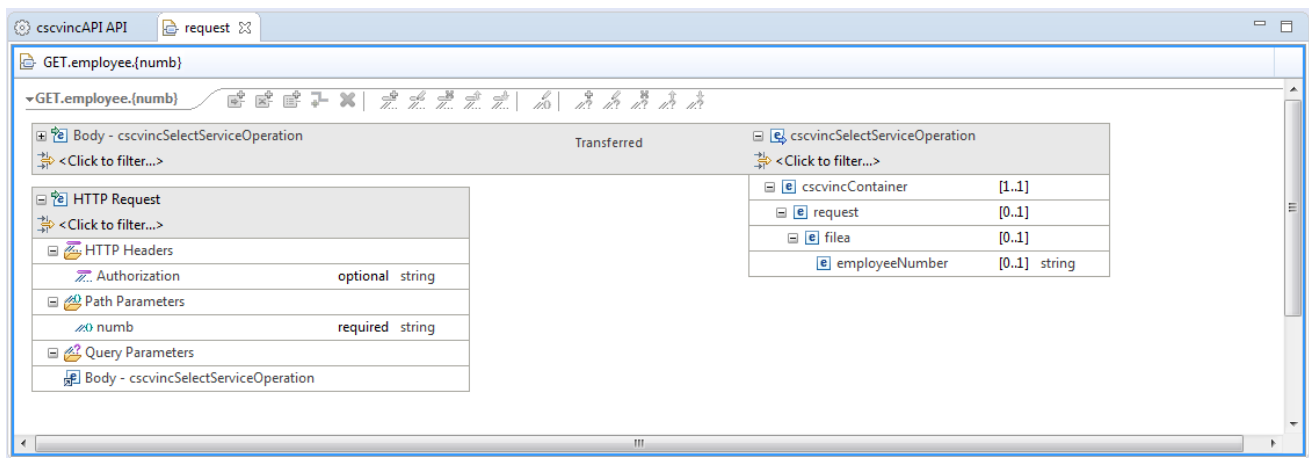
1. Start by entering a Path of **/employee/{numb}** in the z/OS Connect API Editor view. Remove the *POST* and *PUT* methods by the red X's next to each.
2. Click the **Service** button beside the *GET* method and on the *Select a z/OS Connect Service* window click the **Workspace** button. On the *Import z/OS Connect Services* window expand the *services* folder and select *cscvincDeleteService.sar*, *cscvincInsertService.sar*, *cscvincSelectService* and *cscvincUpdateService.sar* service archive files. Click OK three times to import these SAR files into this project.
3. Click the Service button beside the *GET* method and on the *Select a z/OS Connect Service* window select *cscvincSelectService.sar* and click **OK**.
4. Click the **Service** button beside the *Delete* method and on the *Select a z/OS Connect Service* window select *cscvincDeleteService.sar* and click **OK**.



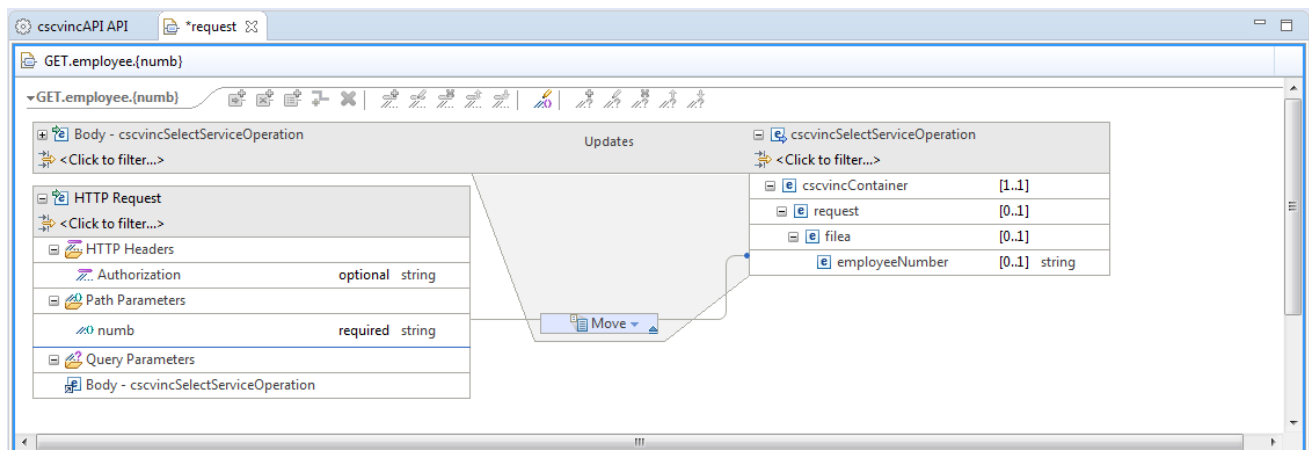
5. Next, click on the **Mapping** button beside the **GET** method and then select *Open Request Mapping*:



6. In the mapping view that opens, go to the right side of the mapping (which represents interface fields included in the service by the service developer), and click the little + signs to expand *cscvincContainer*, *request* and *filea*. You should see only the *employeeNumber* field.

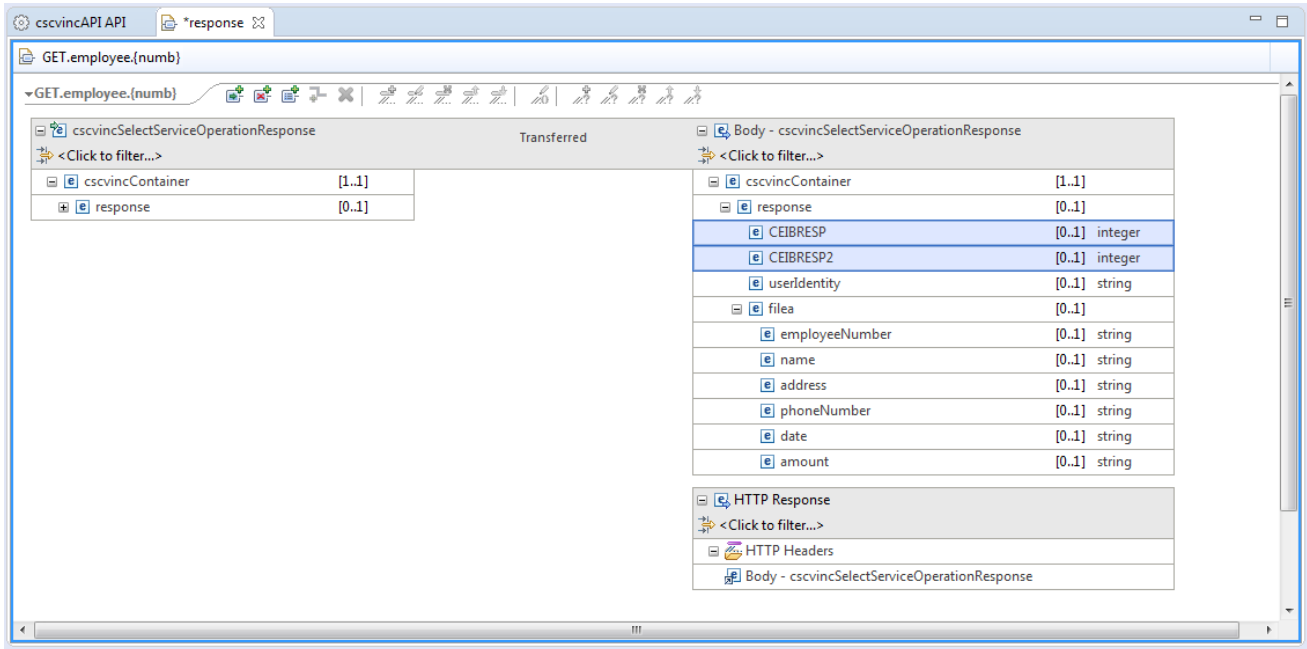


7. The contents of the *employeeNumber* field should be provided by the *numb* path parameter from the URL. On the left-hand side of the view select the *numb* path parameter with the left mouse button and drag it without releasing the mouse button) to the *employeeNumber* field on the right-hand side to create a “move” connection from the *numb* path parameter to the *employeeNumber* field in the *cscvincContainer* structure.

