IBM z/OS Connect EE V3.0

Developing RESTful APIs for MVS Batch

Wildfire
Technical Hands-On Workshops

WSC Wildfire Team
IBM z Systems

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.

Overview

Important – You do not need any skills with MVS Batch to perform this exercise. Even if MVS Batch is not relevant to your current plans performing this exercise will give additional experience using the API Toolkit with developing 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 you can start with section *z/OS Connect EE APIs and MVS Batch* 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.

- ✓ 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.

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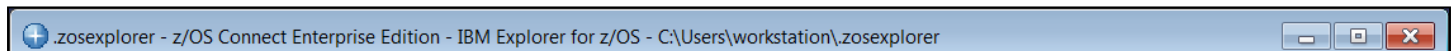
___1. On the workstation desktop, locate the *z/OS Explorer* icon and double click on it to open the Explorer.

___2. You will be prompted for a workspace:
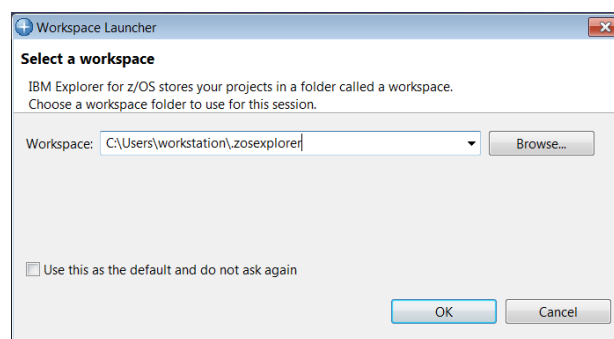
Take the default value by clicking **OK**.

___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.
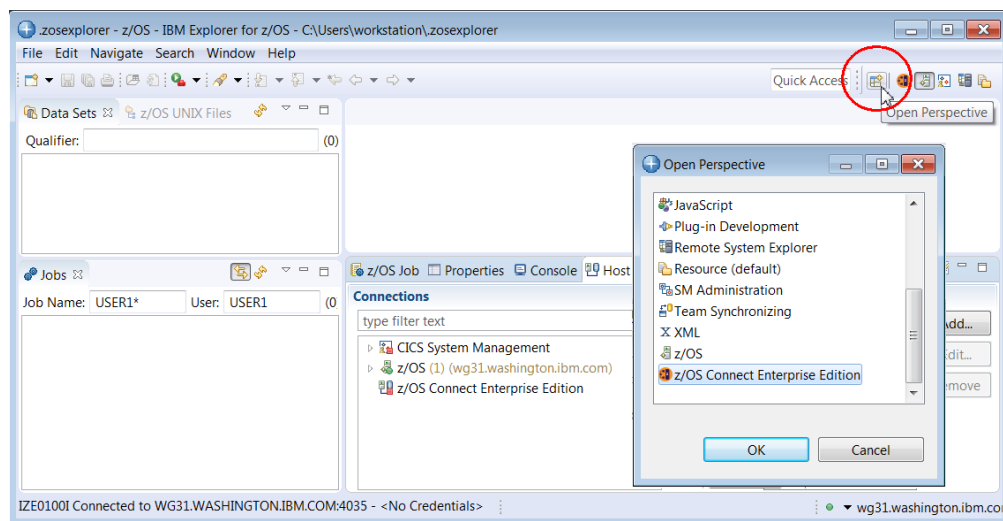
___5. To add a connection to the z/OS Connect Server select *z/OS Connect Enterprise Edition* connection in the *Host connection*s tab in the lower view and then click the **Add** button.

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.

___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.

___10. A connection to the remote z/OS system was previously added. In the *Host Connection* view expand *z/OS Remote System* under *z/OS* and select *wg31.washington.ibm.com*. If the connection is not active the **Connect** button will be enabled. Click the **Connect** button and this will establish a session to the z/OS system. This step is required when submitting job for execution and viewing the output of these jobs later in this exercise.

Summary

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

z/OS Connect EE APIs and MVS Batch

In this section the Eclipse-based z/OS Explorer tool with the z/OS Connect EE API Editor will be used to compose an API for a MVS batch program. When development of the API is completed, the API archive (AAR) file will be deployed to the z/OS Connect server configuration directory and tested.

Run the job to create the data conversion artifacts

First run the BAQLSJ2JS utility to generate the conversion artifacts required for the execution of a service, the service that will be used to access a MVS batch program. In this exercise the service will be named *Filea.*

___1. Switch to the *Remote System Explorer* perspective and in the *Remote Systems* view expand data set *USER1.ZCEE.CNTL,* see page 4.

___2. Under data set *USER1.ZCEE.CNTL* you will see members created ahead of time to generate the data conversion artifacts, e.g. JSON request and response schemas, SAR files and WSBIND files.
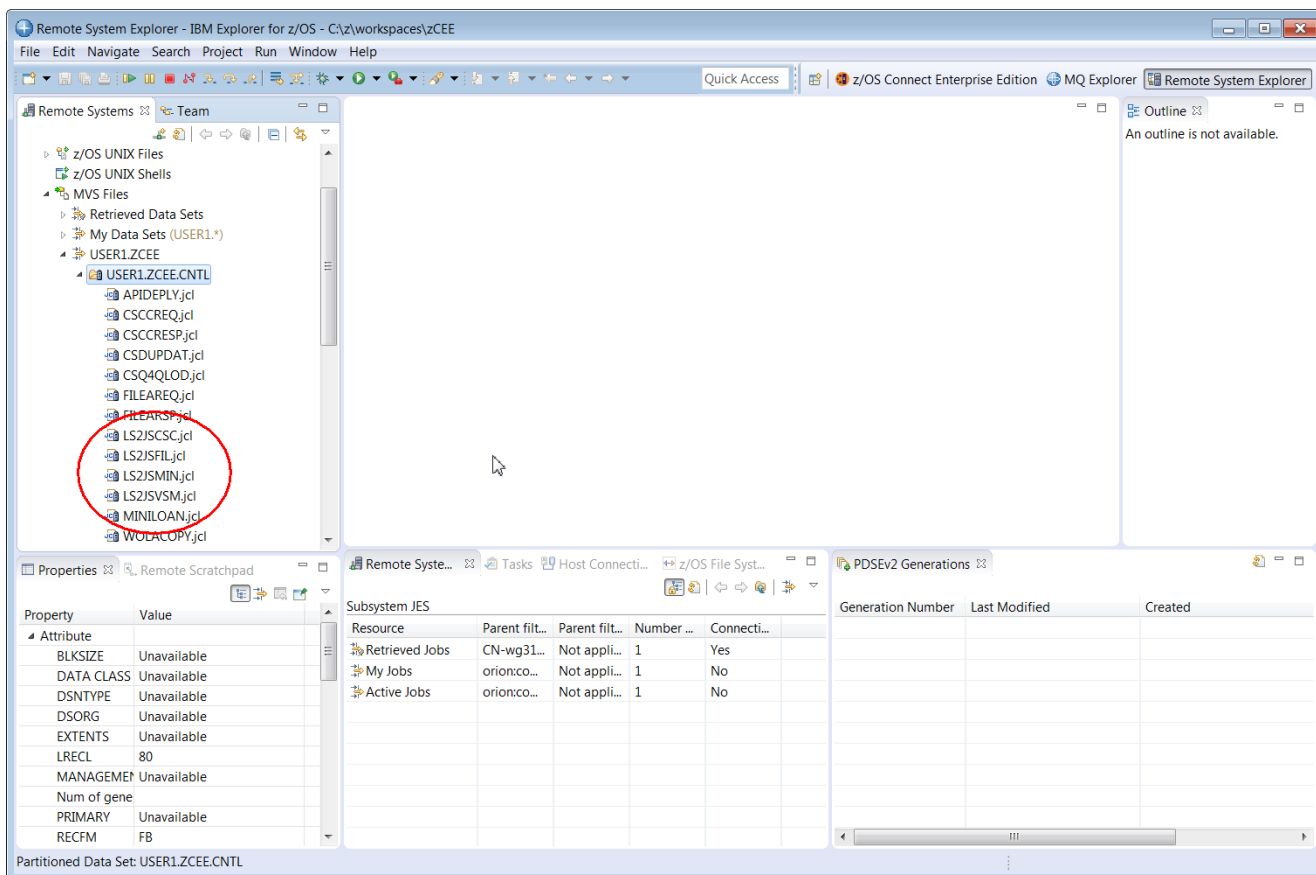
Member *LS2JSVSM* is the job that executes the BAQLS2JS program that will convert the COPYBOOKs for service *Filea*. N.B. disregard the file extensions of jcl.

___3. Open member *LS2JSVSM* by double clicking the member**.** You will see that it is a very simple JCL job:

```
//LS2JSVSM JOB (0),MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID,REGION=500M
 //ASSIST   EXEC PGM=BPXBATCH
 //STDPARM DD DSN=USER1.ZCONN2.INPUT(FILEA),DISP=SHR
 /*
 //STDOUT   DD SYSOUT=*
 //STDERR   DD SYSOUT=*
 //STDENV   DD *
 JAVA_HOME=/shared/java/J8.0_64
 //
```

The key thing we want you to notice is the STDPARM DD, which points to a data set member where the input parameters for this job are located. The parameters can be included inline, but in this exercise some were longer than 80 columns wide and the utility does not yet support continuation. So we put the parameters in a data set allocated with width 256 columns.

___4. Expand data set *USER1.ZCEE. INPUT*.

___5. You will see four members:

Mitch Johnson (mitchj@us.ibm.com)
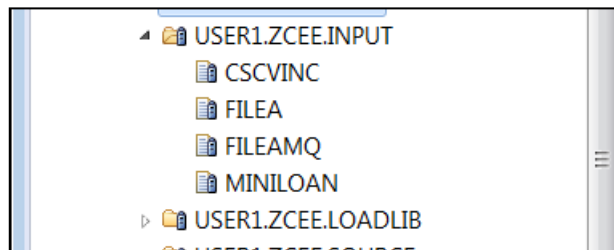
___6. Open member *FILEA* member and you should see something like this:

```
PGM /shared/IBM/zosconnect/V3r0/bin/baqls2js
PDSLIB=USER1.ZCEE.CNTL
REQMEM=FILEAREQ                              1
RESPMEM=FILEARSP
MAPPING-LEVEL=4.0
CHAR-VARYING=COLLAPSE
JSON-SCHEMA-REQUEST=/var/ats/zosconnect/servers/zceesrvr/dataXform/json/Filea_request.json      2
JSON-SCHEMA-RESPONSE=/var/ats/zosconnect/servers/zceesrvr/dataXform/json/Filea_response.jso
LANG=COBOL
LOGFILE=/var/ats/zosconnect/servers/zceesrvr/dataXform/Filea.log
PGMINT=COMMAREA                              3
PGMNAME=ATSFILEA
WSBIND=/var/ats/zosconnect/servers/zceesrvr/dataXform/bind/Filea.wsbind      4
SERVICE-ARCHIVE=/var/ats/zosconnect/servers/zceesrvr/dataXform/sars/Filea.sar
SERVICE-NAME=Filea
URI=http://zceesrvr.example.org:8080/Filea
```

1. Pointers to the data set where the request and response COPYBOOK members are located

2. Pointers to the directory where the request and response JSON schemas will be created.

3. The PGMINIT indicates that this is a *COMMAREA* application.

4. The location where the wsbind and service archive files will be written.

Note that even though the target application is an MVS batch application setting PGMINIT to COMMAREA is required. The default for this parameter is CHANNEL which would cause the generation of an invalid interface for an MVS batch application. Pointers to the target directories for the binding file (required at runtime) and the service archive (SAR) file. The SAR file will be downloaded to a workstation where it will be imported in the z/OS Connect Enterprise Edition eclipse tooling.

___7. Submit job *LS2JSVSM* for execution. This job should complete with a maximum return code of zero (e.g. MAXCC=0000). Hint: To submit a member for execution right mouse button click on the member and select the Submit option.
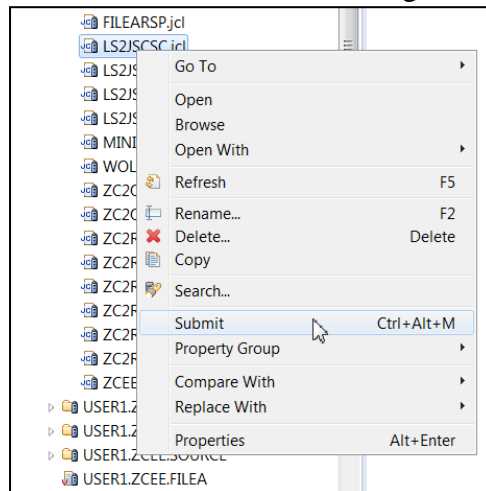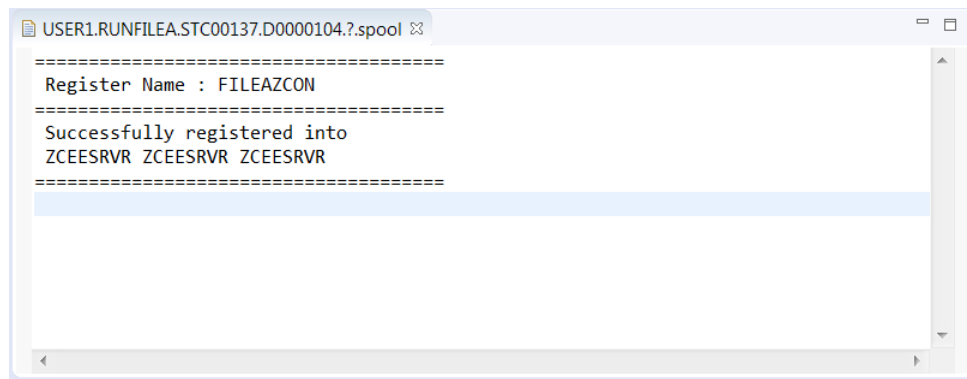

> . **Tech Tip:** To submit a job for execution when using the IBM z/OS Explorer select the member and right mouse button click and select the *Submit* option. Or if the member is currently opened simply right mouse button click and select the *Submit* option. Click the **Locate Job** button on the *Job Confirmation* pop-up. This will display the job output in the *Retrieved Job* section under *JES* in the *Remote Systems* view. The job's output can be viewed right mouse button clicking and selecting the *Open* option.

