IBM z/OS Connect Enterprise Edition

Introduction and Overview

Mitch Johnson
mitchj@us.ibm.com
Washington System Center

IBM Z
Wildfire Team –
Washington System Center

Agenda

- An Introduction and Overview of using REST API
- Enabling RESTful API to various z/OS resources, e.g.
  - CICS
  - Db2
  - IMS/TM
  - IMS/DB
  - MQ
  - MVS Batch
  - Outbound REST APIs
  - IBM DVM
  - Host Access Transformation Services (3270 screen-based applications)
  - IBM File Manager
- Accessing RESTful API from z/OS COBOL Applications
- A brief overview of z/OS Connect Security*

*For more on security, contact your local IBM rep regarding the schedule of workshop *zCEESEC IBM z/OS Connect Administration/Security Wildfire Workshop*

mitchj@us.ibm.com

z/OS Connect EE exposes z/OS resources to the “cloud” via RESTful APIs

z/OS Connect EE (an API Gateway)

z/OS Connect EE

- CICS
- IMS/TM
- IMS/DB
- Db2
- MQ
- IBM File Manager+
- HATS(3270)
- IBM DVM+
- MVS
- WAS
- Custom*

+ HCL and Rocket Software
*Other Vendors or your own implementation

mitchj@us.ibm.com

/but_first, what_is_REST?

What makes an API “RESTful”?

mitchj@us.ibm.com

REST is architectural programming style

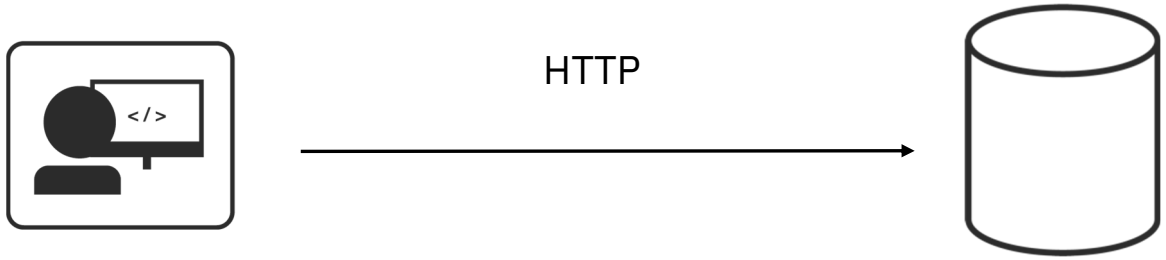
z/OS Connect EE

REST stands for **Re**presentational **S**tate **T**ransfer.

An architectural programming style for **accessing** and **updating** data over the internet.

Typically using HTTP… but not all HTTP interfaces are “RESTful”.

Simple and intuitive for the end consumer (**the developer**).



Roy Fielding defined REST in his 2000 PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures" at UC Irvine. He developed the REST architectural style in parallel with HTTP 1.1 of 1996-1999, based on the existing design of HTTP 1.0 of 1996.

mitchj@us.ibm.com

Key Principles of REST

z/OS Connect EE

Use Path and Query parameters to refine the request

Use HTTP verbs for Create, Read, Update, Delete (CRUD) operations

POST
GET
PUT
DELETE

```
http://<host>:<port>/path/parameter?name=value&name=value
```

URI path identifies a resource (or lists of resources)

URL identifies the protocol, host and port and includes the URI Path

Request/Response Body is used to represent the data object

```
GET http://www.acme.com/customers/12345?personalDetails=true
RESPONSE: HTTP 200 OK
BODY { “id” : 12345
       “name” : “Joe Bloggs”,
       “address” : “10 Old Street”,
       “tel” : “01234 123456”,
       “dateOfBirth” : “01/01/1980”,
       “maritalStatus” : “married”,
       “partner” : “http://www.acme.com/customers/12346” }
```

mitchj@us.ibm.com

REST vs RESTful

REST is an architectural style of development having these principles plus..

- It should be stateless (transaction management should be managed by the client)
- It should access and/or identify all server resources using only a single URI
- For performing CRUD operations, it should use HTTP verbs such as get, post, put and delete
- It should return the result only in the form of consistent and simple JSON

When an API follows these basic principles, it is considered a RESTful API, whereas a REST API only follows some but not all the above principles

- Remember - Not all REST APIs are RESTful APIs
- The key is consistency, RESTful APIs are consistent with these basic principles, REST APIs are not

mitchj@us.ibm.com

RESTful Examples

z/OS Connect EE

POST `/account/ +`  *(a JSON request message with Fred's information)*

GET `/account?number=1234`

PUT `/account/1234 +`  *(a JSON request message with dollar amount of deposit)*

HTTP Verb conveys the method against the resources; i.e., POST is for create, GET is for balance, etc.

URI conveys the resource to be acted upon; i.e., Fred's account with number 1234

The JSON body carries the specific data for the action (verb) against the resource (URI)

REST APIs are increasingly popular as an integration pattern because it is stateless, relatively lightweight, is relatively easy to program

https://martinfowler.com/articles/richardsonMaturityModel.html

mitchj@us.ibm.com

Not every REST API is a RESTful API

z/OS Connect EE

(How to know if an API is not RESTful)

1. Different URIs with the same method for operations on the same object

```
POST http://www.acme.com/customers/GetCustomerDetails/12345
POST http://www.acme.com/customers/UpdateCustomerAddress/12345?address=
```

2. Different representations of the same objects between request and response messages

```
POST http://www.acme.com/customers
BODY { "firstName": "Joe",
       "lastName" : "Bloggs",
       "addr"     : "10 Old Street",
       "phoneNo"  : "01234 0123456" }
```



```
RESPONSE HTTP 201 CREATED
BODY { "id"      : "12345",
       "name"    : "Joe Bloggs",
       "address": "10 New Street"
       "tel"     : "01234 0123456"}
```

3. Operational data (update, etc.) embedded in the request body

```
POST http://www.acme.com/customers/12345
BODY { "updateField": "address",
       "newValue"   : "10 New Street"}
```



```
RESPONSE HTTP 200 OK
BODY { "id"      : "12345",
       "name"    : "Joe Bloggs",
       "address": "10 New Street"
       "tel"     : "01234 123456" }
```

mitchj@us.ibm.com

Why is REST popular?

z/OS Connect EE

Ubiquitous Foundation	It's based on HTTP, which operates on TCP/IP, which is a ubiquitous networking topology.
Relatively Lightweight	Compared to other technologies (for example, SOAP/WSDL), the REST/JSON pattern is relatively light protocol and data model, which maps well to resource-limited devices.
Relatively Easy Development	Since the REST interface is so simple, developing the client involves very few things: an understanding of the URI requirements (path, parameters) and any JSON data schema.
Increasingly Common	REST/JSON is becoming more and more a de facto "standard" for exposing APIs and Microservices. As more adopt the integration pattern, the more others become interested.
Stateless	REST is by definition a stateless protocol, which implies greater simplicity in topology design. There's no need to maintain, replicate or route based on state.

mitchj@us.ibm.com

How do we describe a REST API?

mitchj@us.ibm.com

/swagger/open_api

The industry standard framework for describing RESTful APIs is by a Swagger document

mitchj@us.ibm.com

Why use Swagger?

z/OS Connect EE

It is more than just an API framework

OPEN API INITIATIVE

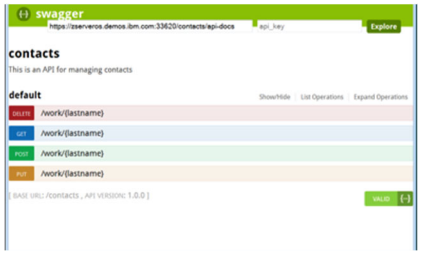
There are a number of tools available to aid consumption:

Consume Swagger*

Swagger Codegen create stub code to consume APIs from various languages

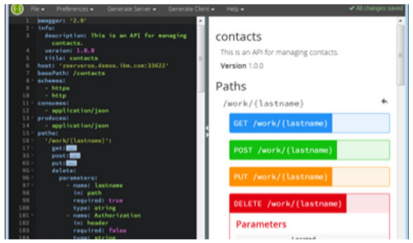
Read Swagger+

Swagger UI allows API consumers to easily browse and try APIs based on Swagger Doc.