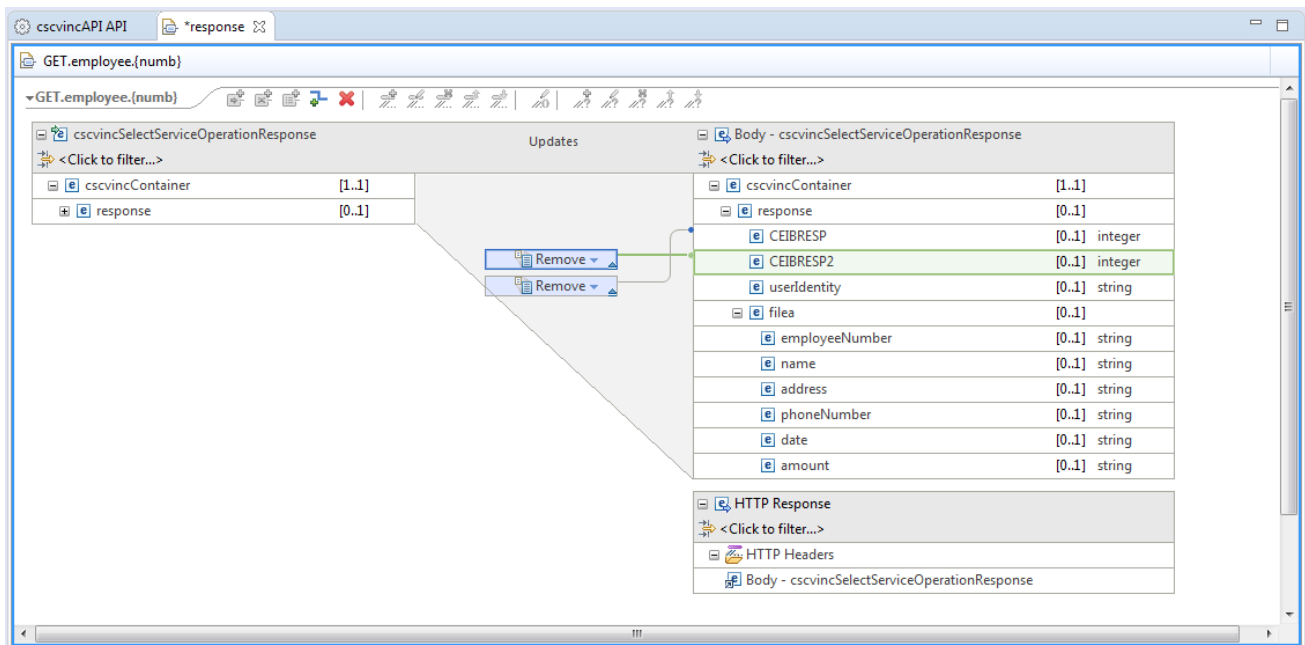


8. Use the **Ctrl-S** key sequence to save all changes and close the *GET.employee.{numb}* view.

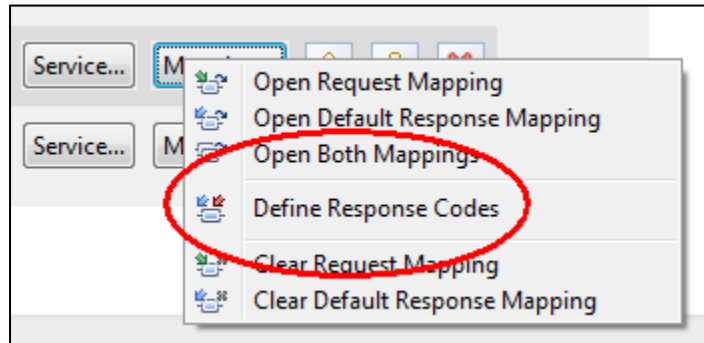
9. For **GET** the method we want the default response mapping to return all fields except *CEIBRESP* and *CEIBRESP2*. To change/review the fields that will be returned click the **Mapping** button beside the **GET** method and select the *Open Default Response Mapping* option.
10. Fully expand *cscvincContainer* and use the slider bar to fully expose the *cscvincContainer* structure. Use the left mouse button and draw a dotted line box that fully includes the *CEIBRESP* and *CEIBRESP2* fields. When you release the button, these fields should be selected (the background should be blue).



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



12. This action generates multiple “remove” requests (see above) for the selected fields. These fields will not be exposed to the REST clients using this method.
13. Use the **Ctrl-S** key sequence to save all changes and close the *GET.employee.{numb}* view
14. Click the **Mapping** button again but this time select *Define Response Codes*.



This will allow the setting of the HTTP response code based on the contents of the response message.

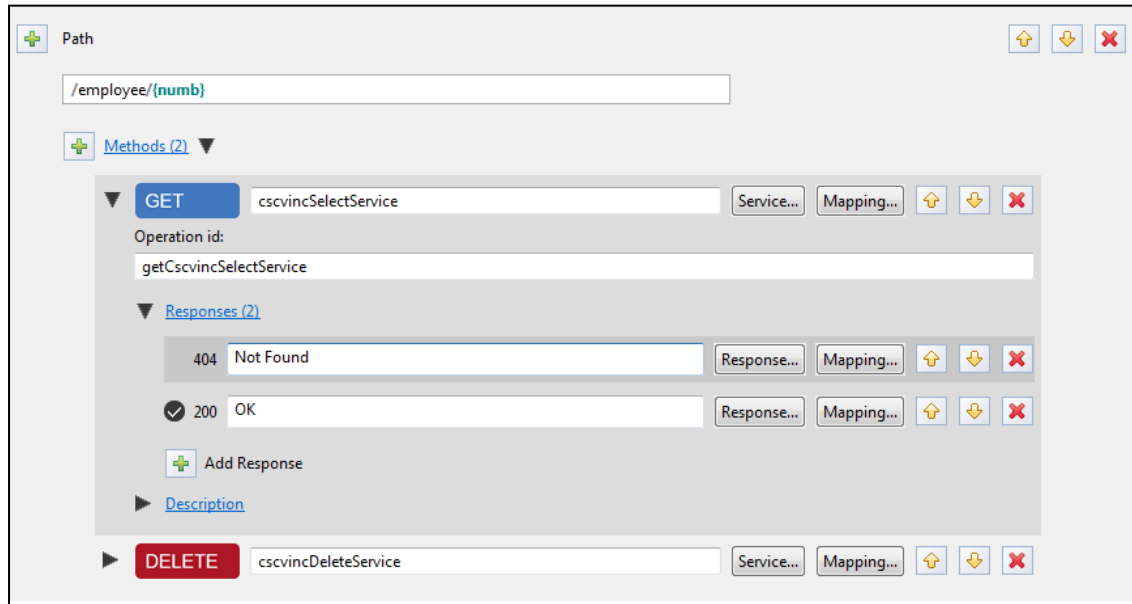
15. Click the plus sign beside *Add Response* and use the pull-down arrows to select a *Response code* of *404 – Not Found*, use the pull-down to select field *cscvincSelectServiceOperationResponse/cscvincContainer/response/CEIBRESP* for the first operand, an equal sign for the operation and enter *13* (NOTFND) for the second operand.

A screenshot of the 'Add Response' dialog box. The 'Response code' is set to '404 - Not Found' and the 'Description' is 'Not Found'. Below this, there is a section titled 'Define rules that indicate whether to use this response code and apply its response mapping, if defined.' Under this section, 'Rule1' is defined with the field 'cscvincSelectServiceOperationResponse/cscvincC' followed by an equals sign and the value '13'. There are buttons for 'Add Rule', 'OK', and 'Cancel'. A 'Summary' section at the bottom shows 'Rule1'.

Important: The CICS program returns the response container the EIBRESP and EIBRESP2 values from the EXEC CICS commands in the program. For other possible EIB values see the *CICS Knowledge Center* at URL

https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.5.0/reference/commands-api/dfhp4_eibfields.html