___8. Submit job RUNFILEA for execution. This job should not terminate.

___9. Click *OK* on the next screen. After submitting the job, expand the *JES* icon and expand *My Jobs* to display the held output. The output of an entire job can be viewed by opening the spool file or output for a specific DD statement can be viewed by expanding the job and opening a specific output file.

Mitch Johnson (mitchj@us.ibm.com)
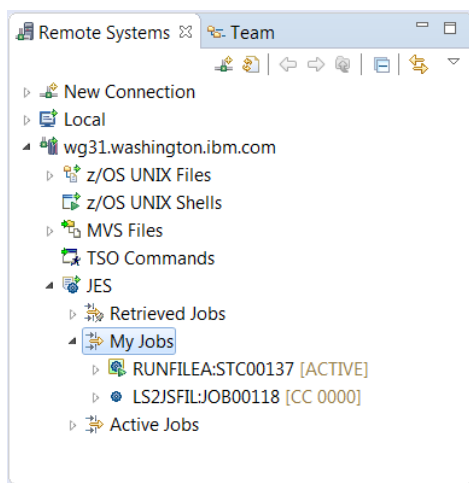
11. Expand the active *RUNFILEA* job to display its output and open *ATSFILEA:SYSOUT* to display the job's output for this DD statement.

```
USER1.RUNFILEA.STC00137.D0000104.?.spool ⊠
======================================
 Register Name : FILEAZCON
======================================
 Successfully registered into
 ZCEESRVR ZCEESRVR ZCEESRVR
======================================
```

Tech Tip: As you perform the subsequent steps of this exercise this would be a good way to monitor the execution of the APIs. Just refresh the output of this DD statement and the actions being performed will be displayed here.

Close any opened editor views and do not save any changes

Import the SAR file

The next step is to move the SAR file from the OMVS directory structure by importing it into the workspace.

___1. In the z/OS Explorer in the *z/OS Connect Enterprise Edition* perspective in the the *Project Explorer* view (upper left). 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 *services* folder. Right mouse button click and select *Import* from the list of options. On the *Import – Select* window expand *Remote Systems* and select *Remote file system* and click **Next** to continue. On the *Import – Remote file system* press the *Browse* button. On Browse for Folder window use the pull-down allow to select *wg31.washington.ibm.com*. Expand the zCEE folder (this is the filter created earlier) and expand the *zceesrvr* folder, then expand the *dataXform* folder and then select the *sars* folder. Click **OK** to continue.

Mitch Johnson (mitchj@us.ibm.com)
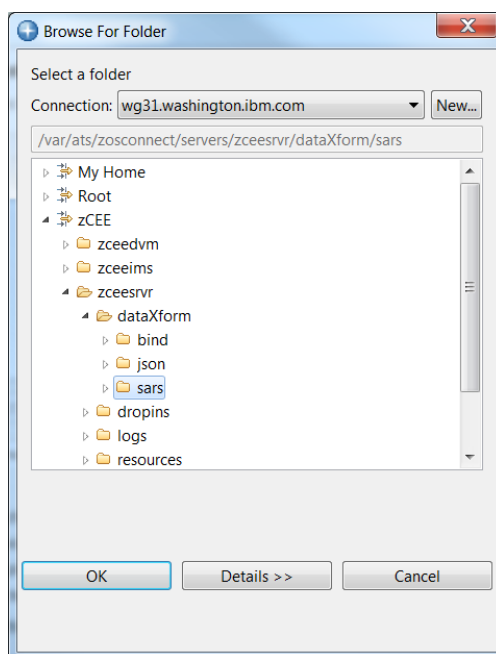
___4. On the *Import – Remote file system* window expand the *sars* folder and check the box beside *Filea.sar* and click the **Finish** button:
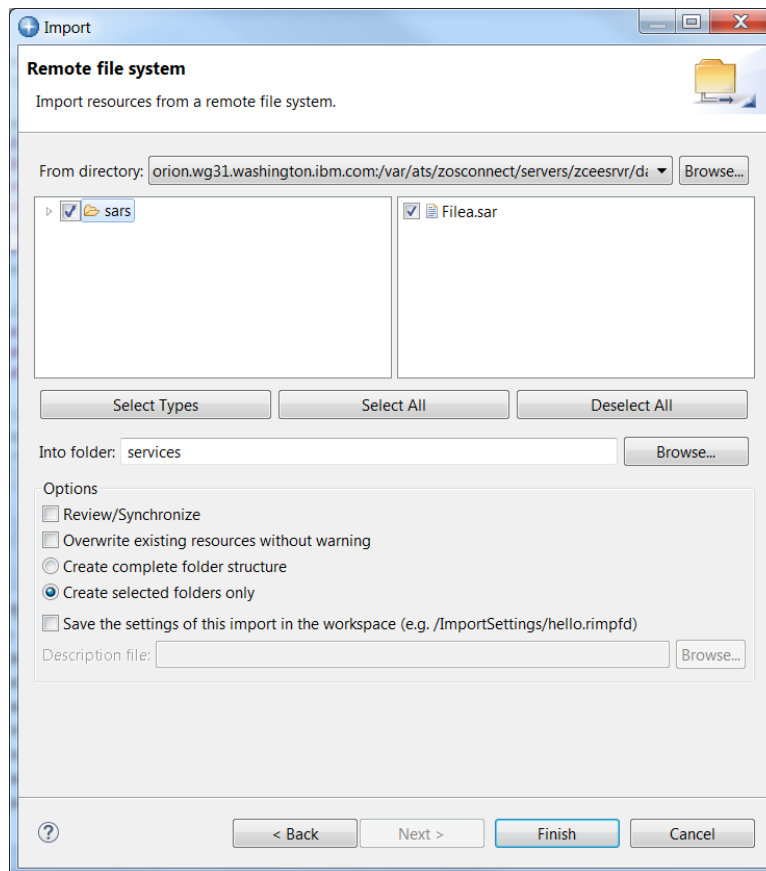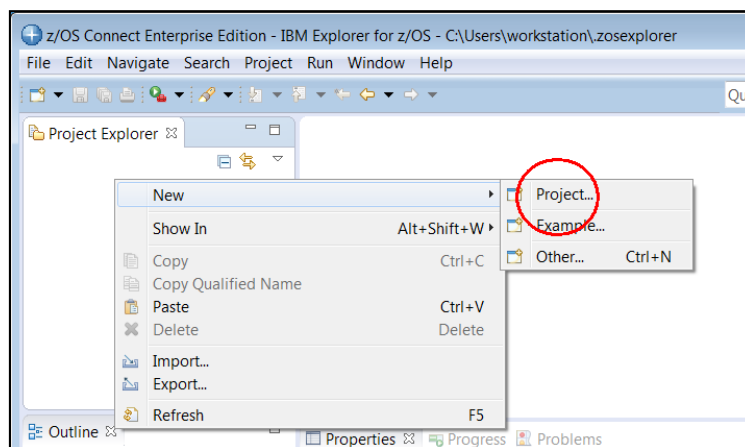

___5. In the *Project Explorer* view (upper left), expand the *services* folder to see the the imported SAR file.

Create the Filea API project

Next create the *Filea* API project.

___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* 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.
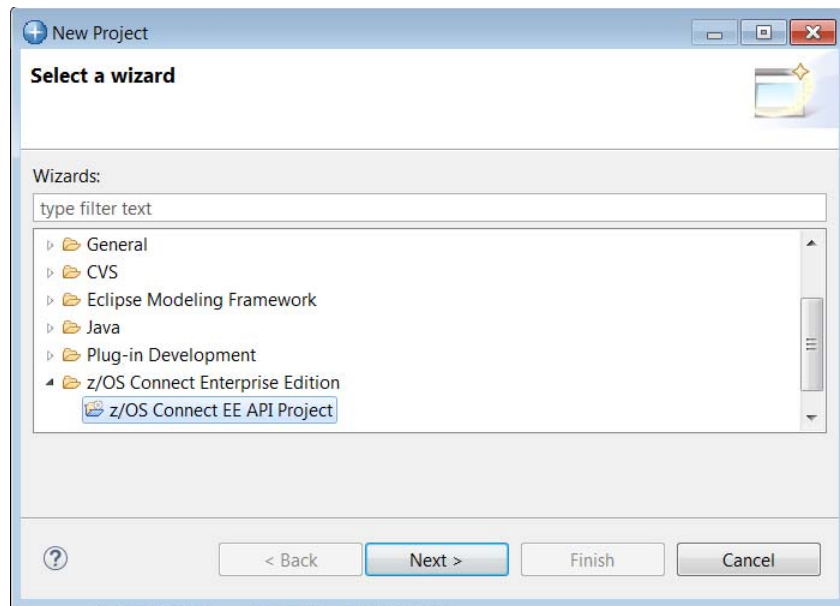

___3. Enter *Filea* for the *Project name*, *filea* for the *API name* and */filea* for the *Base path* value.   Click **Finish** to continue.
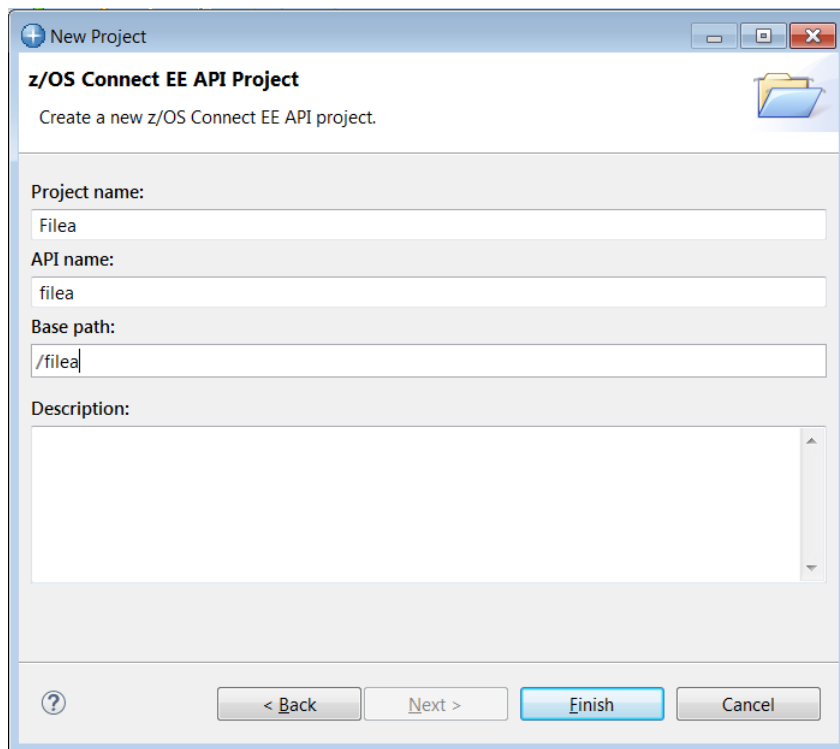
> **Important:** The values are somewhat arbitrary, but they do relate to later tasks. If you use the values and cases as supplied then the subsequent commands and the use of subsequent URLs will work seamlessly. Case is most important with regards to the *Base path*. A base path */Filea* is not the same as */filea* or */filaA* and the same applies to *API name.*

Mitch Johnson (mitchj@us.ibm.com)

___13. You should now see something like below. The displayable areas of the views may be adjusted by dragging the view boundary lines.
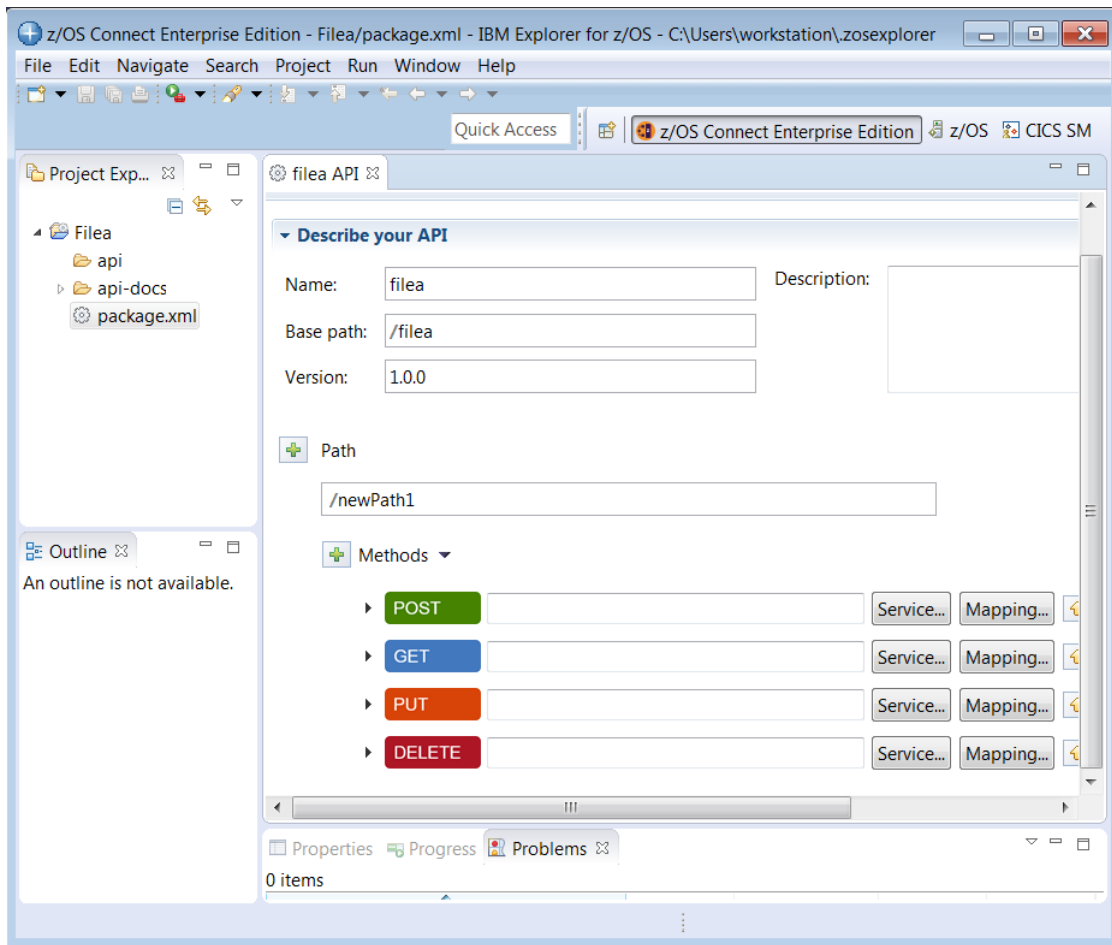
Summary

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

Mitch Johnson (mitchj@us.ibm.com)

Compose the API for the batch application

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

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

The *{employee}* element is a path parameter in the URL that will be used to provide the key of the record for get, put and delete REST requests.