Write Swagger

Swagger Editor allows API developers to design their swagger documents.



* z/OS Connect API Requester

+z/OS Connect, MQ REST support, Zowe

mitchj@us.ibm.com

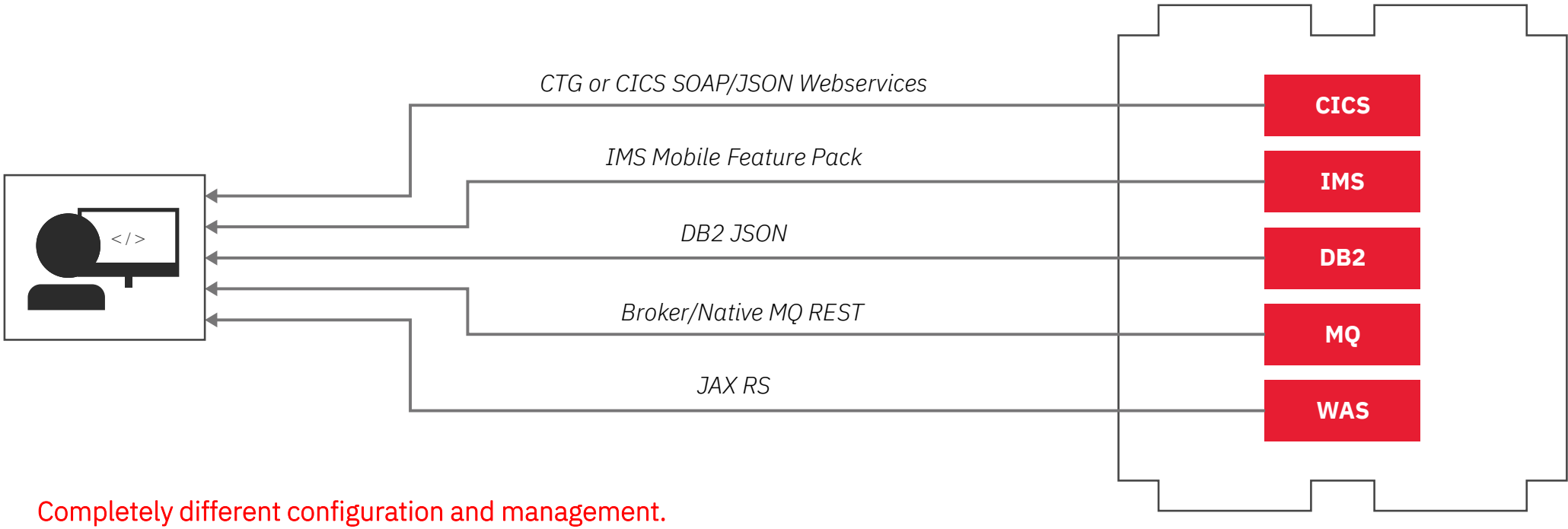
https://blog.readme.io/what-is-swagger-and-why-it-matters/

Why /zos_connect_ee?

Truly RESTful APIs to and from your mainframe.

mitchj@us.ibm.com

There was support for REST before z/OS Connect but..



Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!

mitchj@us.ibm.com

z/OS Connect provides a single-entry point

z/OS Connect EE

To expose z/OS resources without writing any code.

z/OS Connect EE provides

- ❑ Single Configuration Administration
- ❑ Single Security Administration
- ❑ With sophisticated mapping of truly RESTful APIs to existing mainframe and services data without writing any code.

mitchj@us.ibm.com

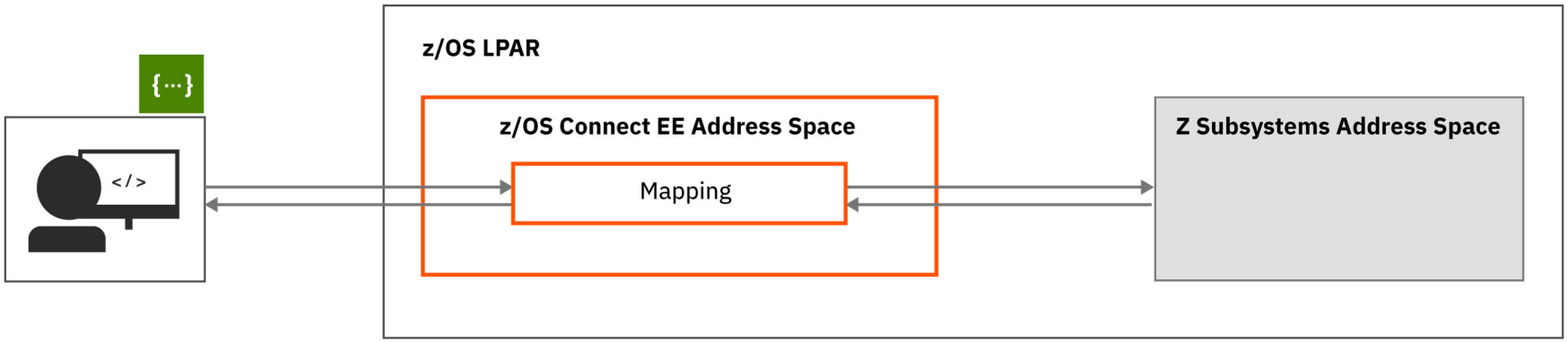
Other than a RESTful interface, what else does z/OS Connect provide?

mitchj@us.ibm.com

Let’s Start with Data mapping

z/OS Connect EE

Converting the JSON message to the format the target’s subsystem expects* .



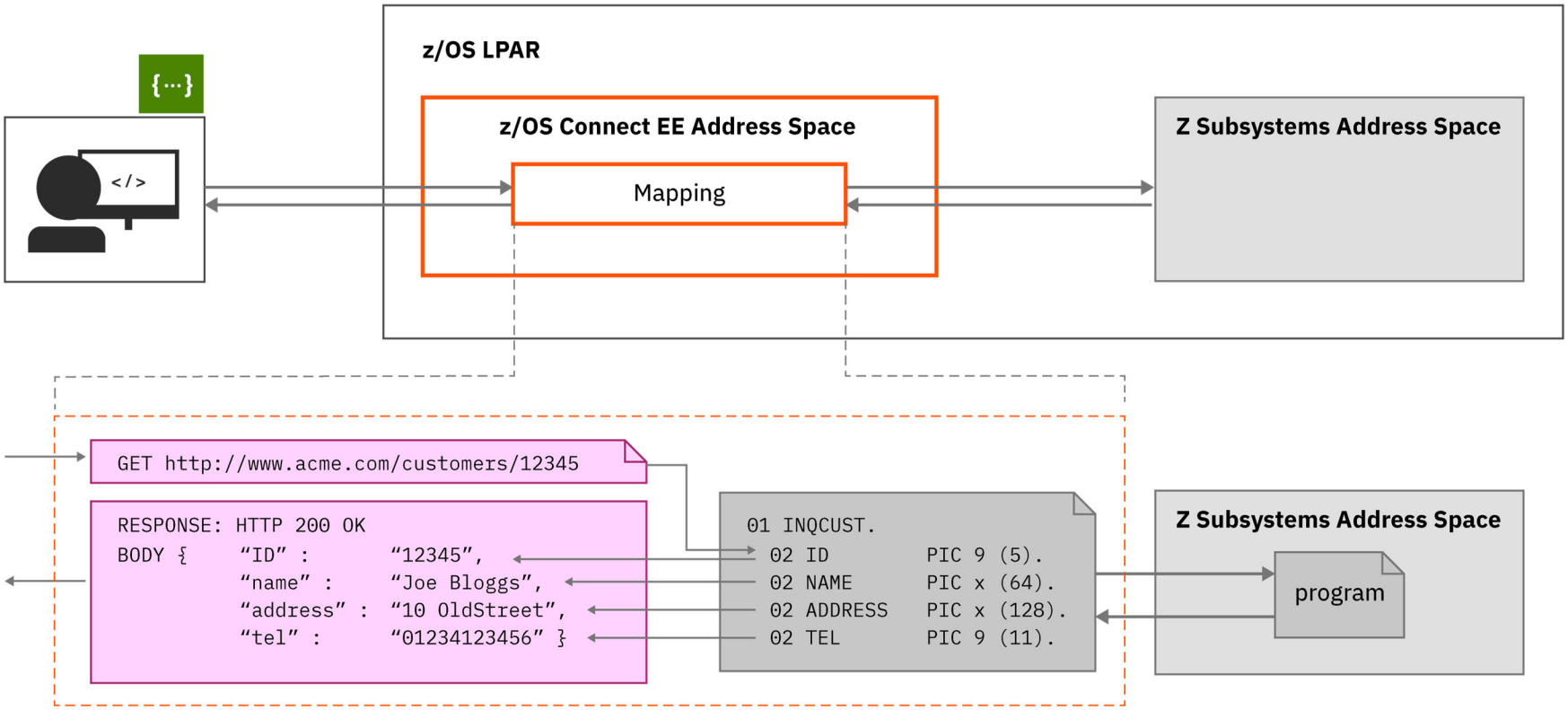
* Most z/OS subsystems depend information in a serial data format and do not normally work with JSON request/response messages. Examples of non-JSON formats are CICS COMMAREAs and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation refers to the process of converting JSON messages to a serialized layout (e.g., sequentially arranged in storage).