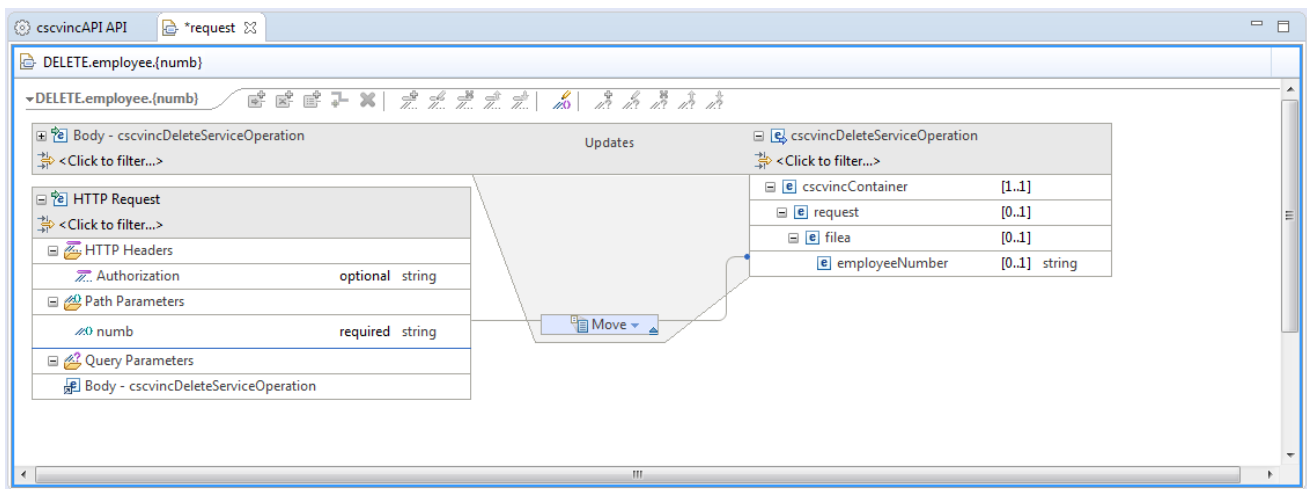
16. Click **OK** and now you should see



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

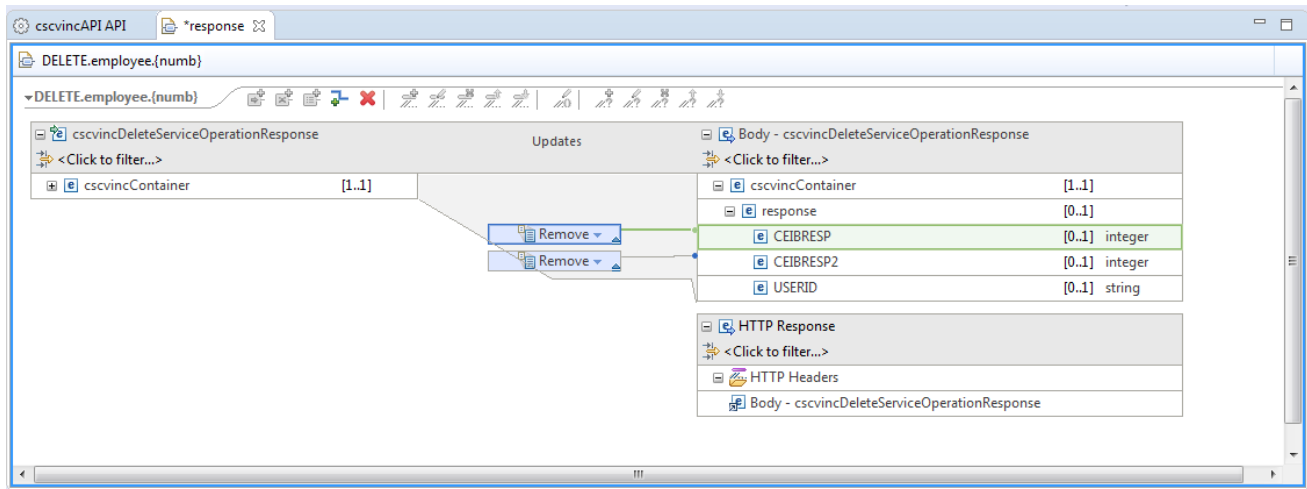


18. Use the drag and drop as before and move the contents of the part parameter *numb* to the *employeeNumber* field in the request.



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

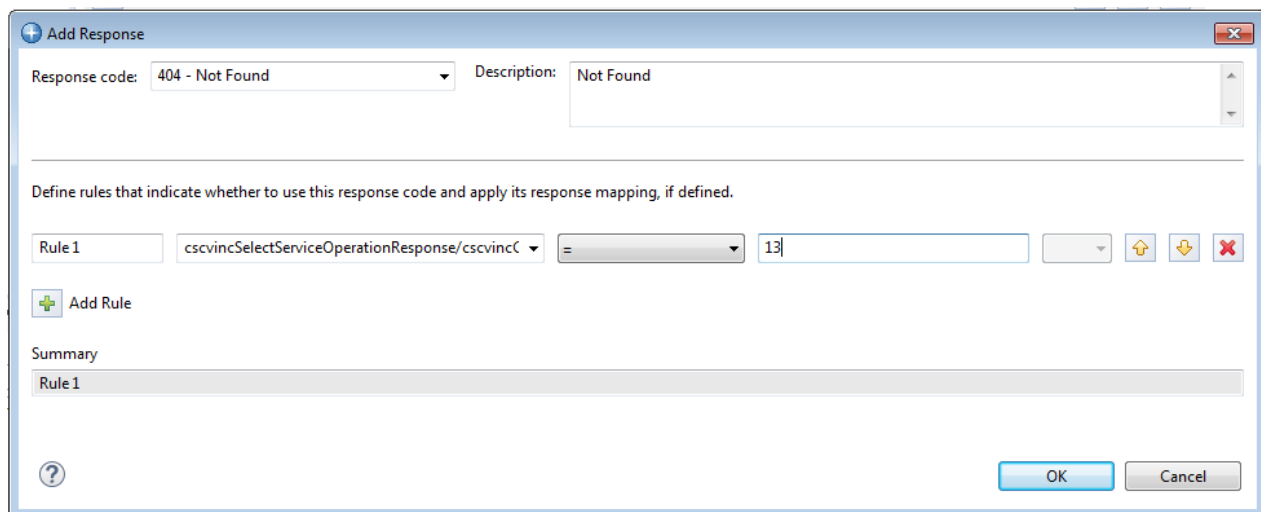
20. For the **DELETE** we want the default response mapping to return only the USERID field to the client. To change/review the fields that will be returned click the **Mapping** button beside the **DELETE** method and select the *Open Default Response Mapping* option.
21. Fully expand *cscvincContainer* and use the slider bar to fully expose the *cscvincContainer* structure. Use the left mouse button and draw a dotted line box that fully includes the *CEIBRESP* and *CEIBRESP2* fields. When you release the button, these fields should be selected (the background should be blue).
22. Right mouse button click on any of the selected fields and select the *Add Remove transform* from the list of options



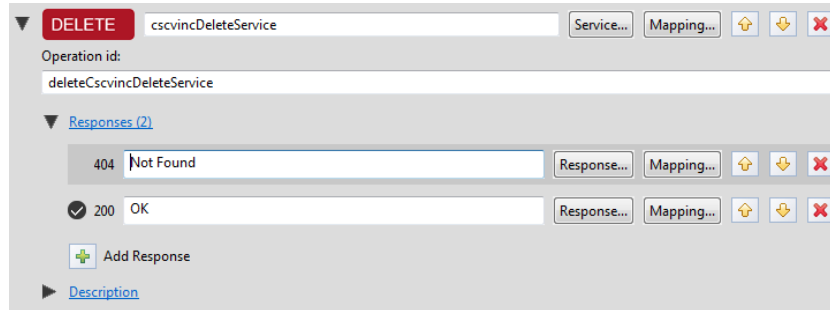
23. This action generates multiple “remove” requests (see above) for the selected fields. These fields will not be exposed to the REST clients using this method.
24. Use the **Ctrl-S** key sequence to save all changes and close the *DELETE.employee.{numb}* view.
25. Click the **Mapping** button again but this time select *Define Response Codes*.

This will allow the setting of the HTTP response code based on the contents of the response message.

26. Click the plus sign beside *Add Response* and use the pull-down arrows to select a *Response code* of *404 – Not Found*, use the pull-down to select field *cscvincSelectServiceOperationResponse/cscvincContainer/response/CEIBRESP* for the first operand, an equal sign for the operation and enter *13* (NOTFND) for the second operand.



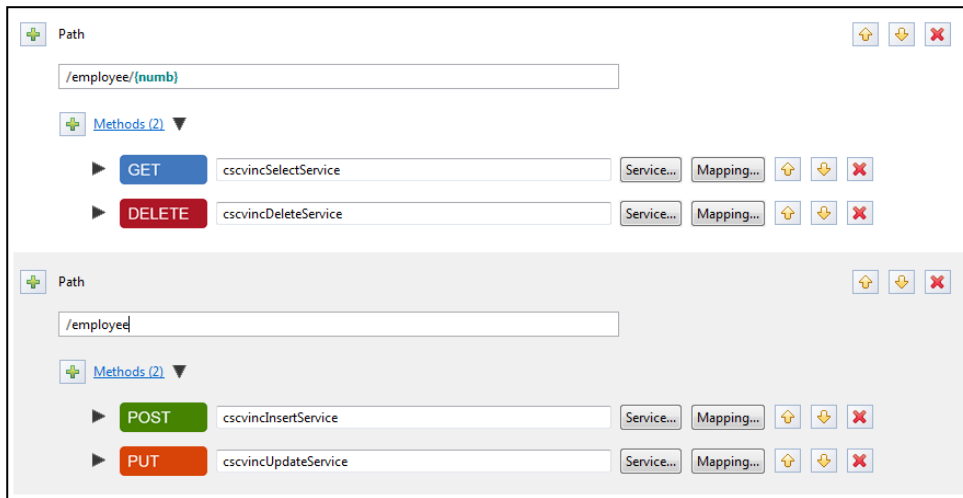
27. Click **OK** and now you should see



28. 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 API Editor* view. Remove the **DELETE** and **GET** methods by the red X's next to each.