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

https://hostname:port/filea/employee/######

where ###### is the key of a VSAM record.

___2. For the *Filea* service when the *employee* path parameter is present the supported HTTP methods will be **GET**, **PUT** and **DELETE**. The **POST** method will be handled by a different URL. Remove the **POST** method by clicking the *X* icon to the right of the POST method.

That should leave you with the **GET, PUT** and **DELETE** methods.

___3. Click on the **Service** button to the right of the **GET** method. Then on the *Select a z/OS Connect EE Service* window clock on the **Workspace** button. Expand the *services* folder and select *Filea.sar*. Click **OK** three times. This will populate the field to the right of the method. This has imported this SAR file into the *Filea* project (expand the new *services* folder in *Filea*). For the **PUT** and **DELETE** methods this means all you have to do to specify the service is to click the **Service** button and select *Filea*. Add service *Filea* to these methods.

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

> **Tech-Tip: I**f 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.

Mitch Johnson (mitchj@us.ibm.com)

___5. Next, click on the **Mapping** button beside the **GET** method and then select *Open Request Mapping*:
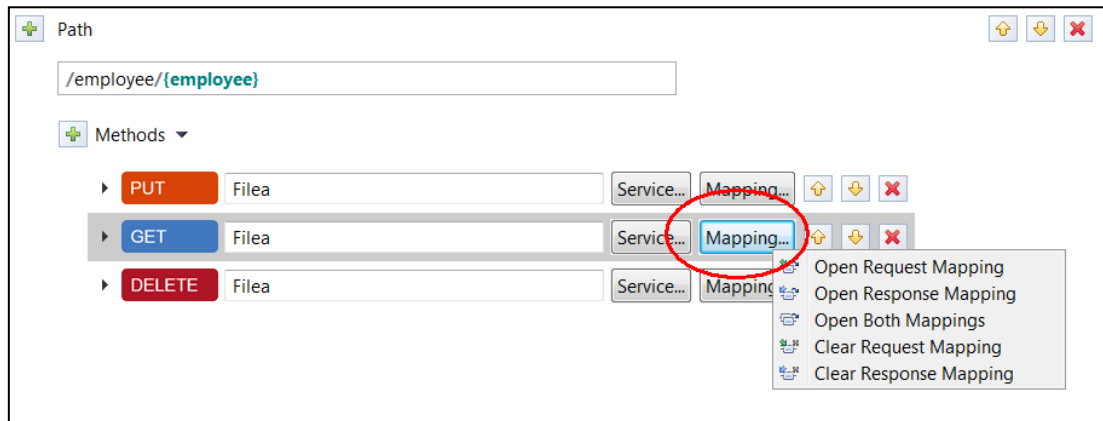
___6. In the mapping view that opens, go the right side of the mapping (which represents the COPYBOOK fields), and click the little + signs to expand *request_data*. You should see fields that correspond to the fields defined in the original COBOL copy book *FILEAREQ* in *USER1.ZCEE.CNTL.*

```
        01 REQUEST-DATA.
            10  request-type PIC X(1).
            10  stat        PIC X.
            10  numb        PIC X(6).
            10  name        PIC X(20).
            10  addrx       PIC X(20).
            10  phone       PIC X(8).
            10  datex       PIC X(8).
            10  amount      PIC X(8).
            10  comment     PIC X(9).
            10  filler      PIC X(21).          .
```