mitchj@us.ibm.com

Data mapping Example

A closer look

z/OS Connect EE



mitchj@us.ibm.com

Tech-Tip: Behind the curtains, COBOL to JSON schema example

z/OS Connect EE

```
01 MINILOAN-COMMAREA.
           10 name pic X(20).
           10 creditScore pic 9(16)V99.
           10 yearlyIncome pic 9(16)v99.
           10 age pic 9(10).
           10 amount pic  9999999v99.
           10 approved pic X.
               88 BoolValue value 'T'.
           10 effectDate pic X(8).
           10 yearlyInterestRate pic S9(5).
           10 yearlyRepayment pic 9(18).
           10 messages-Num pic 9(9).
           10 messages pic X(60) occurs 1 to 10 times
                        depending on messages-Num.
```

```
"MINILOAN_COMMAREA" : {
     "type" : "object",
     "properties" : {
       "NAME" : {
         "maxLength" : 20,
         "type" : "string"
       },
       "CREDITSCORE" : {
         "multipleOf" : 0.01,
         "minimum" : 0,
         "maximum" : 9999999999999999.99,
         "type" : "number",
         "format" : "decimal"
       },
```

"name":"Mitch Johnson",
"creditScore":"72000"

All data is sent as character strings, removing the big v. little endian and +/- issue

miniloanRequest.si

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:message xmlns:ns2="http://www.ibm.com/ims/Transaction" transactionCode="" messageName="miniloanRequest" direction="0" serviceType
    <message id="1" name="miniloanRequest">
        <segment id="1" name="COMMAREA" originalName="COMMAREA">
            <field name="MINILOAN_COMMAREA" originalName="MINILOAN_COMMAREA" included="Y" path="MINILOAN_COMMAREA">
                <startPos>1</startPos>
                <bytes>736</bytes>
                <maxBytes>0</maxBytes>
                <applicationDatatype datatype="STRUCT"/>
                <field name="NAME" originalName="NAME" included="Y" path="MINILOAN_COMMAREA.NAME">
                    <startPos>1</startPos>
                    <bytes>20</bytes>
                    <maxBytes>20</maxBytes>
                    <applicationDatatype datatype="CHAR"/>
                </field>
                <field name="CREDITSCORE" originalName="CREDITSCORE" included="Y" path="MINILOAN_COMMAREA.CREDITSCORE">
                    <startPos>21</startPos>
                    <bytes>18</bytes>
                    <maxBytes>18</maxBytes>
                    <marshaller isSigned="N" isSignLeading="N" isSignSeparate="N" isWCHAROnly="N">
                        <typeConverter>ZONEDDECIMAL</typeConverter>
                    </marshaller>
                    <applicationDatatype datatype="DECIMAL" precision="18" scale="2"/>
                </field>
```

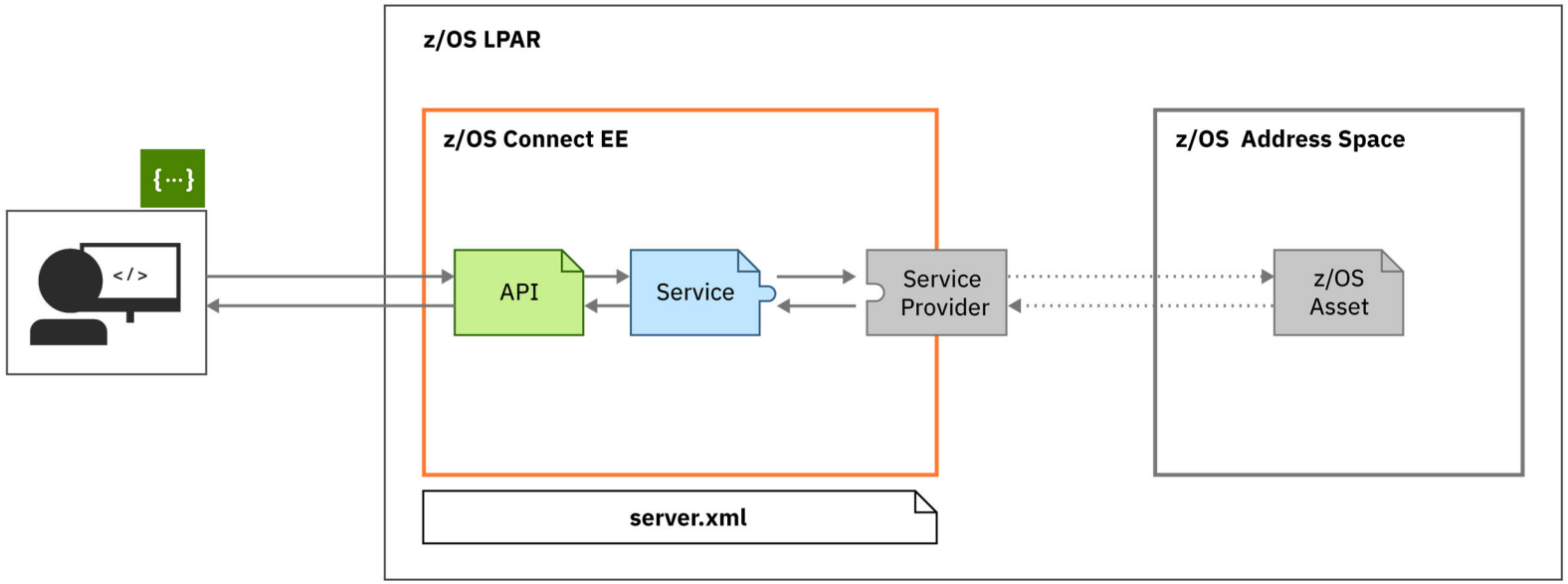
https://www.ibm.com/support/knowledgecenter/en/SSEPEK_10.0.0/char/src/tpc/db2z_endianness.html

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

z/OS Connect does not use a single monolithic interface – but rather separate plug-and-play components



- The API provides the RESTful interface is ready to be consumed by a client and it requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port)

mitchj@us.ibm.com

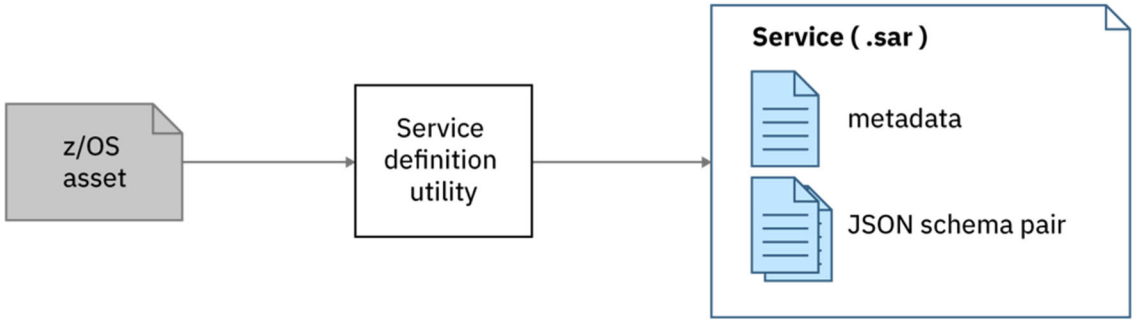
Steps to expose a z/OS application

z/OS Connect EE

Start by creating a service to represent an interaction with the resource

To start mapping an API, z/OS Connect EE needs a representation of the underlying z/OS application: in a **Service Archive file** (`.sar`).

The metadata consists of data mapping XML and provider specific configuration information