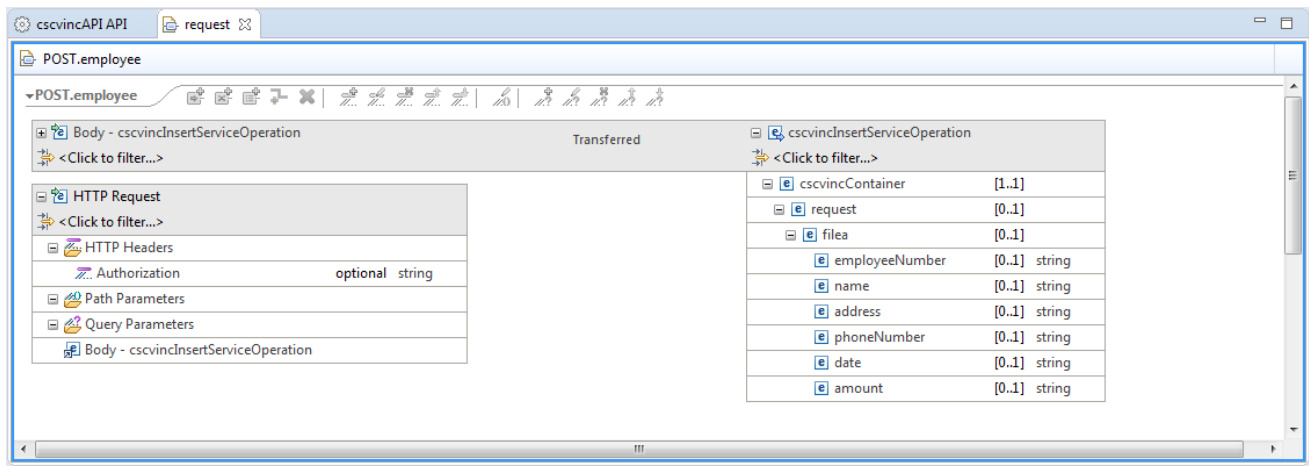
29. Click the **Service** button beside the **POST** method and on the *Select a z/OS Connect Service* window select *cscvincInsertService* and click **OK**

30. Click the **Service** button beside the **PUT** method and on the *Select a z/OS Connect Service* window select *cscvincUpdateService* and click **OK**.

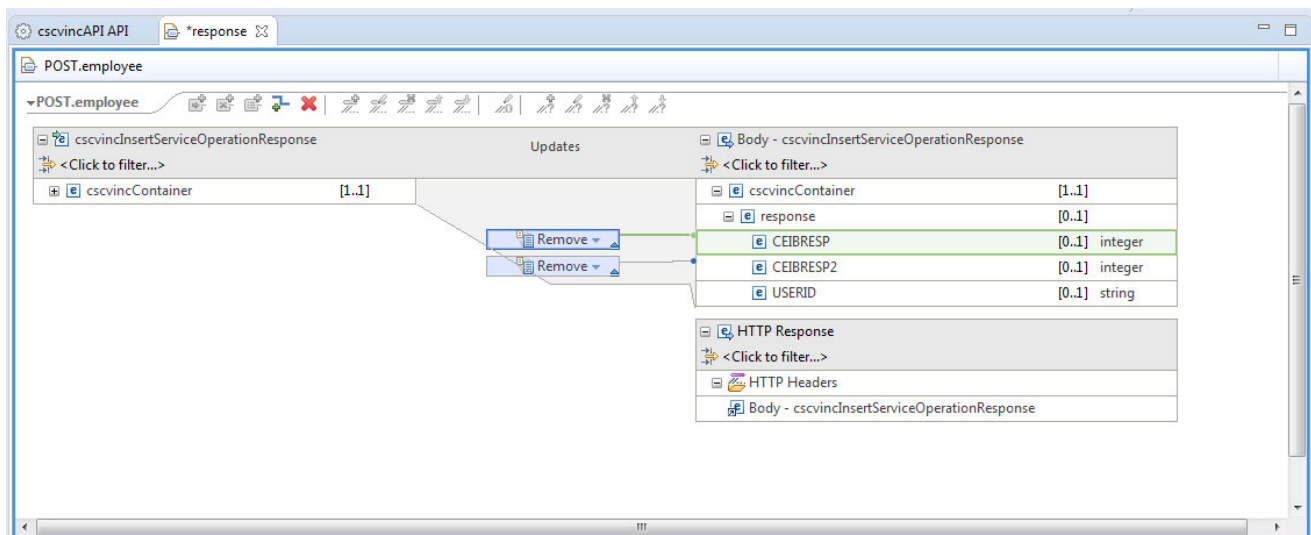


31. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping*:

32. In the mapping view that opens, go to the right side of the mapping (which represents the fields included in the interface by the services developer), and click the little + signs to expand *cscvincContainer*.

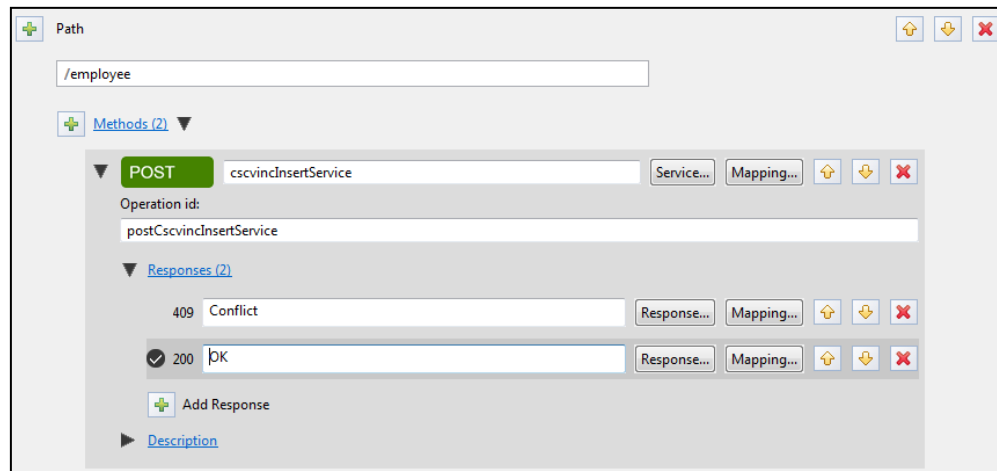


33. Use the **Ctrl-S** key sequence to save all changes and close the *POST.employee* view.
34. For the **POST** method we want the default response mapping to return only the *USERID* field to the client. To change/review the fields that will be returned click the **Mapping** button beside the **POST** method and select the *Open Default Response Mapping* option.
35. Fully expand *cscvincContainer* and use the slider bar to fully expose the *cscvincContainer* structure. Use the left mouse button and draw a dotted line box that fully includes the *CEIBRESP* and *CEIBRESP2* fields. When you release the button, these fields should be selected (the background should be blue).
36. Right mouse button click on any of the selected fields and select the *Add Remove transform* from the list of options



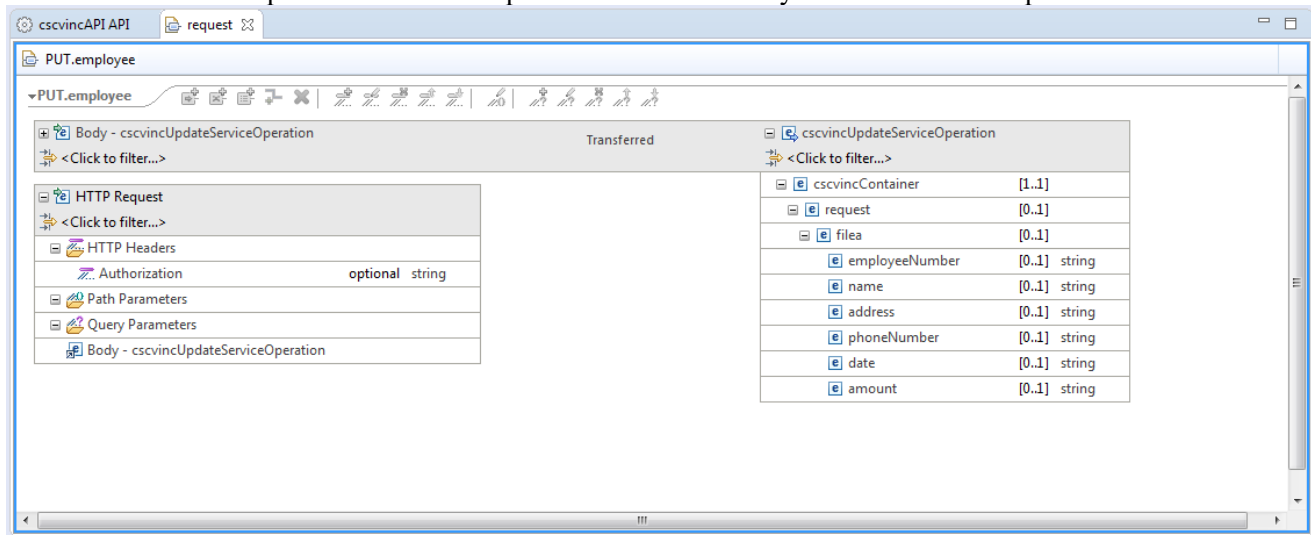
37. Use the **Ctrl-S** key sequence to save all changes and close the *POST.employee* view.
38. Click the **Mapping** button again but this time select *Define Response Codes*.