___7. Use the slider bar to fully expose the *request_data* structure. Use the left mouse button and draw a dotted line box that fully includes the *name, addrx, phone, datex, amount* and *comments* fields.
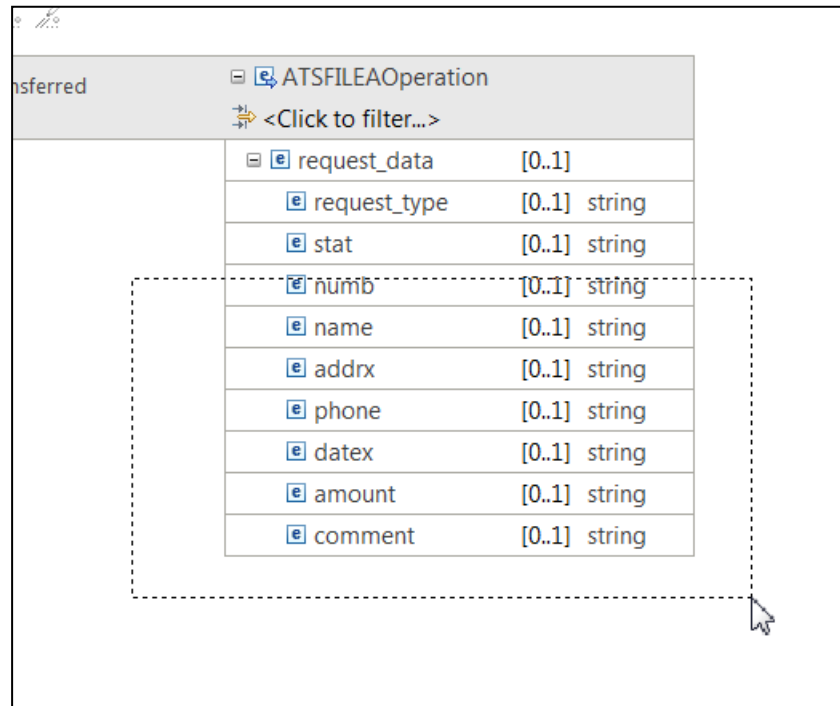
___8. When you release the button all of these fields should be selected (the background should be blue).
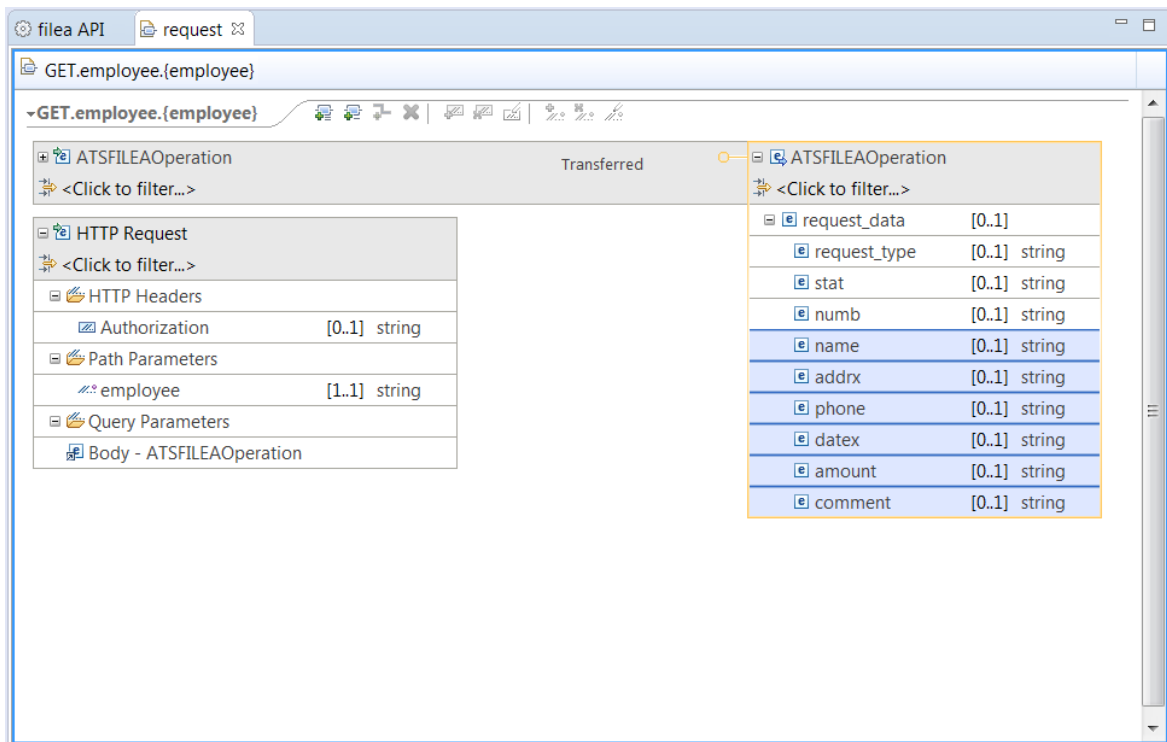
___9. Right mouse button click on any of the selected fields and select the *Add Remove transform* from the list of options.
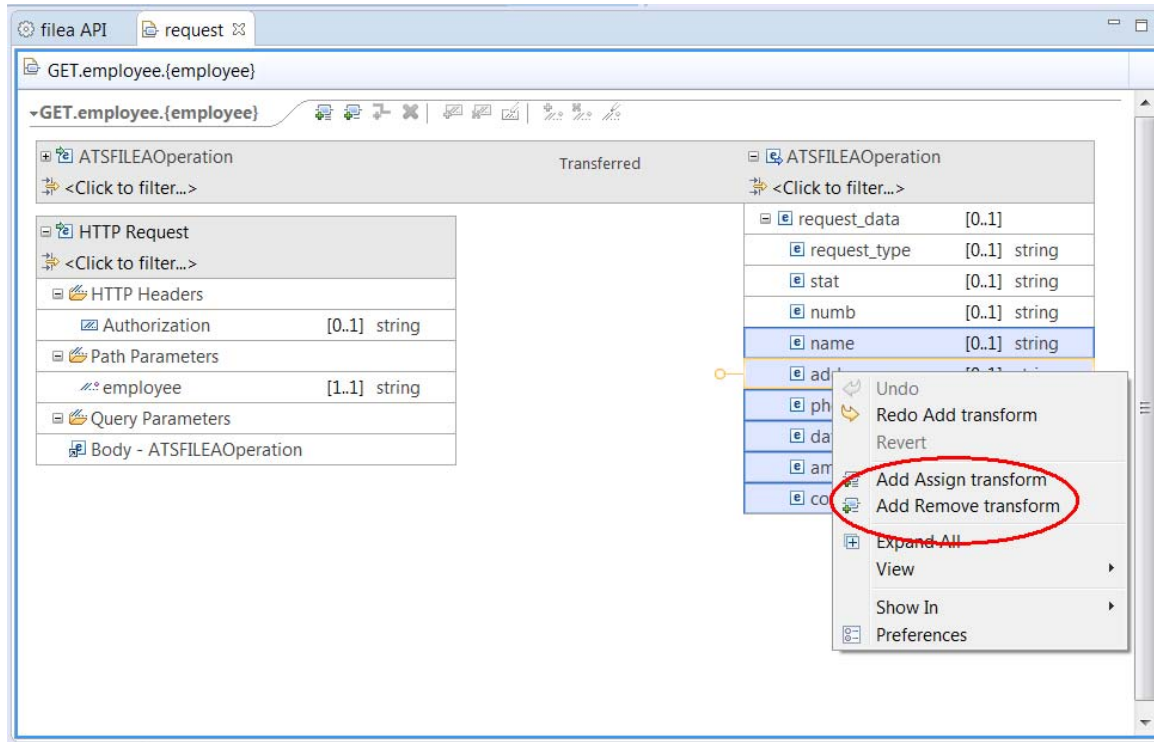
___10. This action generates multiple *Remove* requests (see below) for the selected fields. These fields are not required for a
GET method so these will not be exposed to the REST clients using this method.
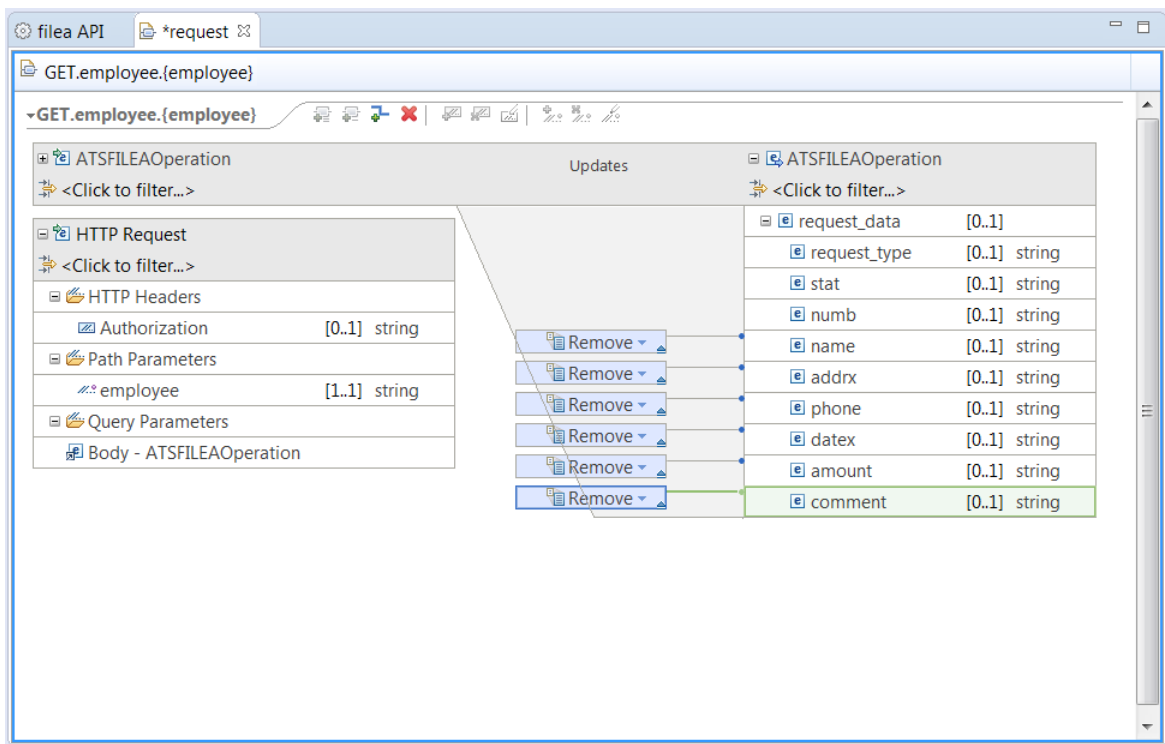
___11. The *stat* field is also not required. Select this single field and right mouse button click and select the *Add Remove transform* option to generate a *Remove* request for this field.
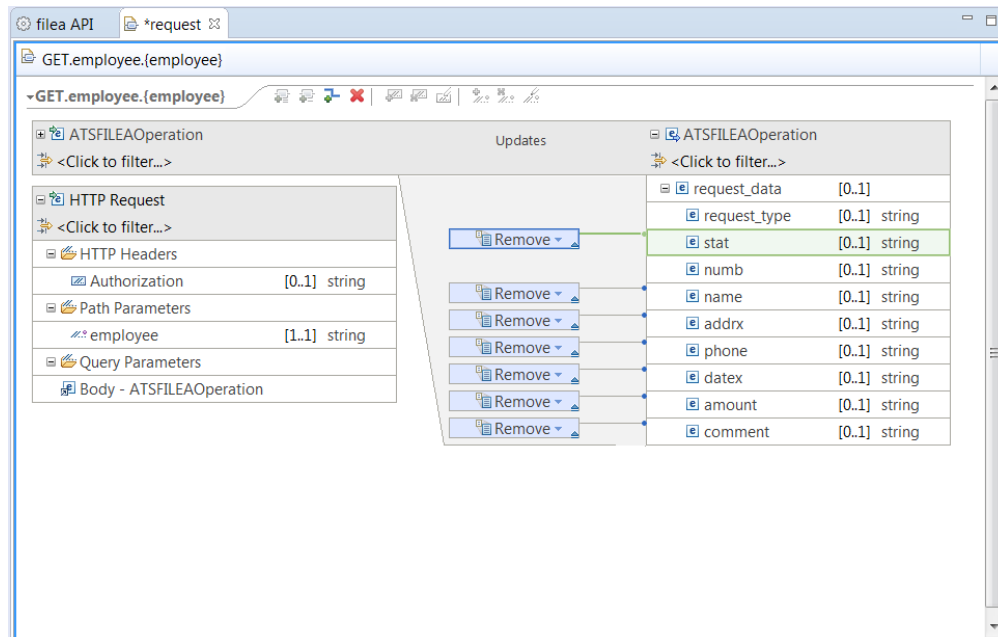
___12. This leaves two fields in the *request_data* structure, *request_type* and *numb*. The contents of the *numb* field should be provided by the *employee* path parameter from the URL. On the left hand side of the view select the *employee* path parameter with the left mouse button and drag it without releasing the mouse button to the *numb* field on the right hand side to create a *Move* connection from the *employee* path parameter to the *numb* field in the *request _data* structure.
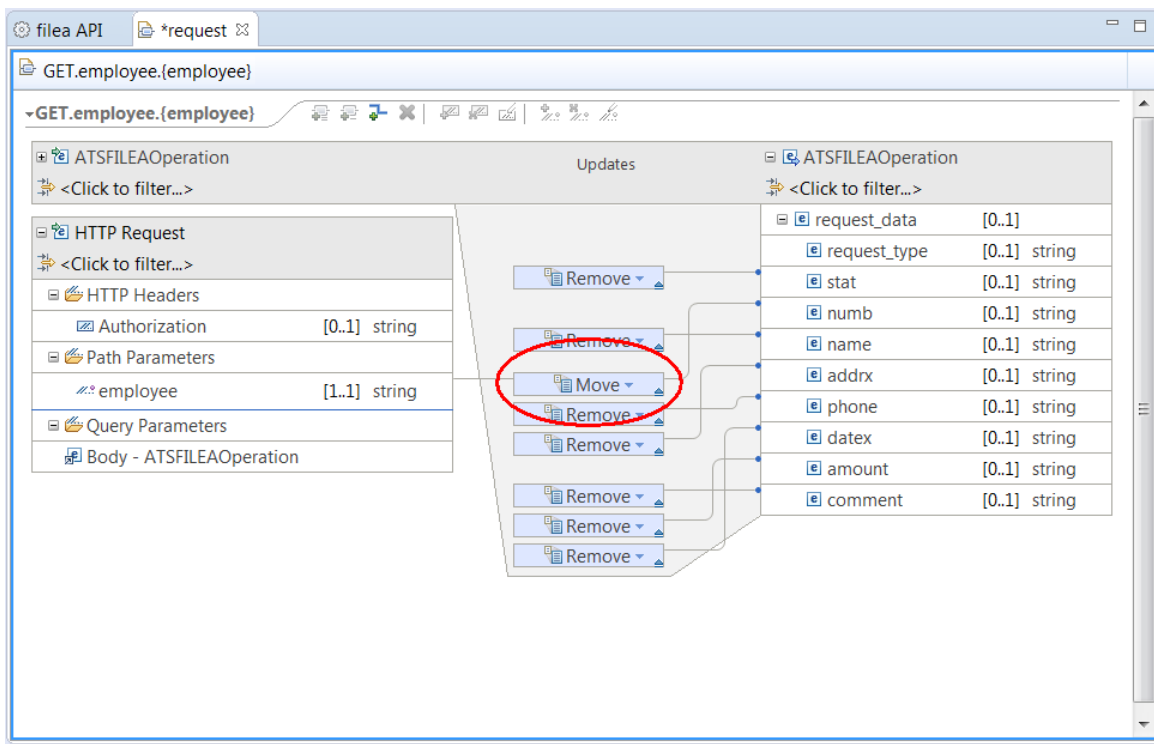
___13. The only *request_data* field left unhandled now is the *request_type* field. The target application uses this field to determine what action should be performed. For a **GET** method request this field should be set to a value of **G**. Select the *request_type* field and right button click and 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 **G** in the area beside *Value* in the *Properties* tab.
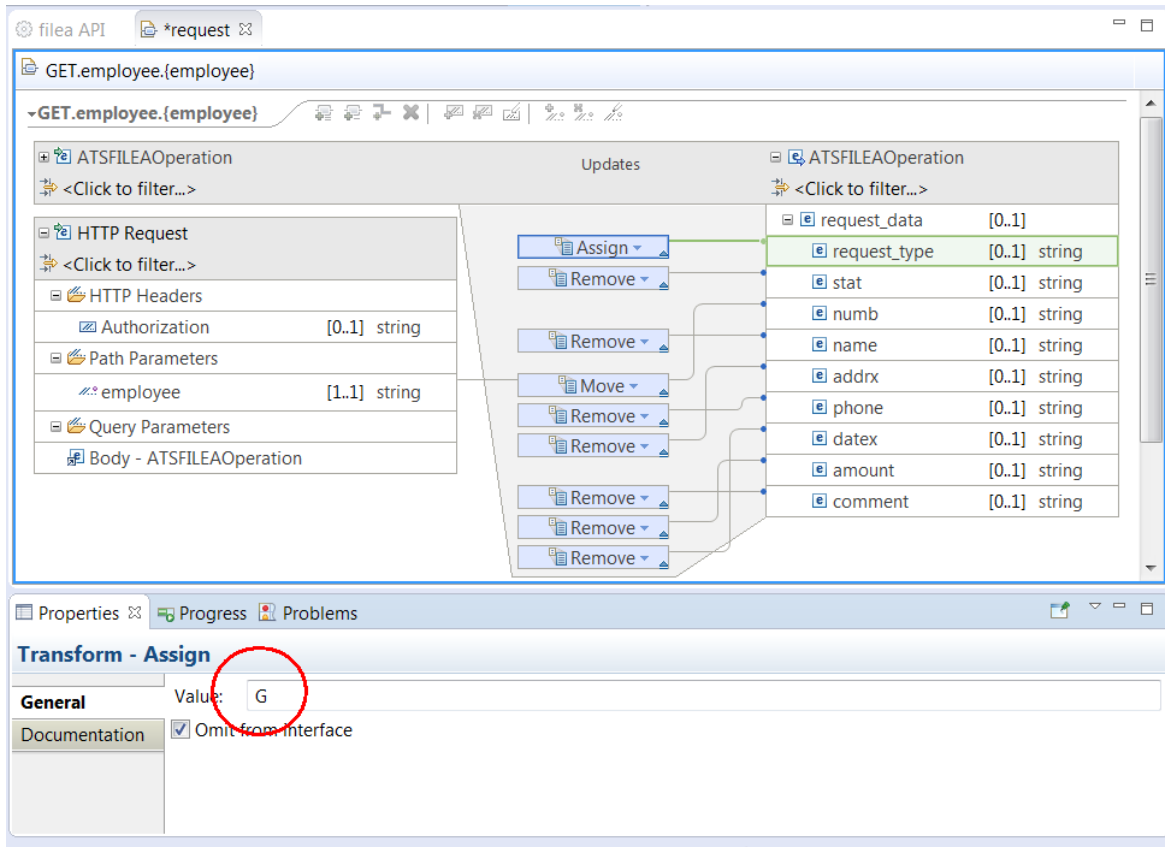
___14. Use the **Ctrl-S** key sequence to save all changes and close the *GET.employee.{employee}* view.

___15. This same pattern used Steps 6 through 13 are repeated to configure the request mapping for the **DELETE** method. All of the fields except *numb* and *request_type* are not exposed to the REST client. The path parameter *employee* is use to set the *numb* field and value of *request_type* is set to **D** for the **DELETE** method. Go ahead and repeat these steps to configure the request mapping for the **DELETE** method.

___16. Request mapping for the **PUT** method differs from **GET** and **DELETE** methods in the respect that we want the REST client to be able to supply all of the fields that were explicitly removed for the **GET** and **DELETE** methods. Update the request mapping for the **PUT** methods but this time do not remove any fields and assign the value **U** to the *request_type* field.   The results should look like the mapping below.