```
EVALUATE ACTION of Request-Container
    WHEN 'D'
      PERFORM Delete-Record
    WHEN 'I'
      PERFORM Insert-Record
    WHEN 'U'
      PERFORM Update-Record
    WHEN 'S'
      PERFORM Select-Record
END-EVALUATE.
```

Use a system-appropriate utility to generate a service archive file for the z/OS application

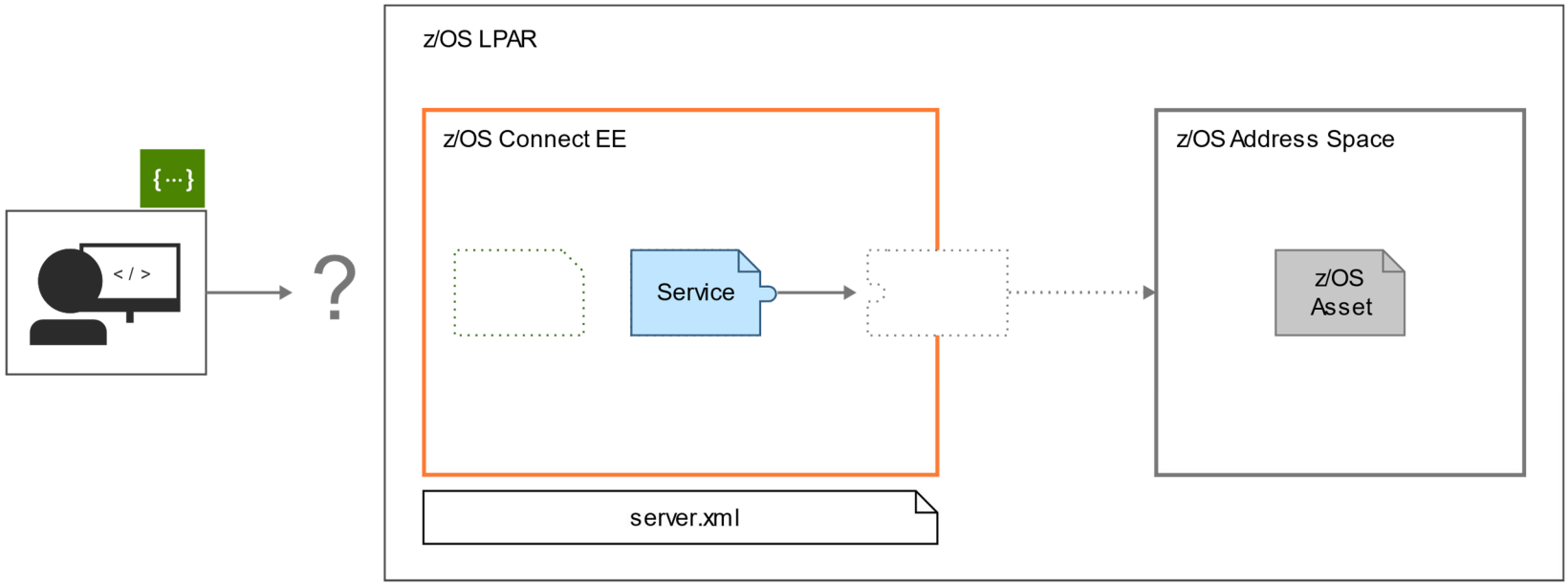
- Eclipse based - API Toolkit (CICS, IMS TM, IMS DB, Db2 and MQ)
- Command line - z/OS Connect EE Build Toolkit (MVS Batch, IBM File Manager and HATS)
- Eclipse based - DVM Toolkit

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

Deploy and export the service



Deploy the service archive file generated in **Step 1** using the right-click deploy in **the API toolkit**.

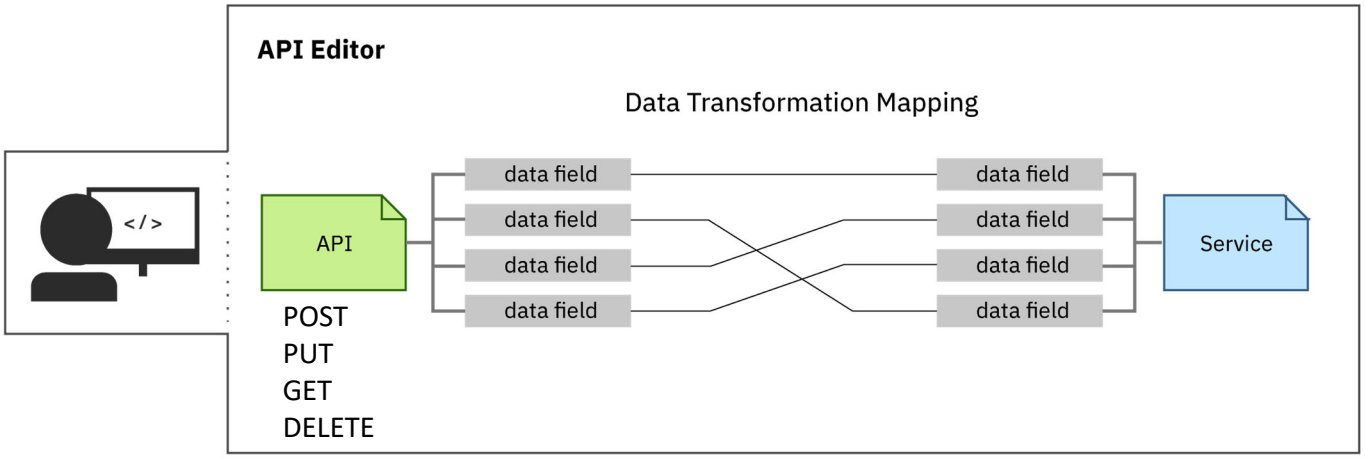
mitchj@us.ibm.com

© 2018, 2021 IBM Corporation

Steps to expose a z/OS application

z/OS Connect EE