39. Click the plus sign beside *Add Response* and use the pull-down arrows to select a *Response code* of *409 – Conflict*, use the pull-down to select field *cscvincSelectServiceOperationResponse/cscvincContainer/response/CEIBRESP* for the first operand, an equal sign for the operation and enter *14 (DUPKEY)* for the second operand.



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

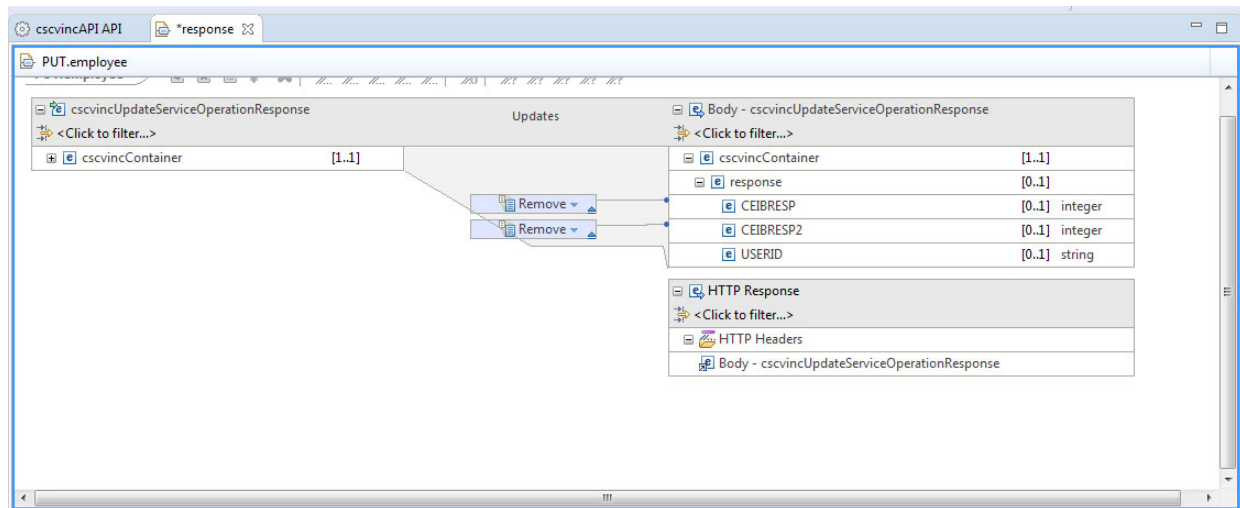
41. In the mapping view that opens, go to the right side of the mapping and fully expand *cscvincContainer*. You should see fields that correspond to the fields exposed in the interface by the services developer.



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

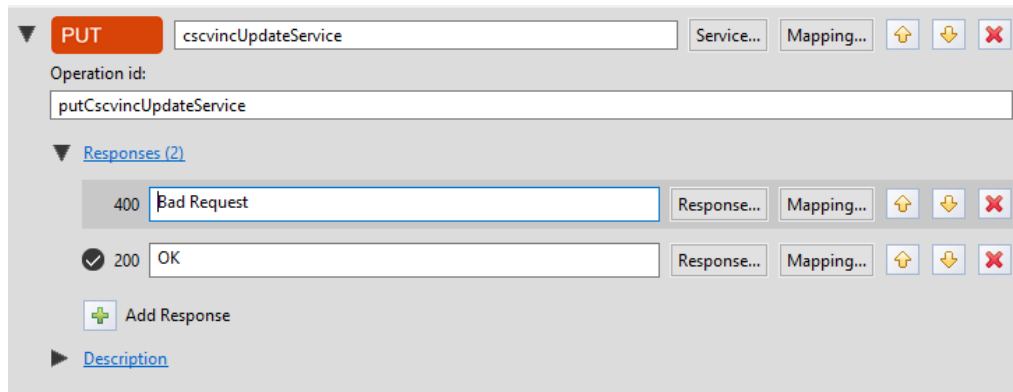
43. For the **PUT** method we want the default response mapping to return only the *USERID* field to the client. To change/review the fields that will be returned click the **Mapping** button beside the **PUT** method and select the *Open Default Response Mapping* option.

44. Fully expand *cscvincContainer* and use the slider bar to fully expose the *cscvincContainer* structure. Use the left mouse button and draw a dotted line box that fully includes the *CEIBRESP* and *CEIBRESP2* fields. When you release the button, these fields should be selected (the background should be blue).
45. Right mouse button click on any of the selected fields and select the *Add Remove transform* from the list of options



46. Use the **Ctrl-S** key sequence to save all changes and close the *PUT.employee.{numb}* view.
47. Click the **Mapping** button again but this time select *Define Response Codes*.

48. Click the plus sign beside *Add Response* and use the pull-down arrows to select a *Response code* of *400 – Bad Request*, use the pull-down to select field *cscvincSelectServiceOperationResponse/cscvincContainer/response/CEIBRESP* for the first operand, an equal sign for the operation and enter *16* (INVREQ) for the second operand.



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. 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.

Deploy the API to a z/OS Connect Server

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

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

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

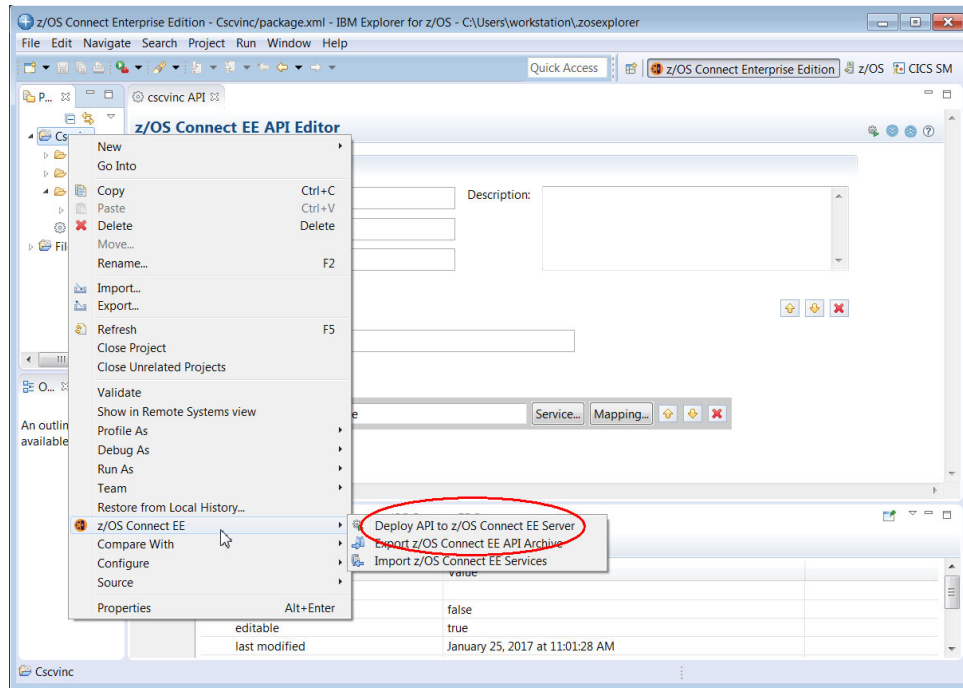
  <zosconnect_cicsIpicConnection id="cscvinc"
    host="wg31.washington.ibm.com"
    port="1491"/>