___17. The response mappings for these methods are also different. For the **PUT** and **GET** methods all fields should exposed to the REST client. The default response mapping will return all fields so no special mapping is required for these methods so no response mapping changes are really required. But this is not the case for the **DELETE** method. Back on the *z/OS Connect EE API Editor view* click the **Mapping** button beside the **DELETE** method and select the *Open Response Mapping* option.
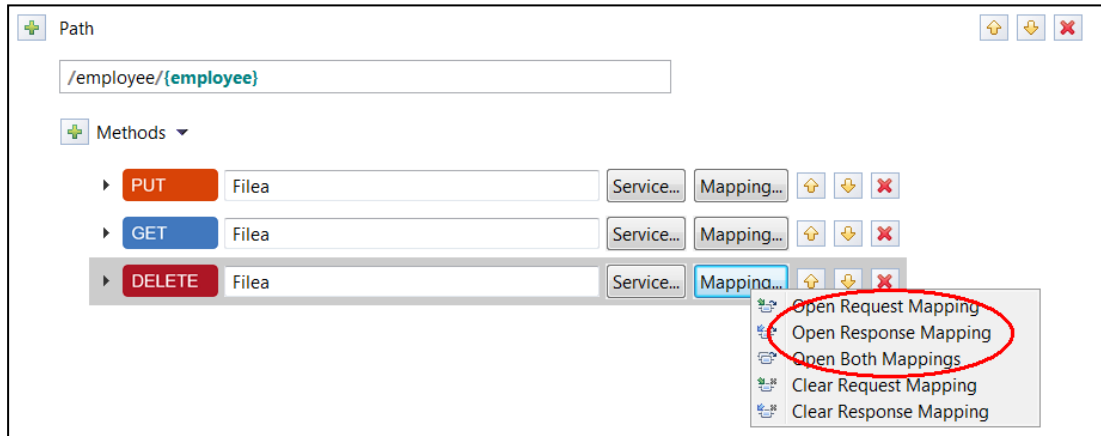
___18. This will open the *DELETE.employee.{employee}* view (see below). Expand the *response_data* structure by click the plus sign beside the structure and expand the view so all fields are completely visible.
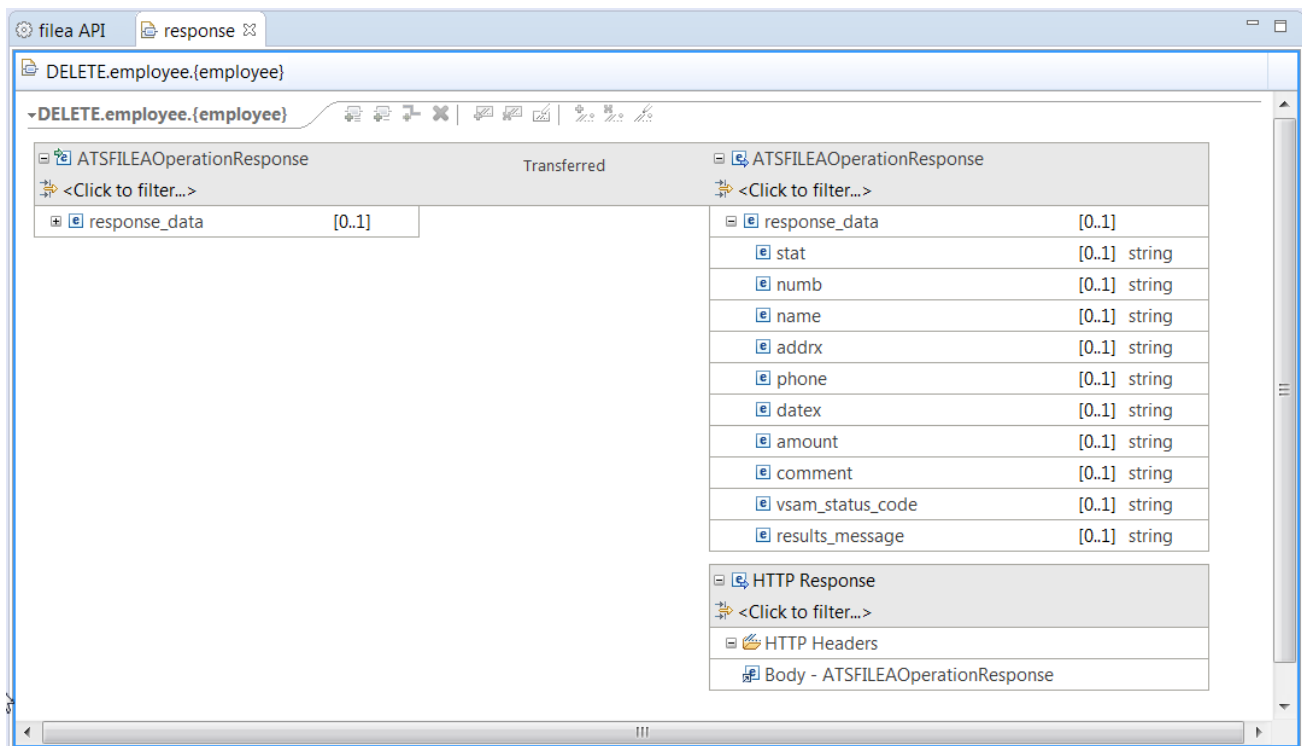
The response_data structure was built using the contents of the FILEARSP COPYBOOK

```
01 RESPONSE-DATA.
   10 stat            PIC X.
   10 numb            PIC X(6).
   10 name            PIC X(20).
   10 addrx           PIC X(20).
   10 phone           PIC X(8).
   10 datex           PIC X(8).
   10 amount          PIC X(8).
   10 comment         PIC X(9).
   10 vsam-status-code PIC X(2).
   10 results-message  PIC X(20).
```

___19. The only fields that should be exposed to the REST client are *vsam_status_code* and *results_message*s. As before use the left mouse button to draw a dotted line box around all of the other fields to do a multiple selection of the other fields.
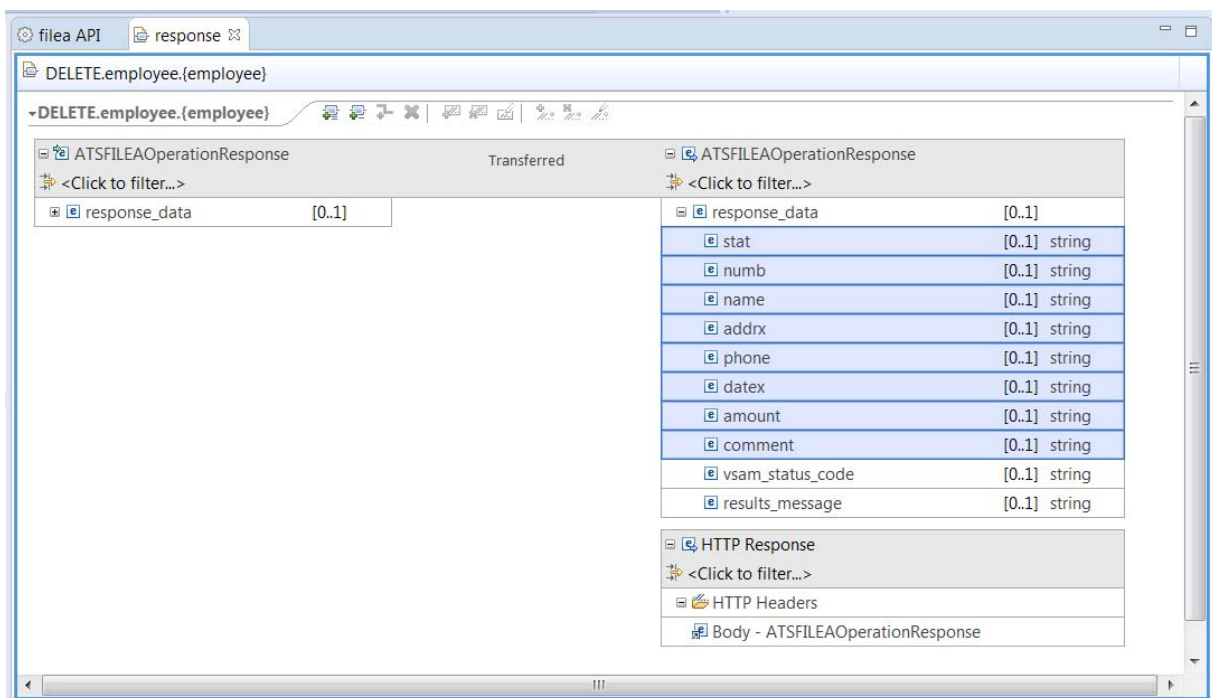
___20. Right mouse button click on one of the selected fields and select the *Add Remove transform* option to remove all of these fields from the response. The REST client will not see these fields in the response from a **DELETE** request.
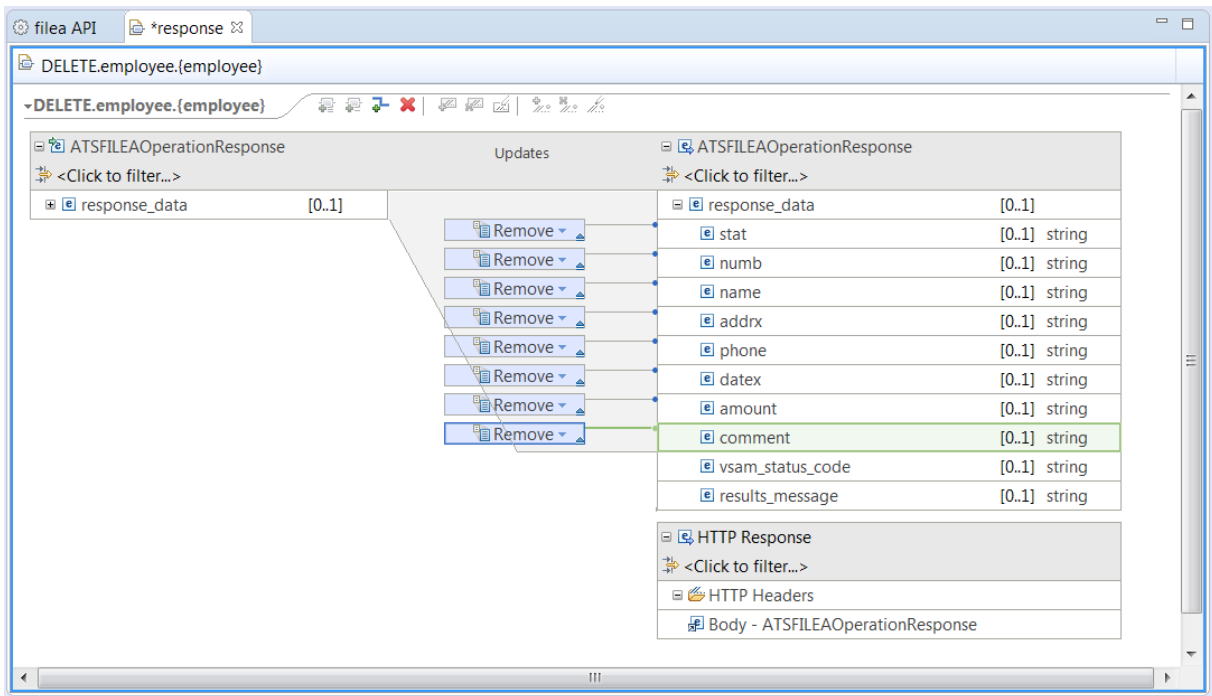
___21. Save all changes with the **Ctrl-S** key sequence and close the response view.

___22. Next we want to add a Path for a **POST** method request. The value for the numb field is included in the JSON request payload, not the URL so no path parameter is required. Click the plus icon beside Path on the z/OS Connect EE API Editor view to add another path to the API.
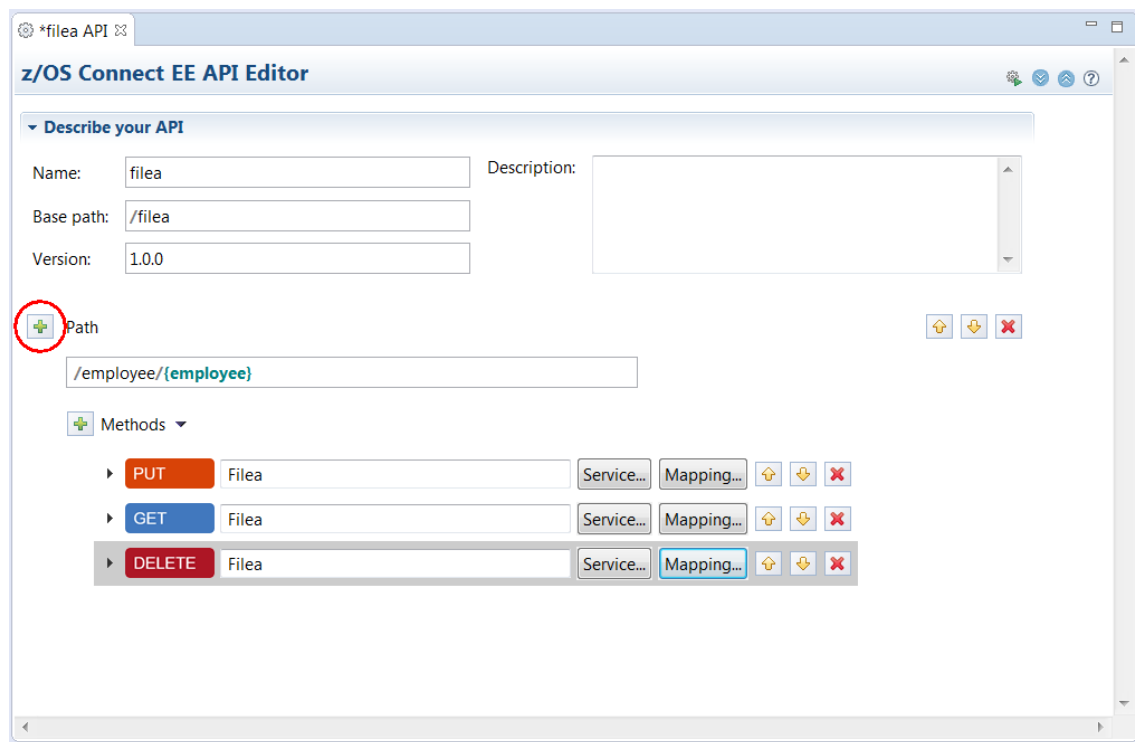
The result is another full set of methods for the new *PATH*.

___23. Enter a path value of ==/employee== and remove the **GET**, **PUT** and **DELETE** methods.

> **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*.

___24. Click the **Service** button beside **POST** and select the *Filea* service:

___25. Save the changes by using the key sequence **Ctrl-S**.

___26. Click on *Mapping* → *Open request mapping*.

___27. Assign a value of *P* to the *request_type* field in the *Property* tab. All of the fields will be exposed to the REST client so no further action is required.