Export the service and then import it to create an API that consumes the service



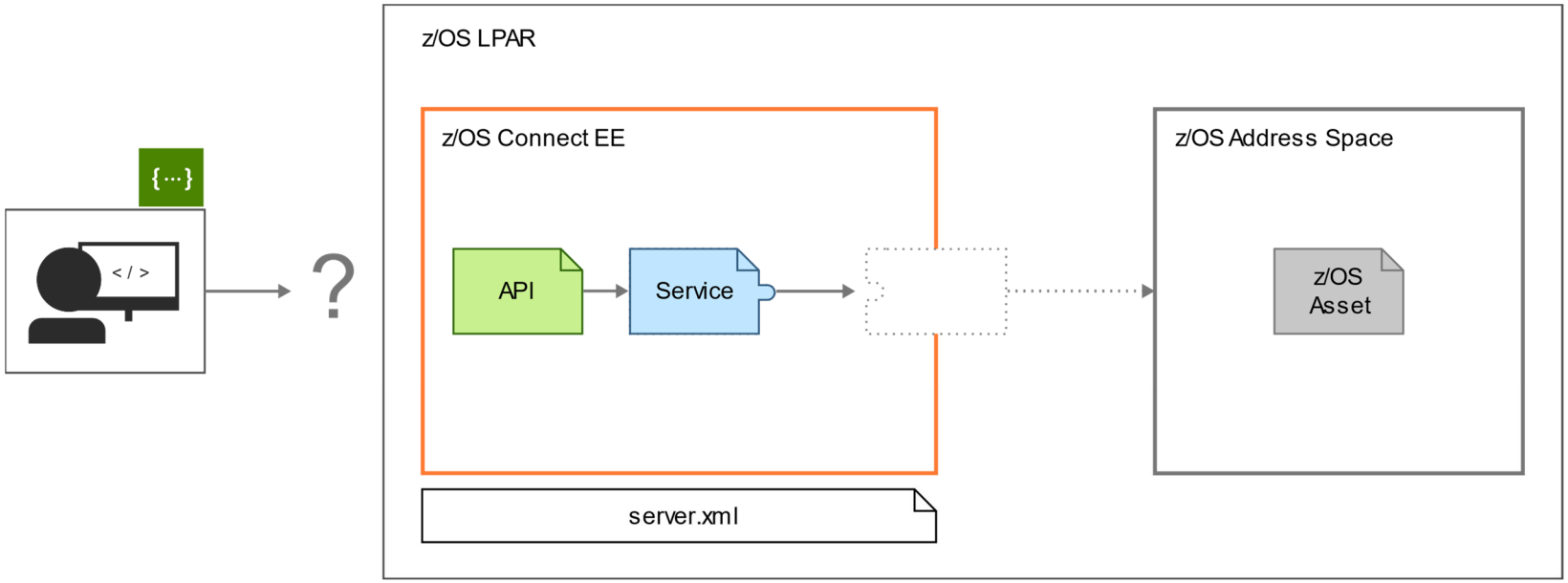
- Import the service archive file into the **API toolkit** and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

Deploy the API



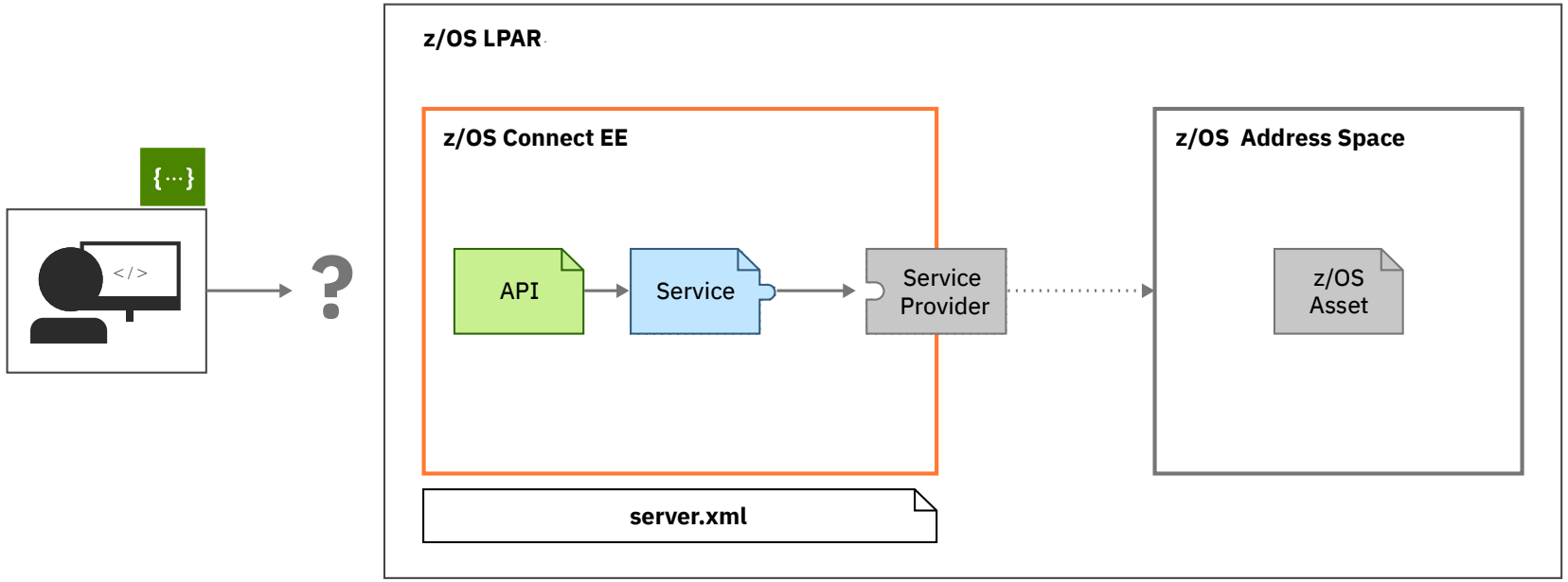
Deploy your API using the right-click deploy in **the API toolkit**

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

Configure the service provider



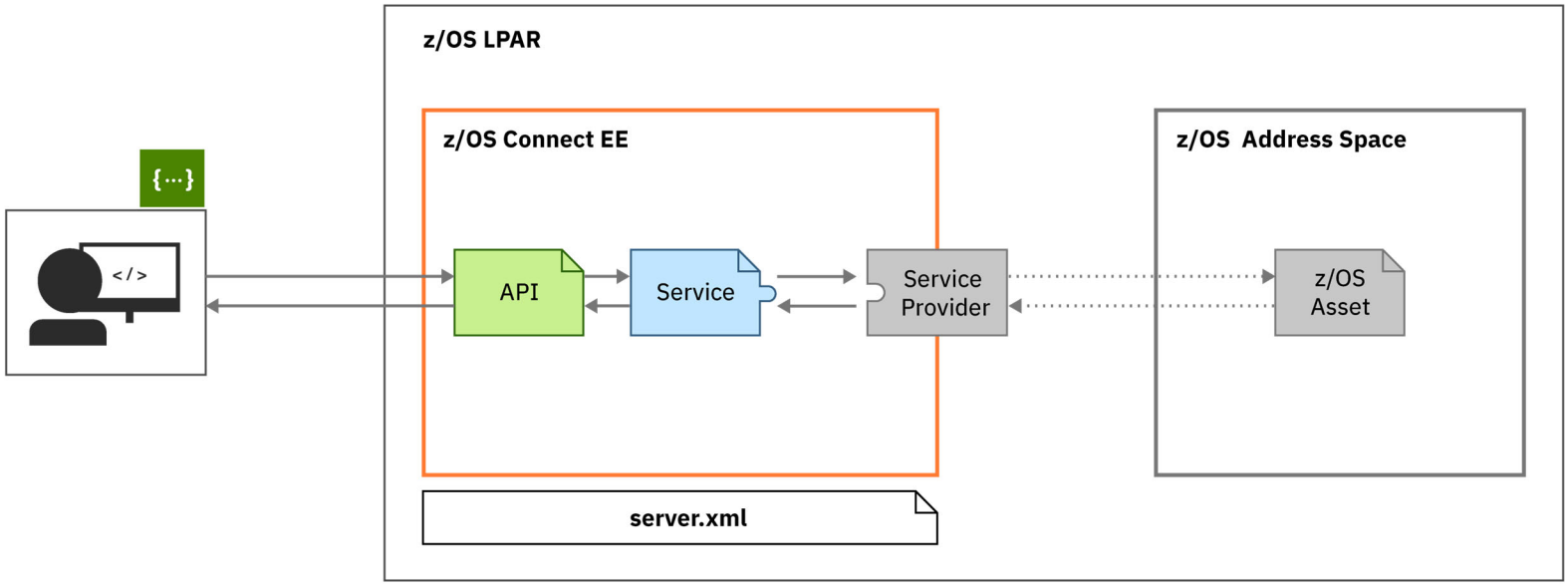
Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

Done

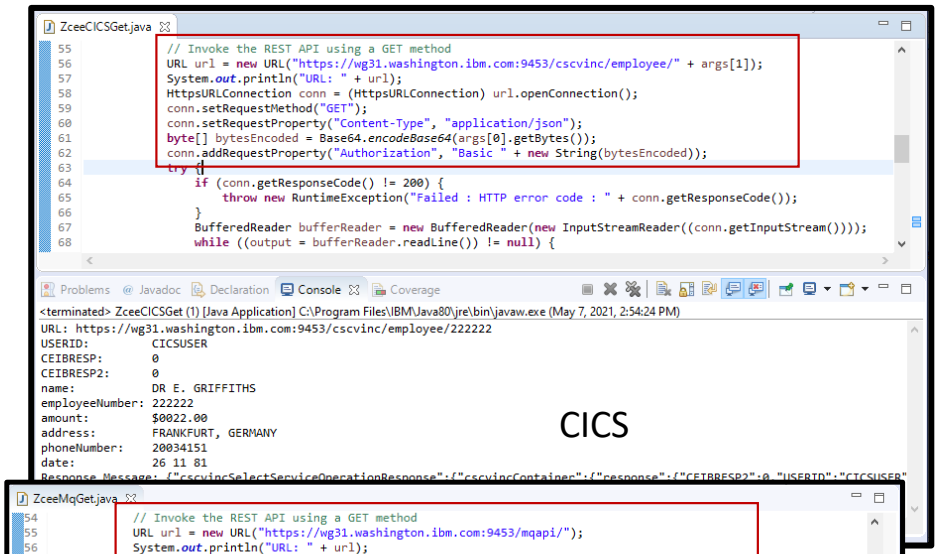


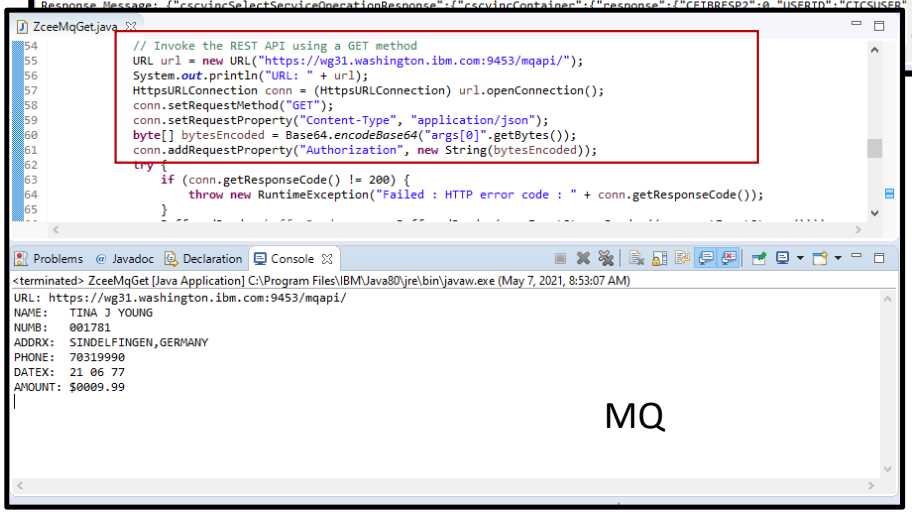
- The API is ready to be consumed and requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port, security)

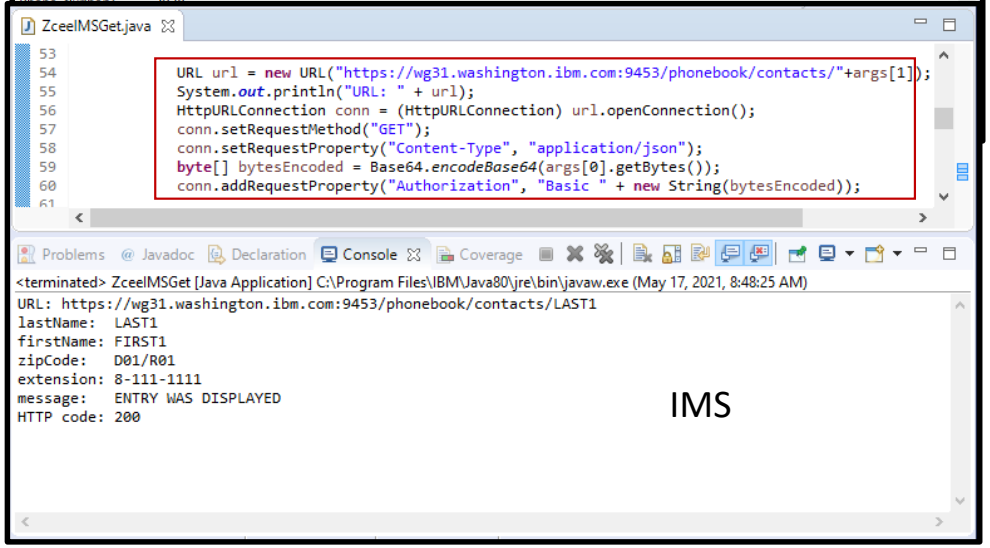
mitchj@us.ibm.com

Results: the client code is totally unaware of the z/OS infrastructure

z/OS Connect EE