</server>
```

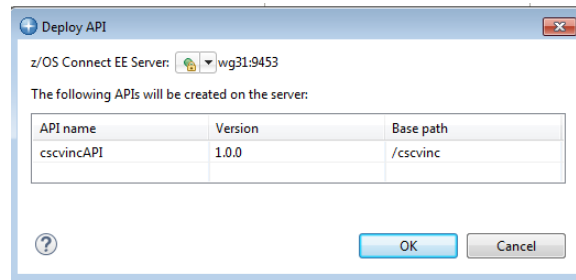
Figure 1 - *ipic.xml*

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

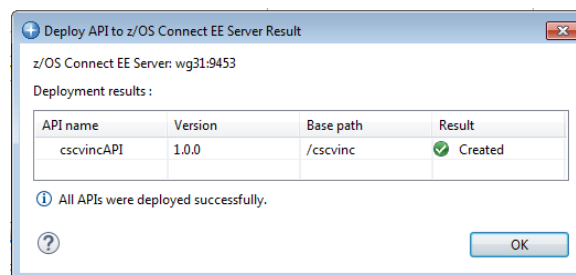
2. In the *Project Explorer* view (upper left), right-mouse click on the *CscvincAPI* folder, then select *z/OS Connect* → *Deploy API to z/OS Connect Server*.



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/ats/zosconnect/servers/zceesrv1/resources/zosconnect/apis* directory.

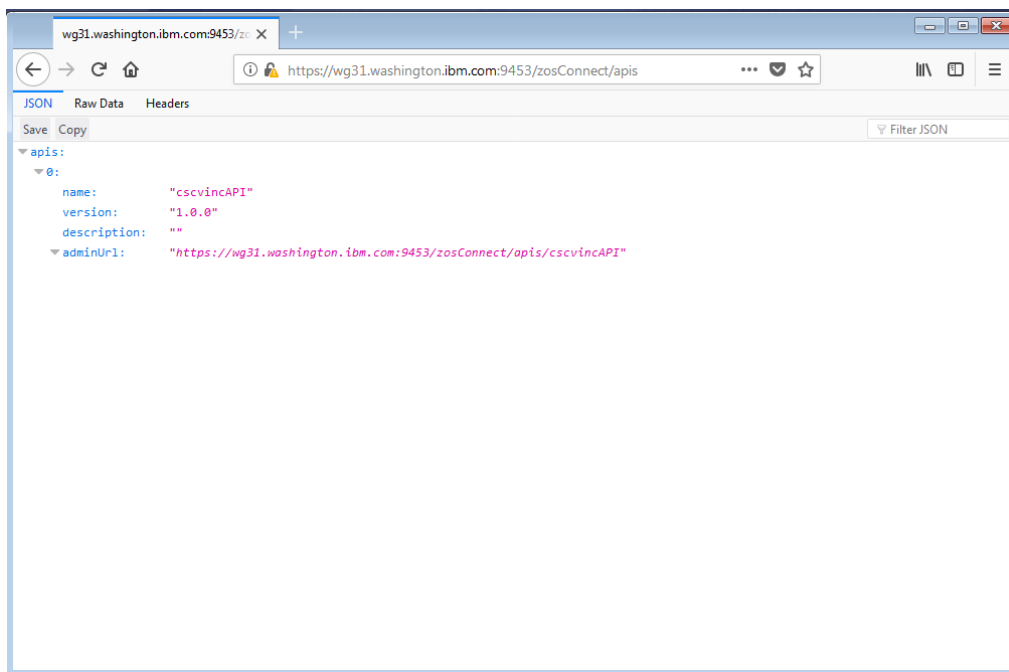


Test the APIs with a CICS Container Application

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.

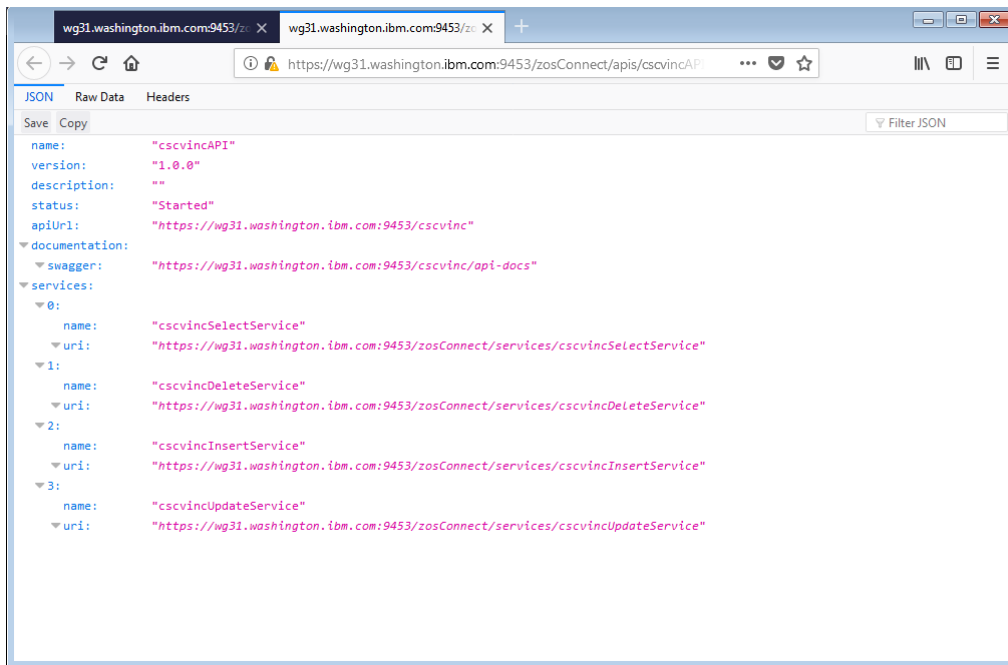
1. First enter URL <https://wg31.washington.ibm.com:9453/zosConnect/apis> in the Firefox browser and you should see the window below. The API *cscvinc* is now displayed. This is because this API was just deployed to this server.

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

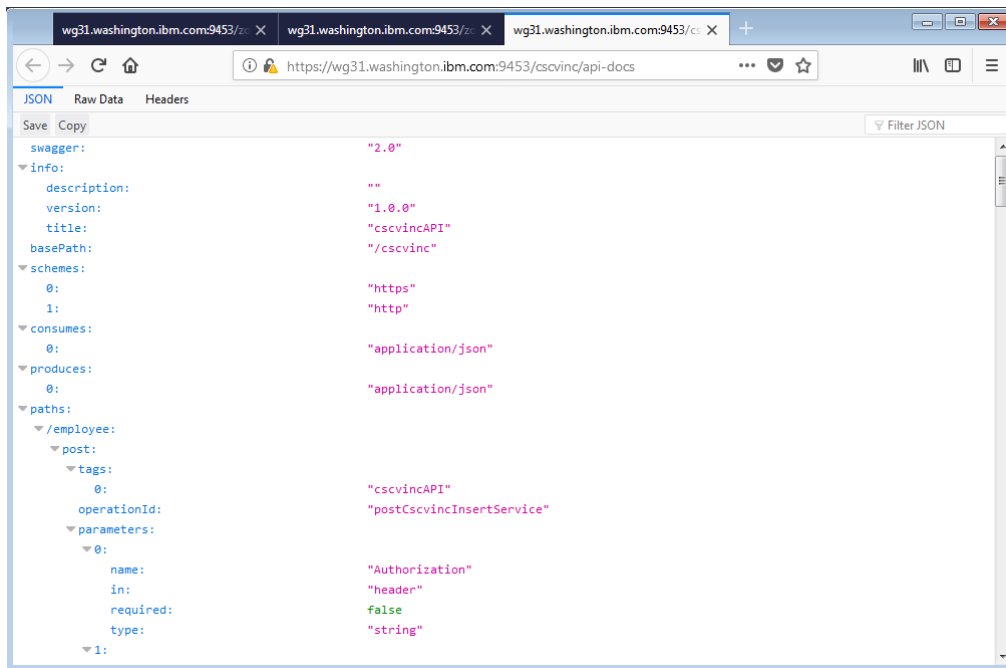


Tech Tip: It is very important to access the z/OS Connect server from a browser prior to any testing using the Swagger UI. Accessing a z/OS Connect URL from a browser starts an SSL handshake between the browser and the server. If this handshake has not performed prior to performing any test the test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

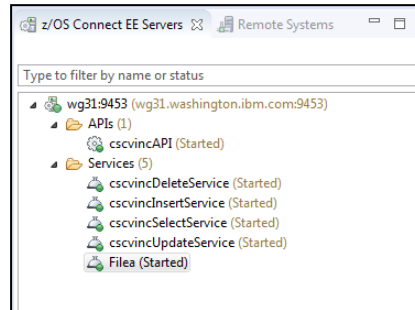


3. Finally click on the *swagger* URL and you should see the Swagger document associated with this API.

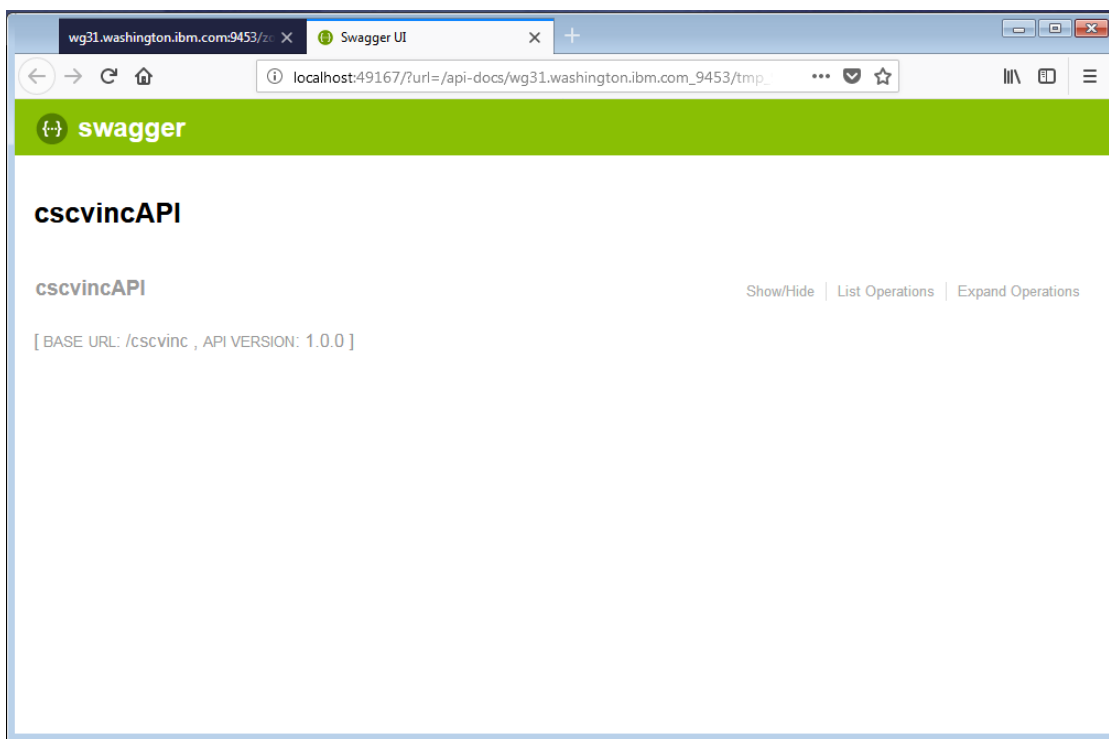


4. Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This document can be used by a developer or other tooling to develop REST clients for this specific API.

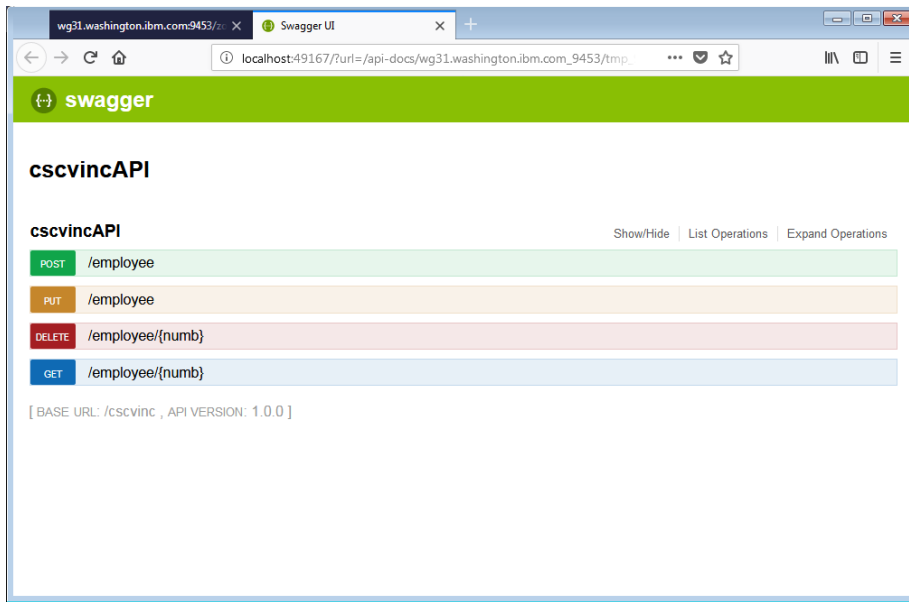
5. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect Servers*. Expand *wg31:9453* and then expand the *APIs* folder. You should see a list of the APIs installed in the server.



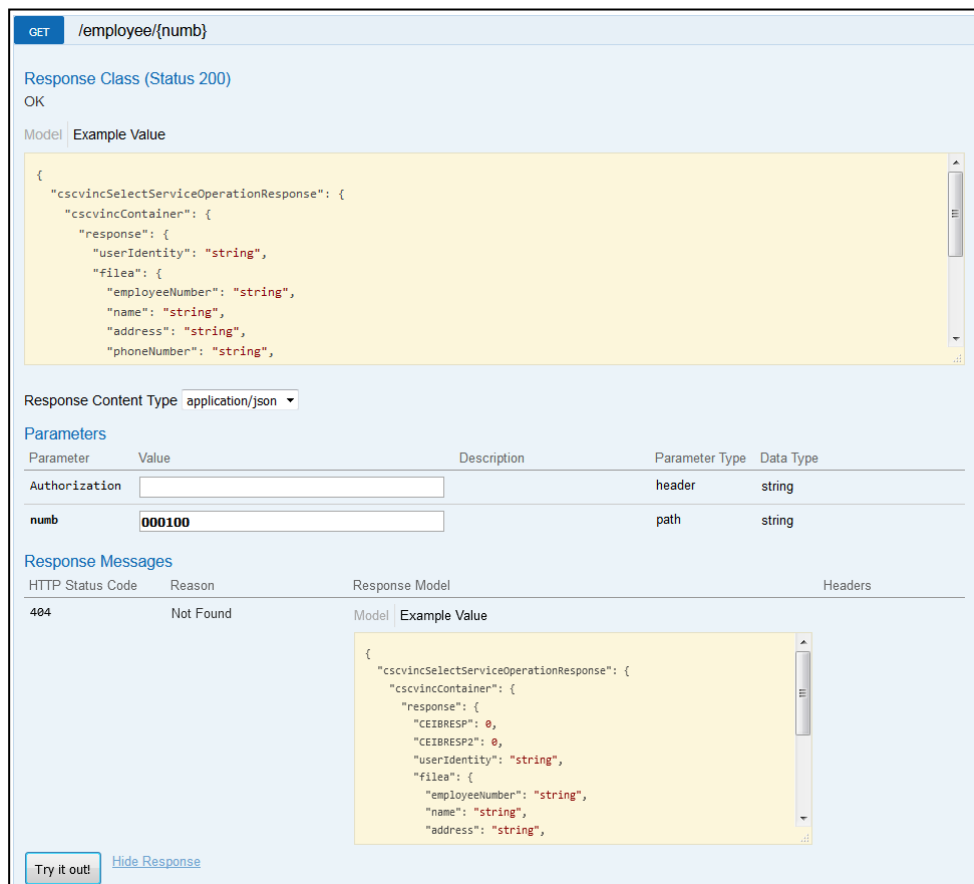
6. Right mouse button click on *cscvincAPI* and select *Open in Swagger UI*. Click **OK** if an informational prompt appears. This will open a Firefox window showing a *Swagger* test client (see below).



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 the *Get* box and scroll down until the method *Parameters* are displayed as shown below:



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

Response Body