___28. The screen below shows the final result.
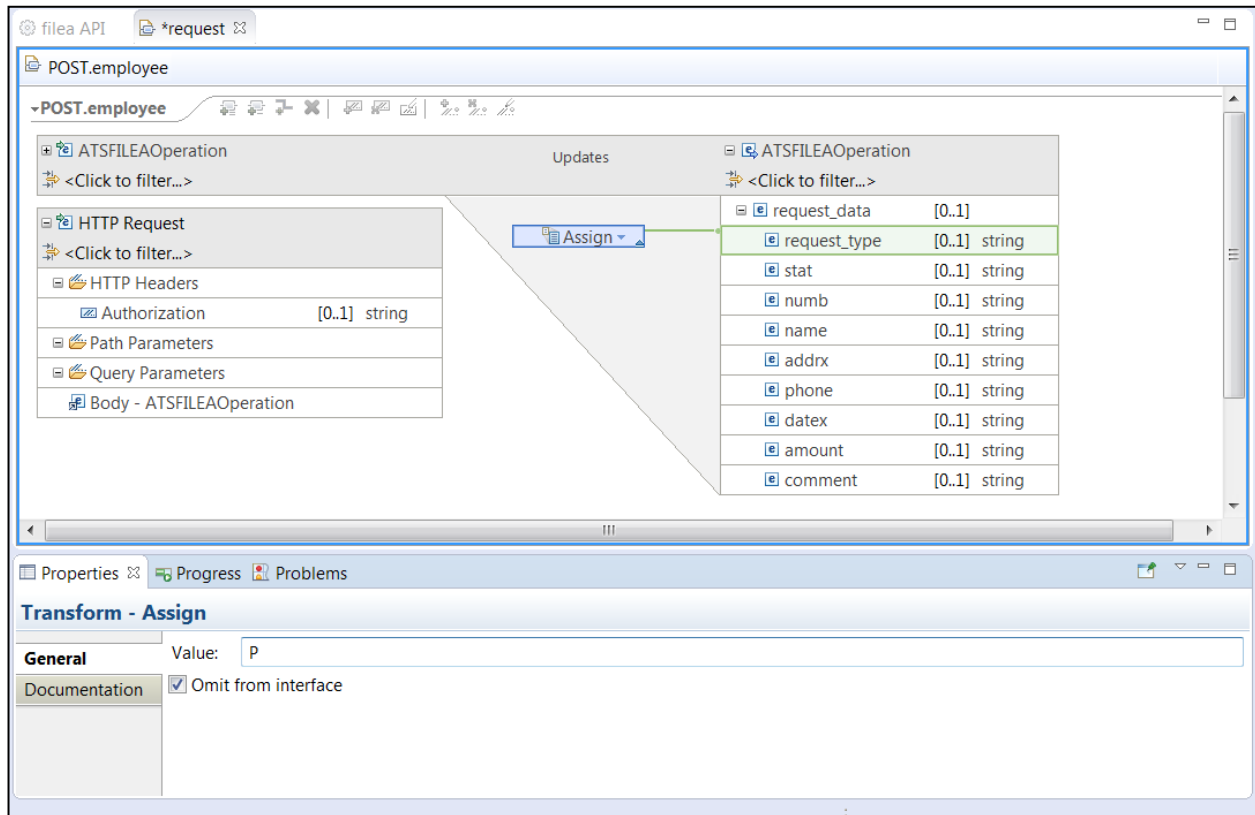
___29. Save the the changes using the key sequence **Ctrl-S**.

___30. Close the *request* mapping tab.

___31. No changes are required to the response mapping since we want all fields returned to the client.

Summary

You created the API, which consists of two paths and the request and response mapping associated with each. That API will now be deployed into z/OS Connect EE V3.0.

Deploy the API to a z/OS Connect EE Server

Before deploying the API review the configuration required to support this API.

___1. The **Filea** service was defined by the inclusion of file *filea.xml* in the *server.xml* we saw earlier.

```
<server description="BATCH VSAM">

 <zosconnect_zosConnectAPIs location="">
        <zosConnectAPI name="filea"/>
 </zosconnect_zosConnectAPIs>

 <zosconnect_localAdaptersConnectService id="wolaFilea"
        registerName="FILEAZCON"
        serviceName="Filea"
        connectionFactoryRef="wolaCF" />

 <zosconnect_zosConnectService id="filea"
        requireAuth="false"
        requireSecure="false"
        serviceName="Filea"
        serviceRef="wolaFilea"
        dataXformRef="xformJSON2Byte"
        serviceDescription="FileA batch program access"/>

</server>
```

Figure 1 - filea.xml

- The *zosConnectAPIs* element identifies the *location* of the *filea* **API ar**chive (AAR) file. Specify a null string set the location to the default which for this server this would be:

     /var/ats/zosconnect/servers/zceesrvr/resources/zosconnect/apis

- The *localAdapterConnectionService* element provides the WOLA register name that will be used for communications with the batch job, e.g. *FILEAZCON*.

- The *zosConnectService* element provides security constraints (none in this case) and the identity of the *dataXformRef* element that provides the location of API artifacts

___2. In the *Project Explorer* view (upper left), right-mouse click on the *Filea* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.
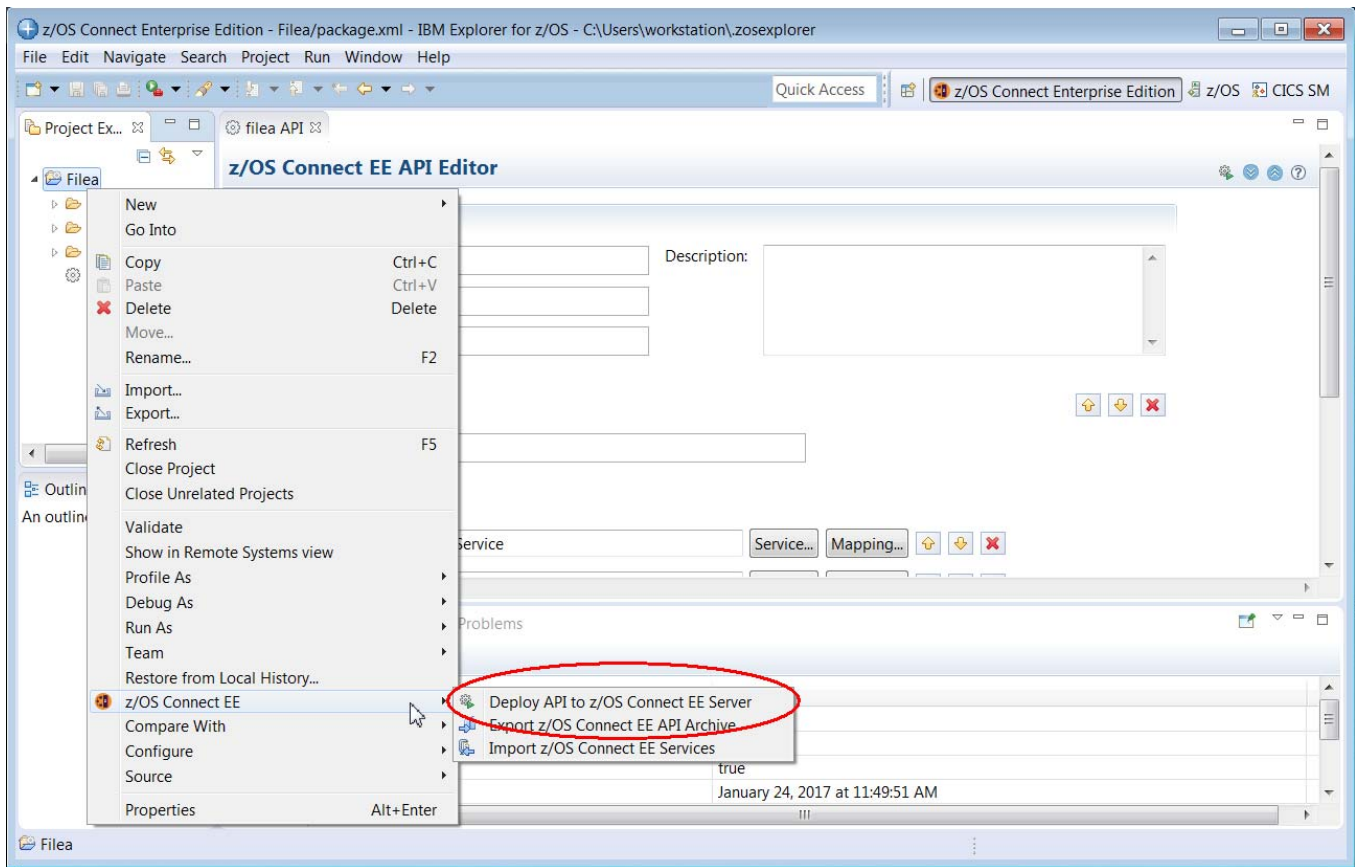
___3. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.
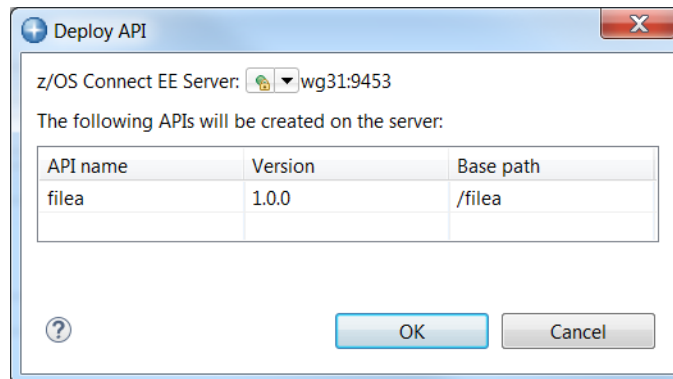
___4. The API artifacts will be transferred to z/OS in an API archive (AAR) file and copied into the */var/ats//zosconnect/servers/zceesrvr/resources/zosconnect/apis* directory.
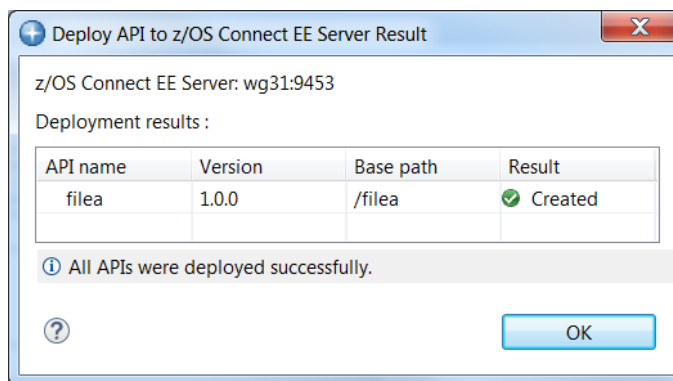
You should see the following message in the STDOUT output of the ZCEESRVR server.

BAQD0007I: BAQR7000I: z/OS Connect API package filea installed successfully.

Hint: Create a *New JES Job Filter* (see below) for the *Active Jobs* in *Remote System Explorer* perspective of the IBM z/OS Explorer and select the STDOUT output for server ZCEESRVR.
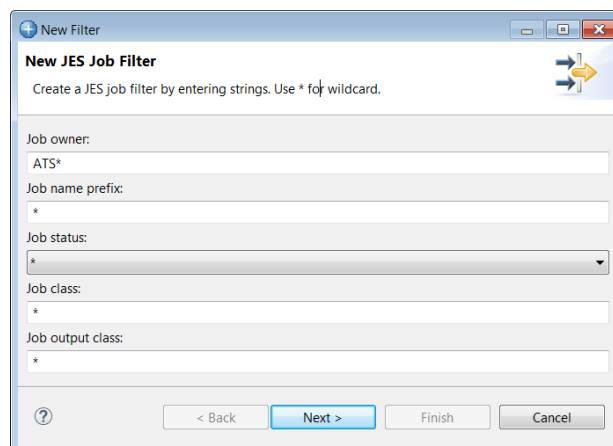
Test the APIs with the MVS batch program