mitchj@us.ibm.com

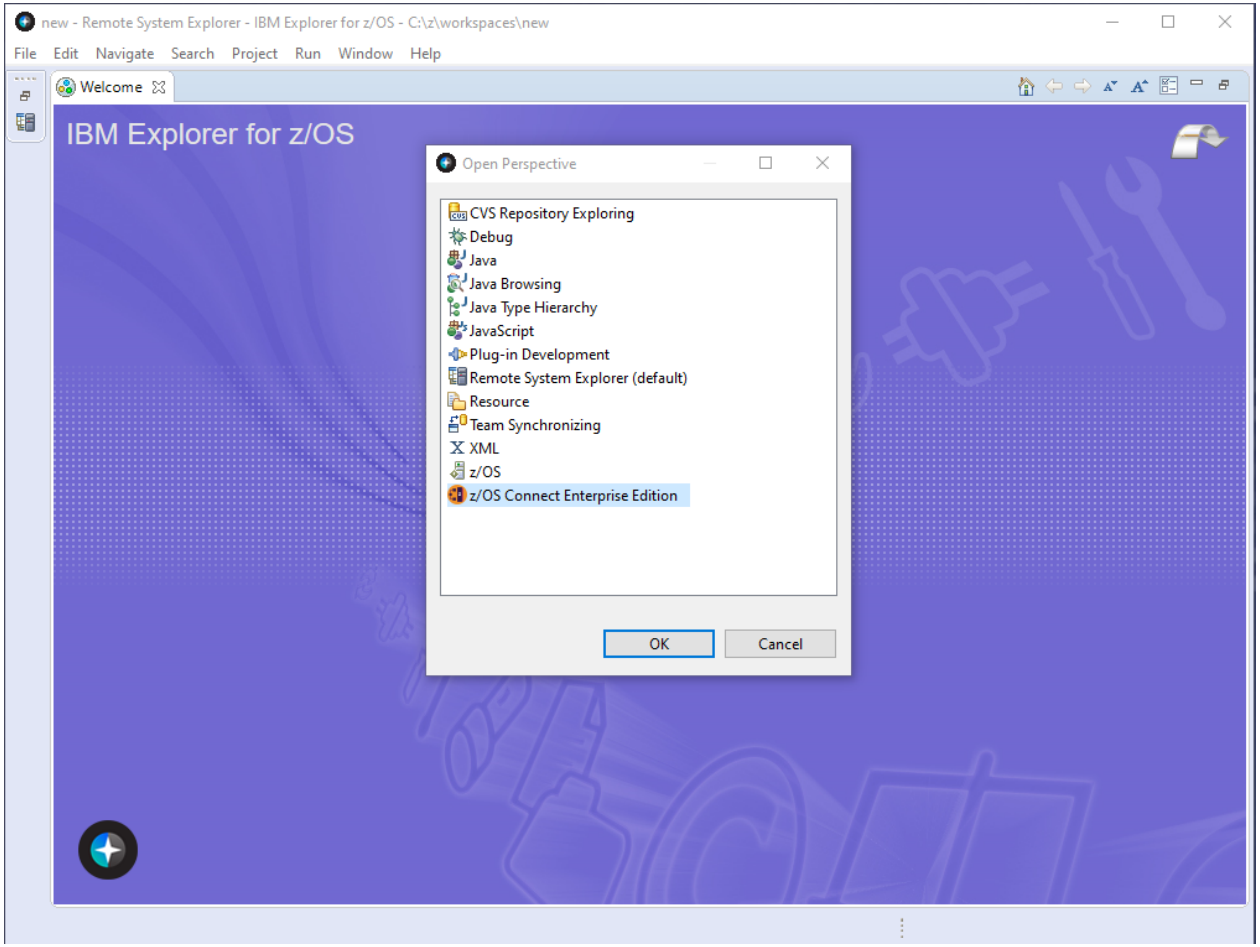
/api_toolkit/services

Simple **service creation**.

mitchj@us.ibm.com

Eclipse API toolkit – Creating Services for CICS, IMS TM, IMS DB, Db2 and MQ

z/OS Connect EE



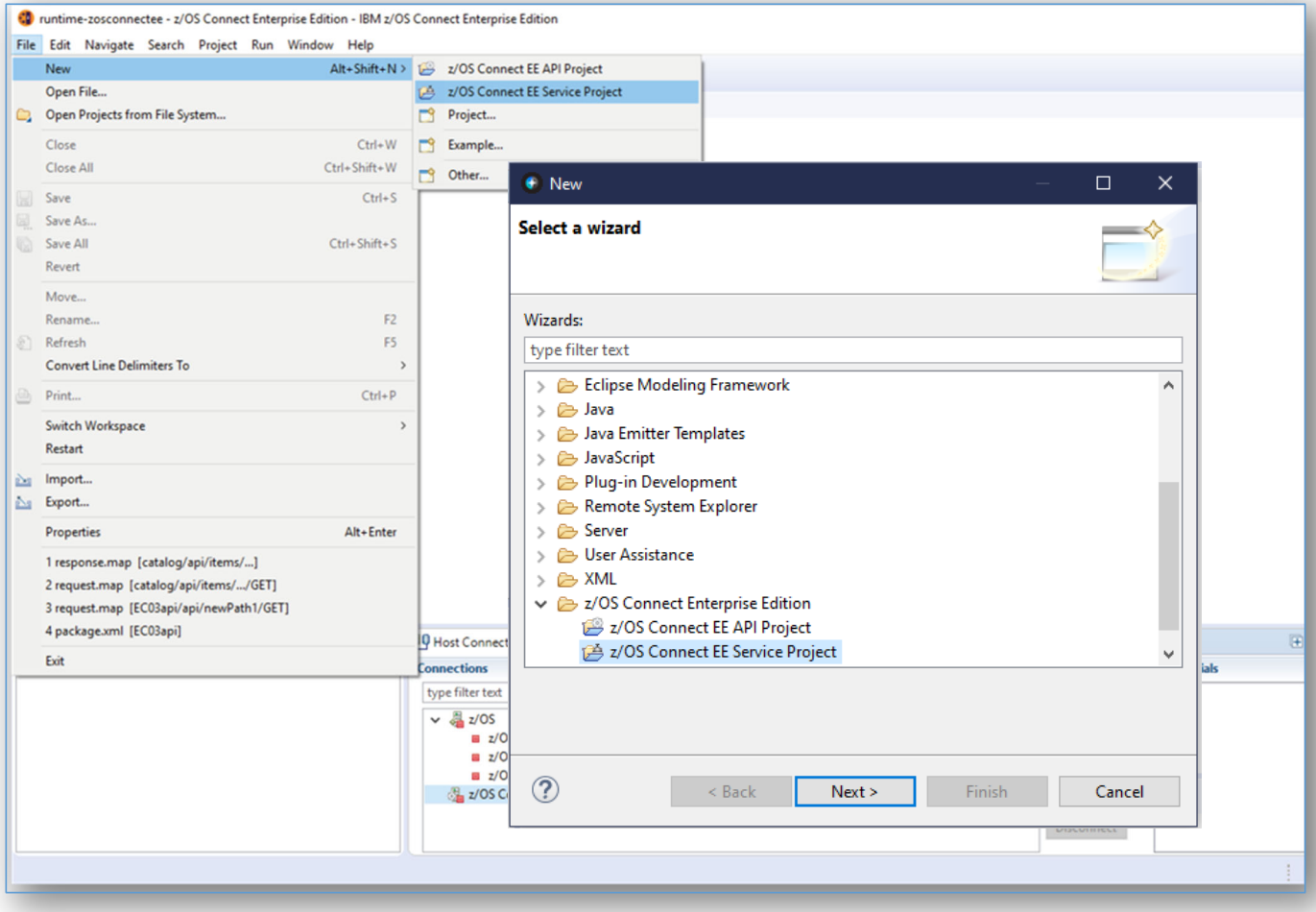
Use the **API toolkit** to create services
through Eclipse-based tooling.

The API toolkit is available in the z/OS Connect Enterprise Edition Perspective in an Eclipse environment.

mitchj@us.ibm.com

API toolkit – Creating Services for CICS, IMS TM, IMS DB, Db2 and MQ

z/OS Connect EE



Use the **API toolkit** to create services through Eclipse-based tooling.

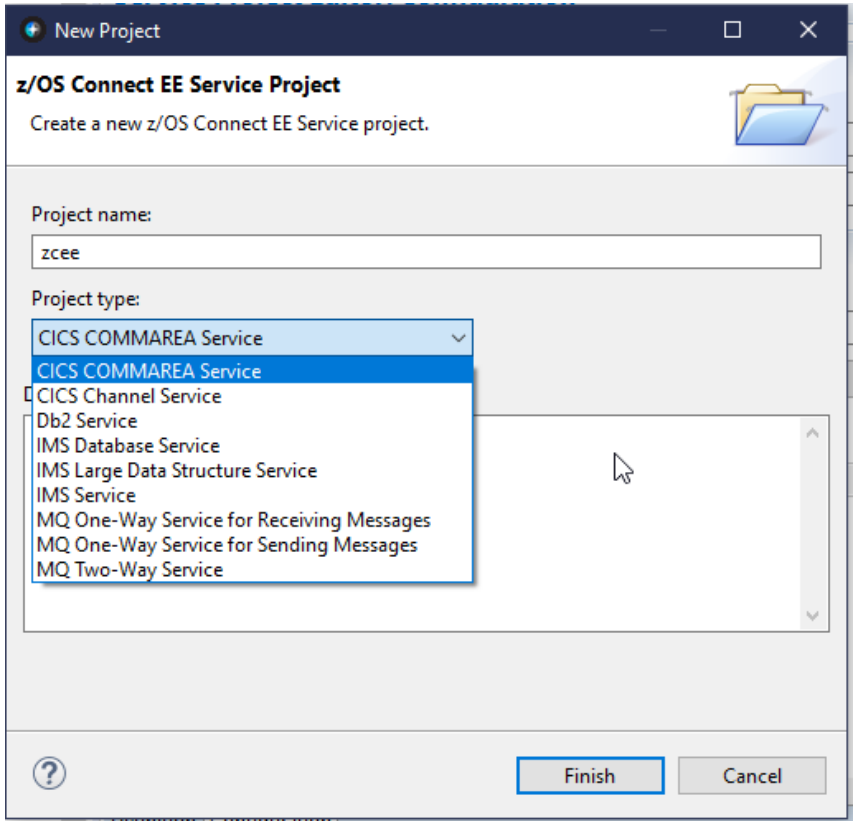
Services are described as Eclipse **Projects**, so they can be easily managed in source control.

mitchj@us.ibm.com

API toolkit – Creating Services for CICS, IMS TM, IMS DB, Db2 and MQ

z/OS Connect EE