```
{
  "cscvincSelectServiceOperationResponse": {
    "cscvincContainer": {
      "response": {
        "userIdentity": "CICSUSER",
        "filea": {
          "date": "26 11 81",
          "amount": "$0100.11",
          "address": "SURREY, ENGLAND",
          "phoneNumber": "32156778",
          "name": "S. D. BORMAN",
          "employeeNumber": "000100"
        }
      }
    }
  }
}
```

Response Code

200

10. Enter **000101** in the area beside numb and click the **Try it Out!** button. Scroll down and you see the response.

Response Body

```
{
  "cscvincSelectServiceOperationResponse": {
    "cscvincContainer": {
      "response": {
        "CEIBRESP2": 80,
        "userIdentity": "CICSUSER",
        "filea": {
          "date": "",
          "amount": "",
          "address": "",
          "phoneNumber": "",
          "name": "",
          "employeeNumber": "000101"
        },
        "CEIBRESP": 13
      }
    }
  }
}
```

Response Code

404

Tech Tip: The HTTP status code of 404 – Not Found was set in the API response message when then CEIBRESP return code was 13.

The record was not found.

11. In a new or existing 3270 session start a session with CICS by entering **CICS**. Clear the screen and enter CICS transaction **CEDX CSMI** to start a CICS execution diagnostic facility (EDF) on transaction CSMI.
12. Send another GET API request and you should see the EDF session start in the 3270 session.

The screenshot shows a 3270 terminal window titled 'WG31'. The header bar displays 'TRANSACTION: CSMI PROGRAM: DFHMIRS TASK: 0000814 APPLID: CICS53Z DISPLAY: 00'. The main screen shows 'STATUS: PROGRAM INITIATION' followed by various system parameters in green text:

```

EIBTIME      = 103343
EIBDATE      = 0119277
EIBTRNID     = 'CSMI'
EIBTASKN     = 814
EIBTRMID     = '/AC7'

EIBCP0SN     = 0
EIBCALEN     = 0
EIBAID       = X'00'
EIBFN        = X'0000'
EIBRCODE     = X'000000000000'
EIBDS        = '.....'
EIBREQID     = '.....'
  
```

Below the parameters, there are control options:

```

ENTER: CONTINUE
PF1 : UNDEFINED      PF2 : SWITCH HEX/CHAR    PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE    PF6 : USER DISPLAY
PF7 : SCROLL BACK     PF8 : SCROLL FORWARD    PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY       PF12: UNDEFINED
  
```

The status bar at the bottom shows 'MA B' and '01/001'. A small icon in the bottom left corner indicates 'Connected to remote server/host wg31 using lu/pool TCP00110 and port 23'.

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

The screenshot shows the same 3270 terminal window, now displaying 'DISPLAY ON CONDITION:-'. The main screen shows the following information in green text:

```

COMMAND: EXEC CICS LINK
OFFSET: X'.....'
LINE NUMBER: .....
CICS EXCEPTIONAL CONDITION: ERROR
ANY CICS CONDITION: NO
TRANSACTION ABEND: YES
NORMAL TASK TERMINATION: YES
ABNORMAL TASK TERMINATION: YES

DLI ERROR STATUS: ■ ...
ANY DLI ERROR STATUS:
  
```

Below this, there are control options:

```

ENTER: CURRENT DISPLAY
PF1 : UNDEFINED      PF2 : UNDEFINED      PF3 : UNDEFINED
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE    PF6 : USER DISPLAY
PF7 : UNDEFINED      PF8 : UNDEFINED      PF9 : UNDEFINED
PF10: UNDEFINED      PF11: UNDEFINED     PF12: REMEMBER DISPLAY
  
```

The status bar at the bottom shows 'MA B' and '13/035'. The same connection icon is present in the bottom left corner.

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

```

WG31
File Edit Settings View Communication Actions Window Help
TRANSACTION: CSMI PROGRAM: DFHMIRS TASK: 0000824 APPLID: CICS53Z DISPLAY: 00
STATUS: ABOUT TO EXECUTE COMMAND
EXEC CICS LINK PROGRAM
PROGRAM ('CSCVINC ')
SYNCONRETURN
CHANNEL ('Channel ')
NOHANDLE

-

OFFSET:X'001CD6' LINE: EIBFN=X'0E02'

ENTER: CONTINUE
PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : UNDEFINED
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK

MA B 11/022
Connected to remote server/host wg31 using lu/pool TCP00110 and port 23

```

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

```

Session A
File Edit View Communication Actions Window Help
TRANSACTION: CSMI PROGRAM: CSCVINC TASK: 0000857 APPLID: CICS53Z DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
EXEC CICS GET CONTAINER
CONTAINER ('Container1 ')
CHANNEL ('JSONCHANNEL ')
INTO ('S..... 111111 '...')
FLENGTH (89)
CONVERTST (734)
NOHANDLE

-

OFFSET:X'000E6A' LINE:000161 EIBFN=X'3414'
RESPONSE: NORMAL EIBRESP=0

ENTER: CONTINUE
PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK

MA A 12/038
Connected to remote server/host wg31b using lu/pool TCP00111 and port 23

```

Tech Tip: You may experience timeouts in because the runtime has been configured to timeout if a response is not received within 30 seconds (connectionTimeout).

16. 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.

The screenshot shows a window titled "Session A" with a menu bar (File, Edit, View, Communication, Actions, Window, Help) and a toolbar. The main display area contains the following text:

```
TRANSACTION: CSMI PROGRAM: CSCVINC TASK: 0000857 APPLID: CICS53Z DISPLAY: 00
STATUS: COMMAND EXECUTION COMPLETE
EXEC CICS PUT CONTAINER
CONTAINER ('Container1 ')
FROM ('S.....CICSUSER 11111C. BAKER OTTAWA, ONTARIO '...')
FLENGTH (97)
NOHANDLE
```

Below this, there is a separator line and then:

```
OFFSET: X'0014C2' LINE: 000259 EIBFN=X'3416'
RESPONSE: NORMAL EIBRESP=0
```

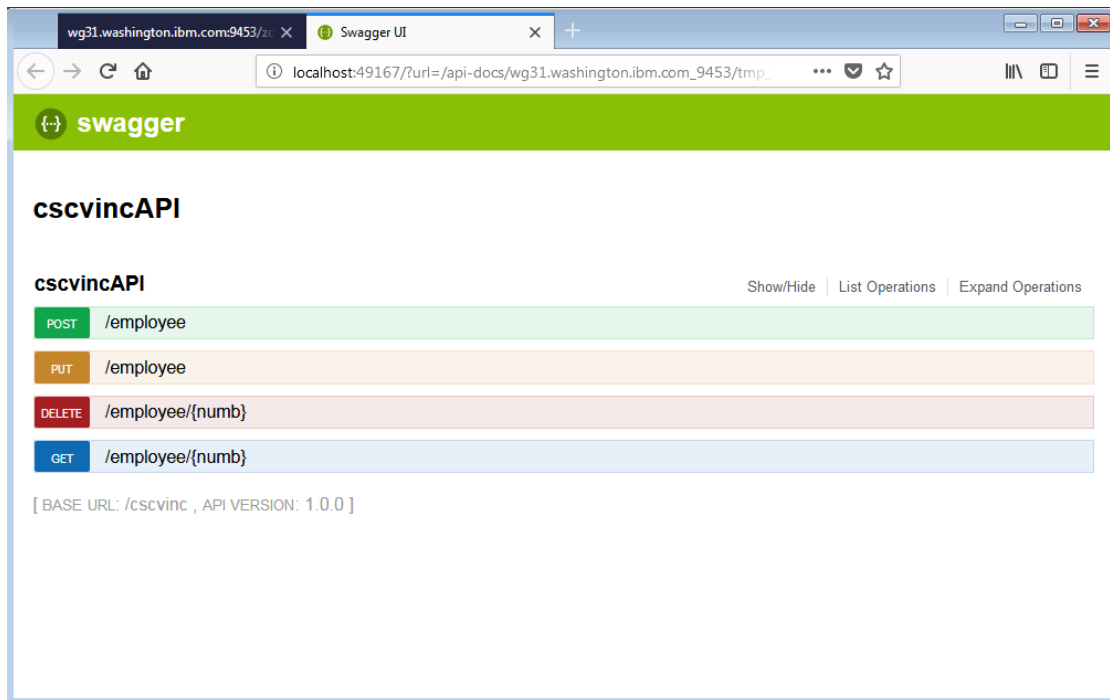
At the bottom, there is a list of function keys:

```
ENTER: CONTINUE
PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : END EDF SESSION
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK
```

The status bar at the bottom shows "MA A" and "10/041". The connection information at the very bottom reads: "Connected to remote server/host wg31b using lu/pool TCP00111 and port 23".

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

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



19. 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*.

DELETE /employee/{numb}

Response Class (Status 200)
normal response

Model **Example Value**

```
{
  "cscvincServiceOperationResponse": {
    "Container1": {
      "RESPONSE_CONTAINER": {
        "CEIBRESP": 0,
        "CEIBRESP2": 0,
        "USERID": "string",
        "FILEA_AREA": {
          "NUMB": "string"
        }
      }
    }
  }
}
```

Response Content Type: application/json

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|---------------|------------------------|-------------|----------------|-----------|
| numb | 666666 | | path | string |
| Authorization | Basic RnJlZDpmcmVkcHdk | | header | string |

Try it out!

20. Press the **Try it out!** button and you see the results in the response body. The response code of 200 indicated the record was successfully deleted.

Response Body

```
{
  "cscvincDeleteServiceOperationResponse": {
    "cscvincContainer": {
      "response": {
        "USERID": "CICSUSER"
      }
    }
  }
}
```

Response Code

200

21. Press the **Try it out!** button again and you should see a response code of 404 (the record to be deleted was not found).

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

| 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 | \$809.99 | ***** |
| 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.