The MVS batch application used for testing the API is a simple COBOL application that accesses a VSAM file. It supports 4 RESTful APIs by providing a *request_type* field in the request, adding a record (**P** for **POST**), updating a record (**U** for **PUT**), retrieving a record (**G** for **GET**) and deleting a record (**D** for **DELETE**). The JCL for this MVS batch application is below:

```
//VERIFY EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//FILEA    DD DISP=SHR,DSN=USER1.ZCEE.FILEA
//SYSIN    DD *
  VERIFY FILE(FILEA)
//ATSFILEA EXEC PGM=ATSFILEA,TIME=1440
//STEPLIB  DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB
//FILEA    DD DISP=SHR,DSN=USER1.ZCEE.FILEA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

To test the z/OS Connect API with this batch application a browser plugin was pre-installed into the Firefox browser for the workstation.

___1. The batch application should already be active. Use the *Remote System Explorer* perspective in z/OS Explorer to display the list of active jobs by expanding the *JES -> My Jobs* (see below). Expand the *RUNFILEA* task (see below) and open the SYSOUT output by double clicking. You should see the messages below indicating the successful registration of the batch job with the z/OS Connect server using the WOLA interface.
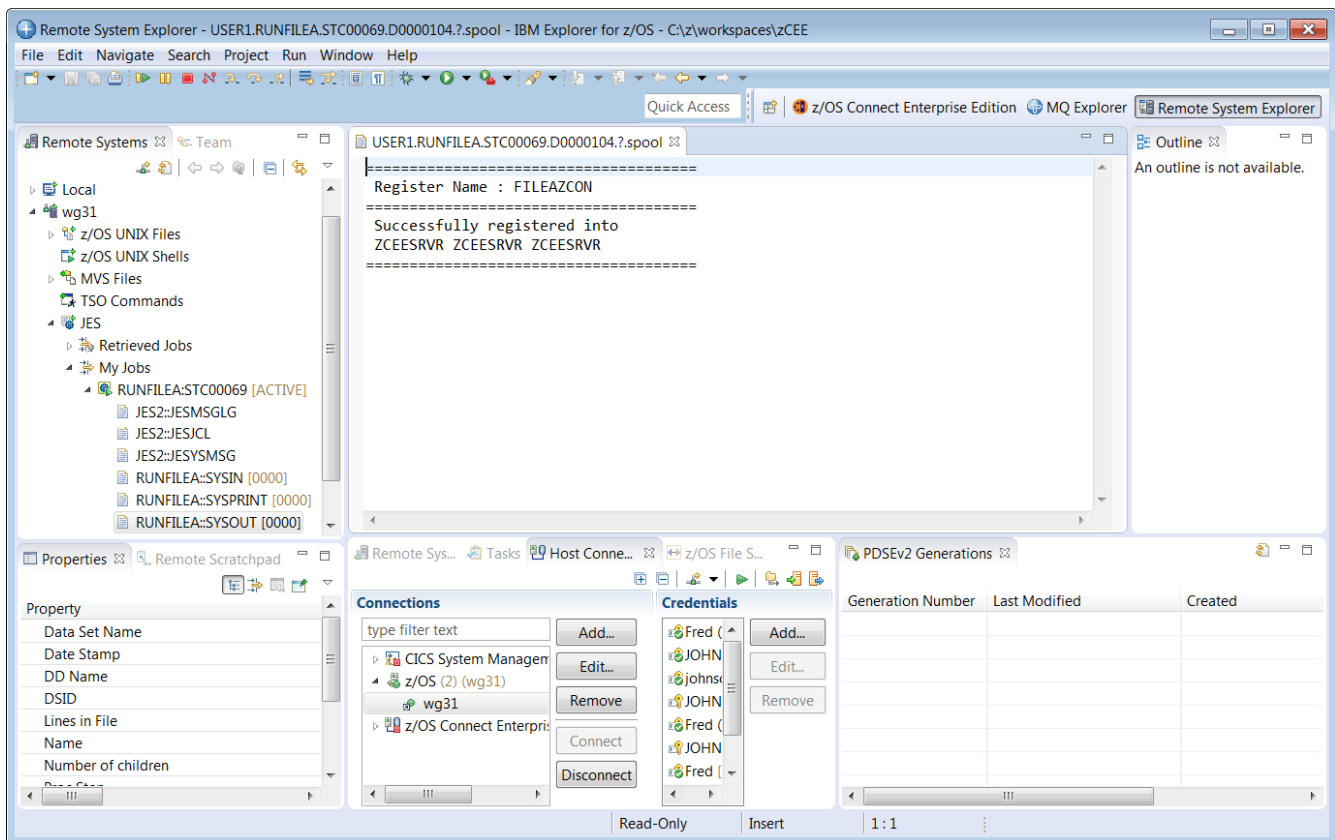
Tech-Tip: The register name of *FILEAZCON* and the use of *ZCEESRVR* must match the names used in the *zosLocatAdatpers* element of the *server.xml* file and the *zosconnect_localAdapterConnectService* element in *filea.xml*.

Important: As the API requests are processed by this batch job this same output will provide a trace or record of the functions being performed. You may want to refer back to this output as you perform the subsequent steps.

___2. Next enter URL ***https://wg31.washington.ibm.com:9453/zosConnect/apis*** in the Firefox browser and you should see the window below. The *filea* API now shows where none were displayed before. This is because this API has now been 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.

Mitch Johnson (mitchj@us.ibm.com)
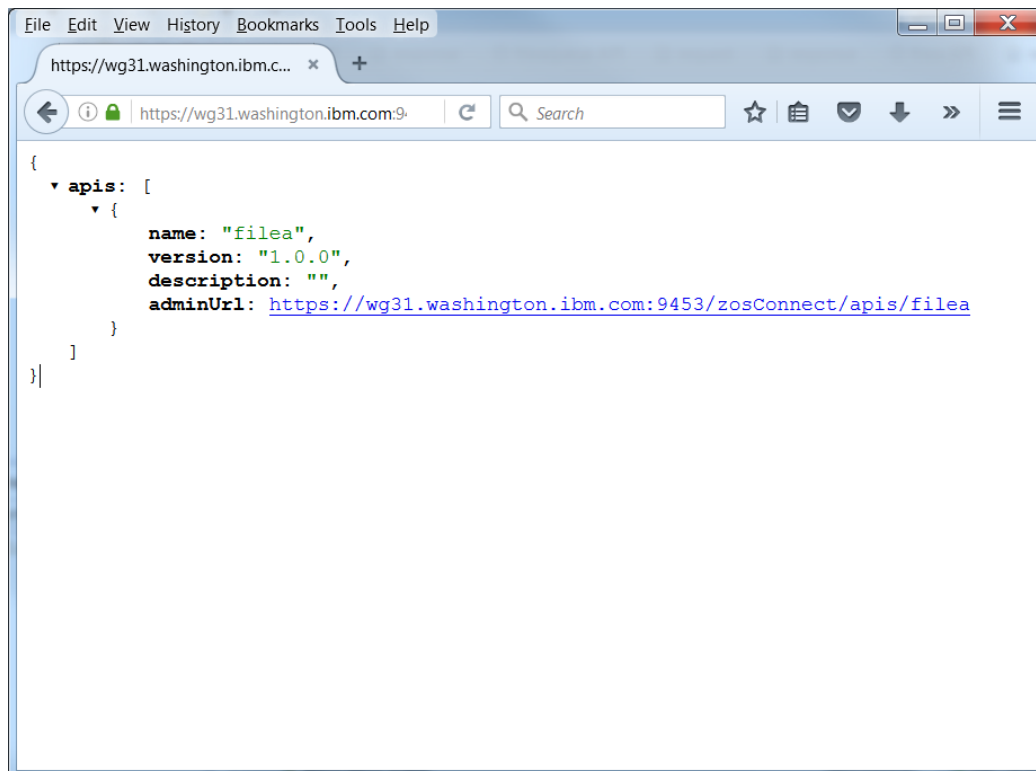
___3. If you click on *adminUrl* URL and the window below should be displayed:

```
{
    name: "filea",
    version: "1.0.0",
    description: "",
    status: "Started",
    apiUrl: https://wg31.washington.ibm.com:9453/filea,
    ▼ documentation: {
        swagger: https://wg31.washington.ibm.com:9453/filea/api-docs
    },
    ▼ services: [
        ▼ {
            name: "Filea",
            uri: https://wg31.washington.ibm.com:9453/zosConnect/services/Filea
        }
    ]
}
```
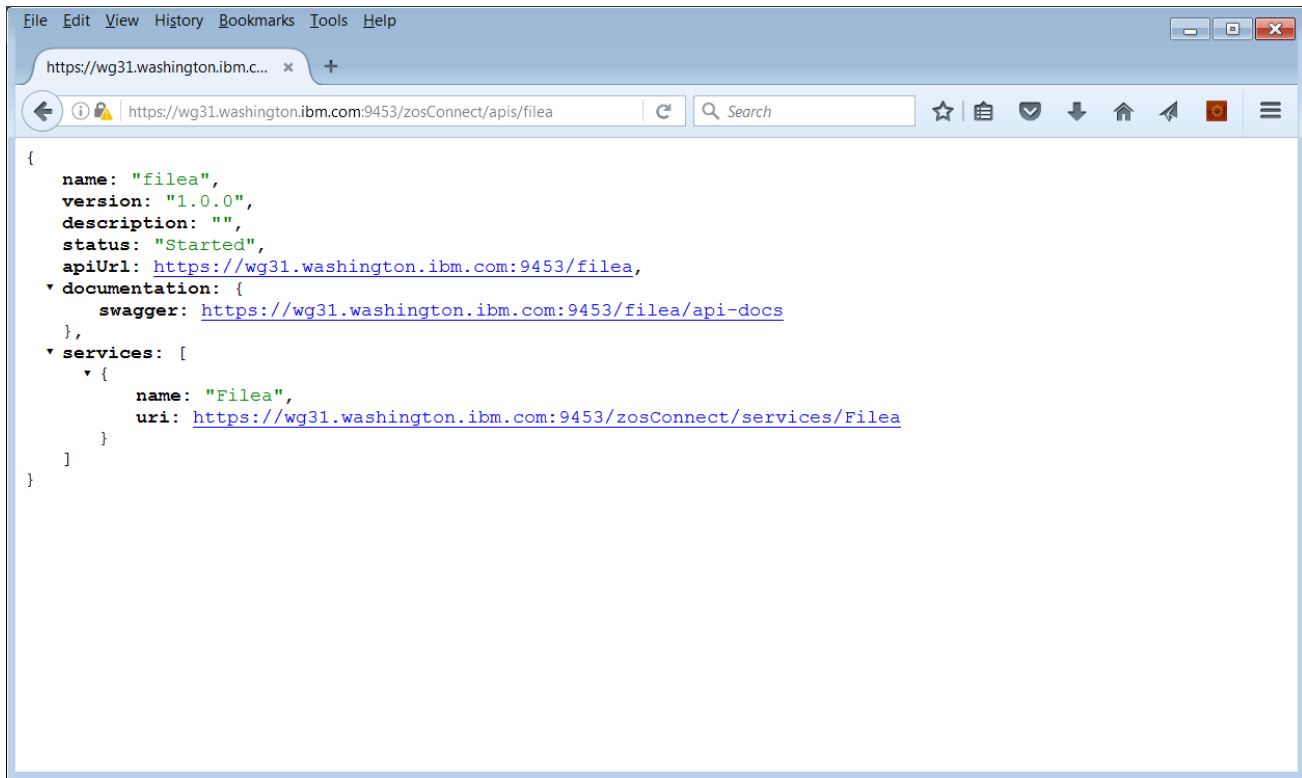
___4. Finally click on the *swagger* URL and you should see the Swagger document associated with this API.

```
{
    swagger: "2.0",
    ▼ info: {
        description: "",
        version: "1.0.0",
        title: "filea"
    },
    basePath: "/filea",
    ▼ schemes: [
        "https",
        "http"
    ],
    ▼ consumes: [
        "application/json"
    ],
    ▼ produces: [
        "application/json"
    ],
    ▼ paths: {
        ▼ "/employee": {
            ▼ post: {
                operationId: "postFileaService1",
                ▼ parameters: [
                    ▼ {
                        in: "body",
                        name: "postFileaService1_request",
```
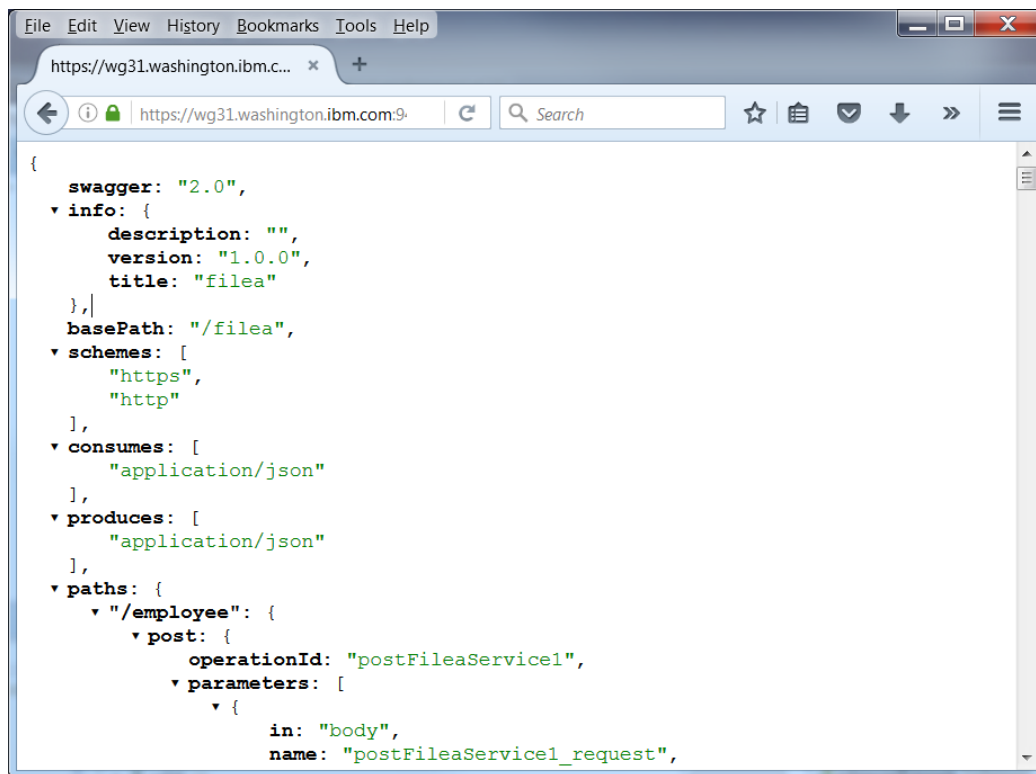
Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This Swagger document can be used by a developer or other tooling to develop REST clients for this specific API.

___5. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect EE Servers*.

> **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.

Expand *wg31:9453* and the expand the *APIs* folder. You should see a list of the APIs installed in the server.
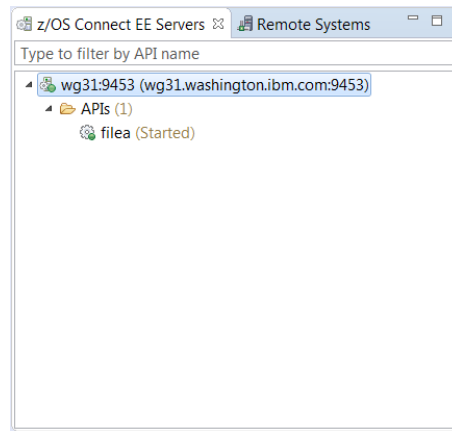
___6. Right mouse button click on *filea* and select *Open in Swagger UI*. Click OK if an informational prompt appears. This will open a Firefox window showing a *Swagger* test client (see below).
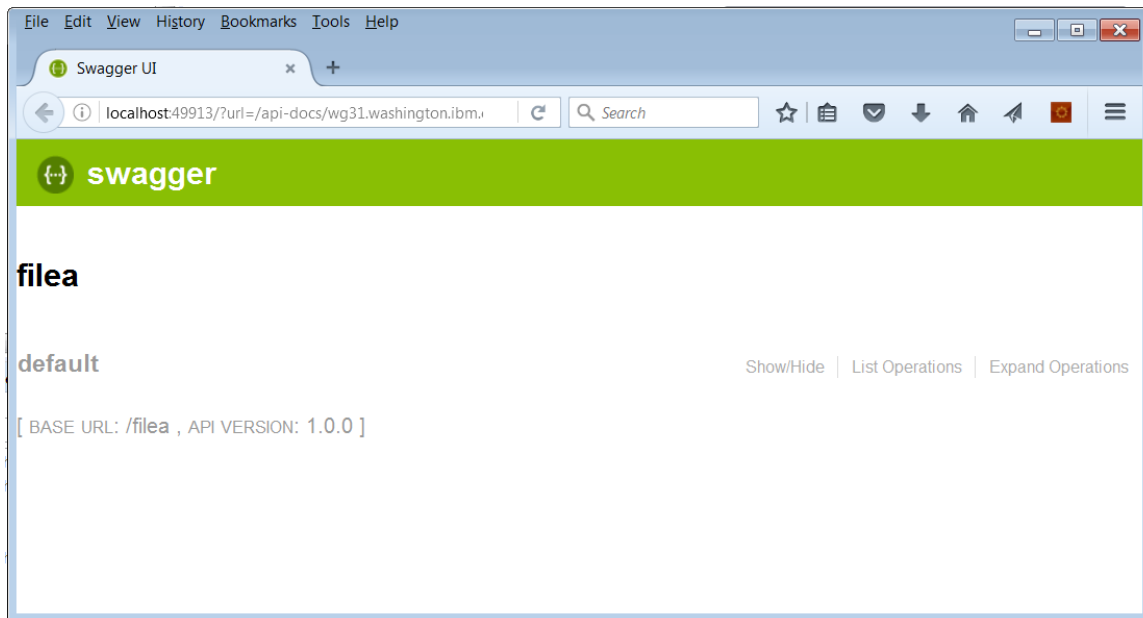
___7. Click the *List Operations* and the browser should show a list of the available HTTP methods like this:
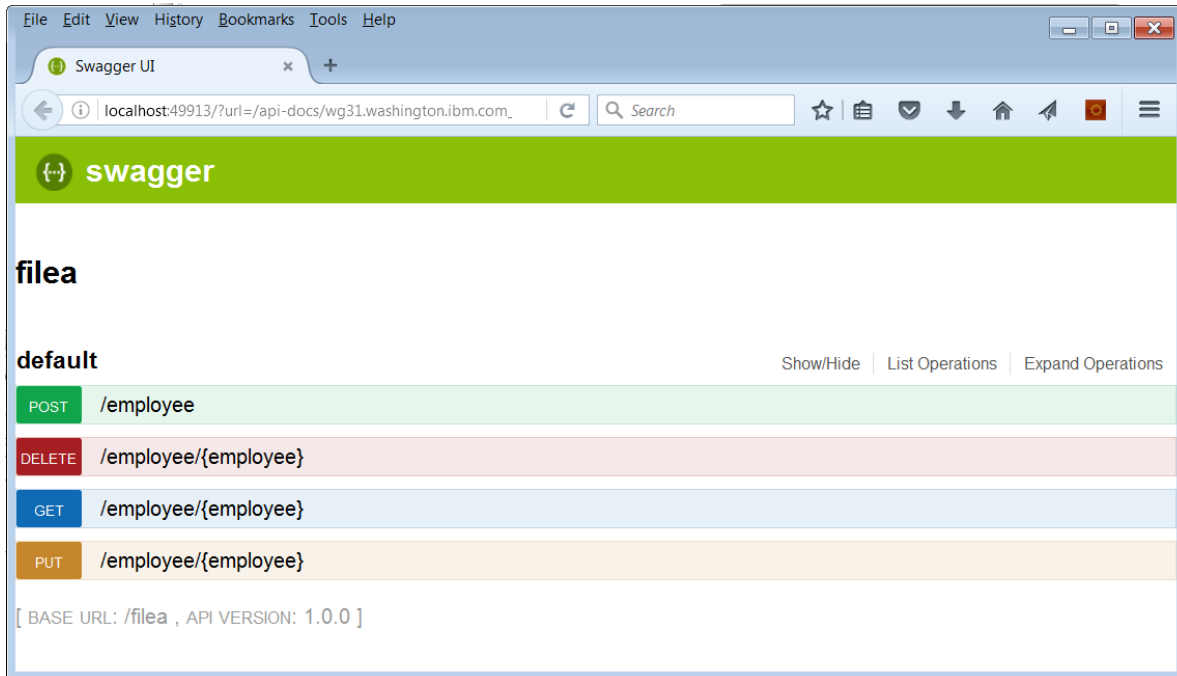
___8. Expand the *GET* method by clicking on the path beside it (e.g. */employee/{employee})*and scroll down until the method *Parameters* are displayed as shown below:

Mitch Johnson (mitchj@us.ibm.com)
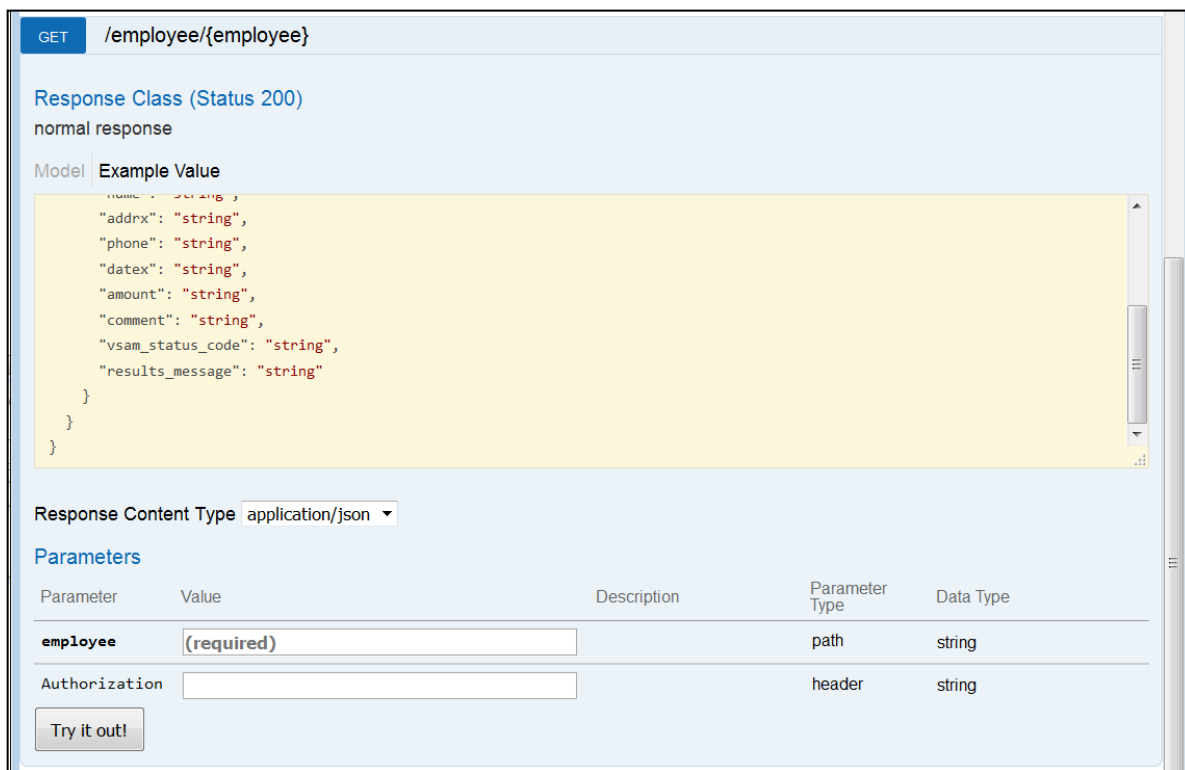
___9. Enter *000100* in the area beside employee and click the **Try it out!** button.

___10. You should see a *200 OK status code* in the *Response Header* area and a display of the VSAM record in the *Response Body (Preview)* area along with *GET was successful* in the *results_message* field.

```
Response Body

  {
    "ATSFILEAOperationResponse": {
      "response_data": {
        "stat": "",
        "addrx": "SURREY, ENGLAND",
        "amount": "$0100.11",
        "vsam_status_code": "00",
        "results_message": "GET successful",
        "phone": "32156778",
        "datex": "26 11 81",
        "name": "S. D. BORMAN",
        "comment": "*********",
        "numb": "000100"
      }
    }
  }

Response Code

  200
```

___11. Enter *001000* and press the **Try it Out!** button again and should see results that no record was found.

```
Response Body

  {
    "ATSFILEAOperationResponse": {
      "response_data": {
        "stat": "",
        "addrx": "",
        "amount": "",
        "vsam_status_code": "23",
        "results_message": "No record found",
        "phone": "",
        "datex": "",
        "name": "",
        "comment": "",
        "numb": "001000"
      }
    }
  }

Response Code

  200
```

___12. Expand the **DELETE** method and try deleting employee ***001000***.

Note that the fields in the DELETE response are different from the fields in the GET request. Remember that each method has a different response map.

___13. Repeat the **DELETE** method with a record that you know exists, e.g. 000100 and the observe the results, see below:

Response Body

```
{
  "ATSFILEAOperationResponse": {
    "response_data": {
      "vsam_status_code": "00",
      "results_message": "DELETE successful"
    }
  }
}
```

Response Code

```
200
```
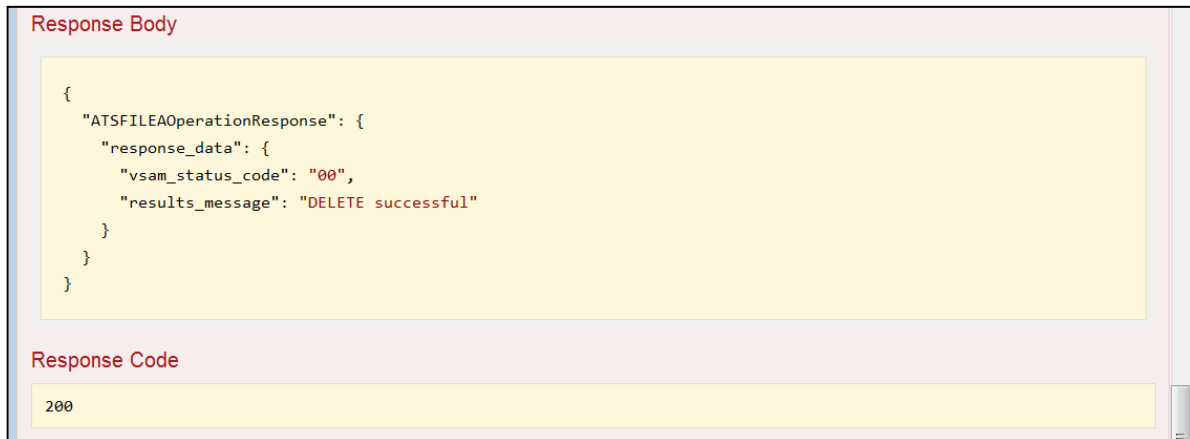
Repeat the **GET** and **DELETE** methods with other records (see table below) and verify the results are as expected.

Note that the **GET** and **DELETE** methods do not required JSON in the *Request*. Only the path variable *employee* was required.

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	*********

___14. Expand **POST** method and enter information like below in the request_data area.

```
postFilea_request                           request   body       Model  Example Value
                    -                        body
                                                                  {
          Request schema for the ATSFILEA JSON interface             "ATSFILEAOperation": {
                                                                       "request_data": {
             ATSFILEAOperation    -                                      "stat": "string",
                                                                         "numb": "string",
                                                                         "name": "string",
                request_data   -                                         "addrx": "string",
                                                                         "phone": "string",
                  stat                                                   "datex": "string",
                  [            ]                                         "amount": "string",
                  numb                                                   "comment": "string"
                  [948480     ]
                  name
                  [Don Bagwell ]
                  addrx
                  [Raleigh, NC ]
                  phone
                  [32156778   ]
                  datex
                  [26 11 81   ]
                  amount
                  [$0100.11   ]
                  comment
                  [            ]

          Parameter content type: application/json ▼

   Authorization       [                          ]    header    string
   [Try it out!]   Hide Response
```

___15. Press the **Try it out!** button and you should see a response that the POST was successful.

```
Response Body

   {
     "ATSFILEAOperationResponse": {
       "response_data": {
         "stat": "",
         "addrx": "Raleigh, NC",
         "amount": "$0100.11",
         "vsam_status_code": "00",
         "results_message": "POST successful",
         "phone": "32156778",
         "datex": "26 11 81",
         "name": "Don Bagwell",
         "comment": "",
         "numb": "948480"
       }
     }
   }
```

___16. Expand the **PUT** method and enter different information for name, address, phone number, etc. for employee 948480 and press the ***Try it Out!*** button. Remember the **PUT** method obtains the value of the *numb* field from the path parameter not the JSON body. You should see a *PUT successful* information and when you retrieve this record with a *GET* the changed fields should be displayed.

Summary

You have verified the API. The API layer operates above the service layer you defined and tested earlier. The service layer is what does the data conversion and mapping to WOLA and 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.