Service creation – a common interface



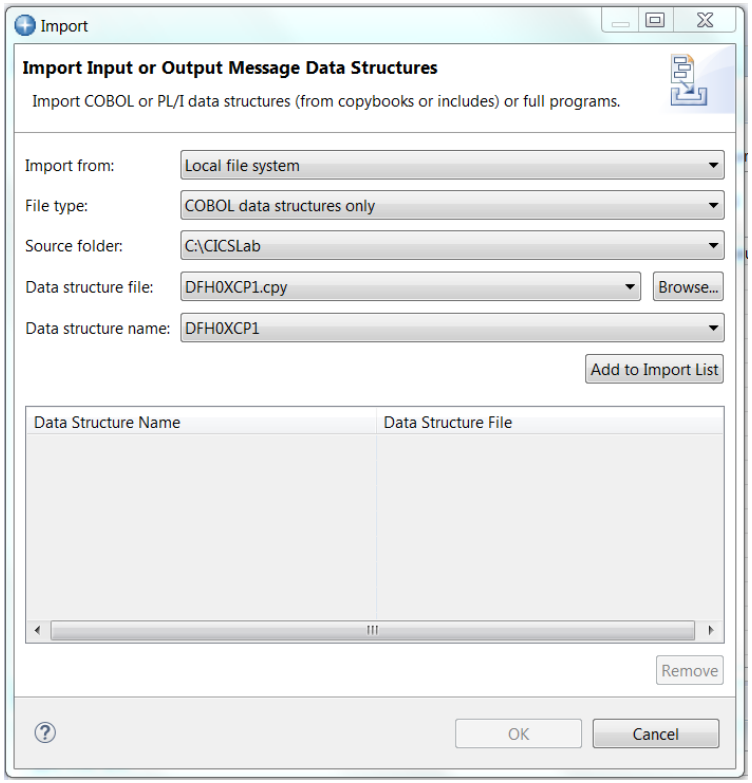
A common interface for service creation, irrespective of back-end subsystem.

mitchj@us.ibm.com

API toolkit – Creating Services for CICS, IMS TM and MQ

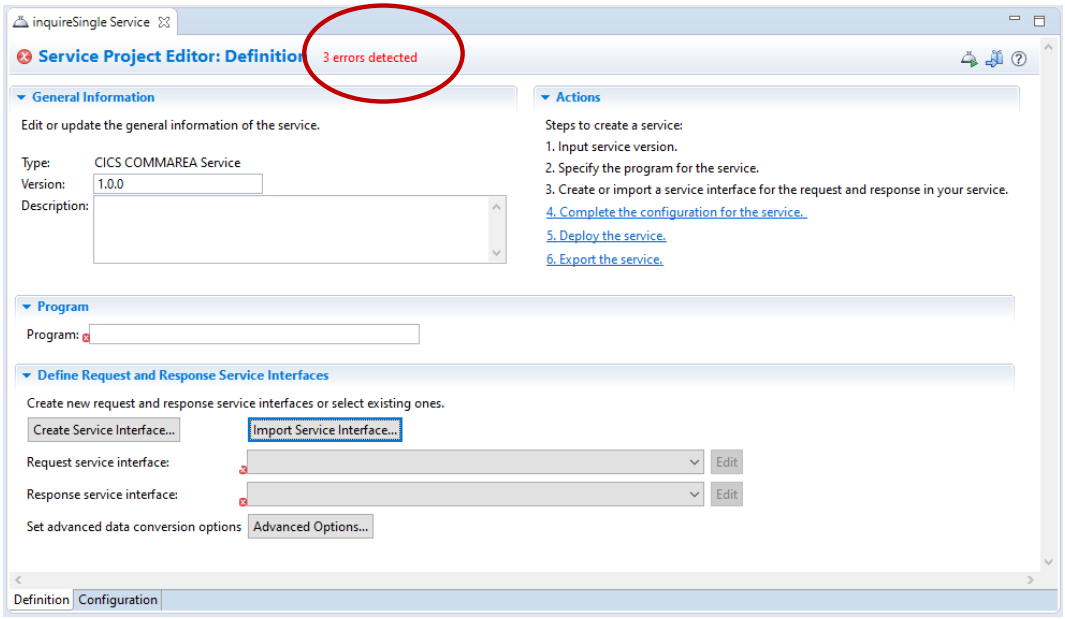
z/OS Connect EE

Creating a service project from source for a COMMAREA, Container or Message



Start by importing data structures into the service interface from the local file system or the workspace to create the request and response service interfaces.

The service interface supports complex data structures, including OCCURS DEPENDING ON and REDEFINES clauses.



mitchj@us.ibm.com

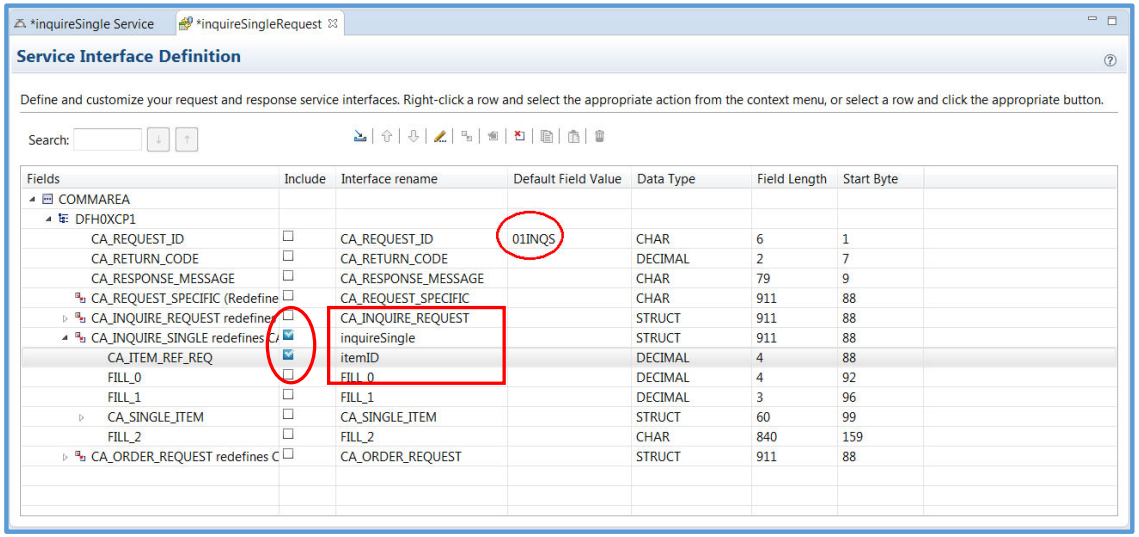
API toolkit – Creating Services for CICS, IMS TM and MQ

z/OS Connect EE

Allows editing a request service interface definition

```
*---------------------------------------------------------------*
* Check which operation in being requested
*---------------------------------------------------------------*
* Uppercase the value passed in the Request Id field
     MOVE FUNCTION UPPER-CASE(CA-REQUEST-ID) TO CA-REQUEST-ID
     EVALUATE CA-REQUEST-ID
         WHEN '01INQC'
*        Call routine to perform for inquire
             PERFORM CATALOG-INQUIRE
         WHEN '01INQS'
*        Call routine to perform for inquire for single item
             PERFORM CATALOG-INQUIRE-SINGLE
         WHEN '01ORDR'
*        Call routine to place order
             PERFORM PLACE-ORDER
         WHEN OTHER
*        Request is not recognised or supported
             PERFORM REQUEST-NOT-RECOGNISED
     END-EVALUATE
```

See the imported data structure and then can **redact fields**, **rename fields**, and **add default values to fields** to make the service more consumable for an API developer.

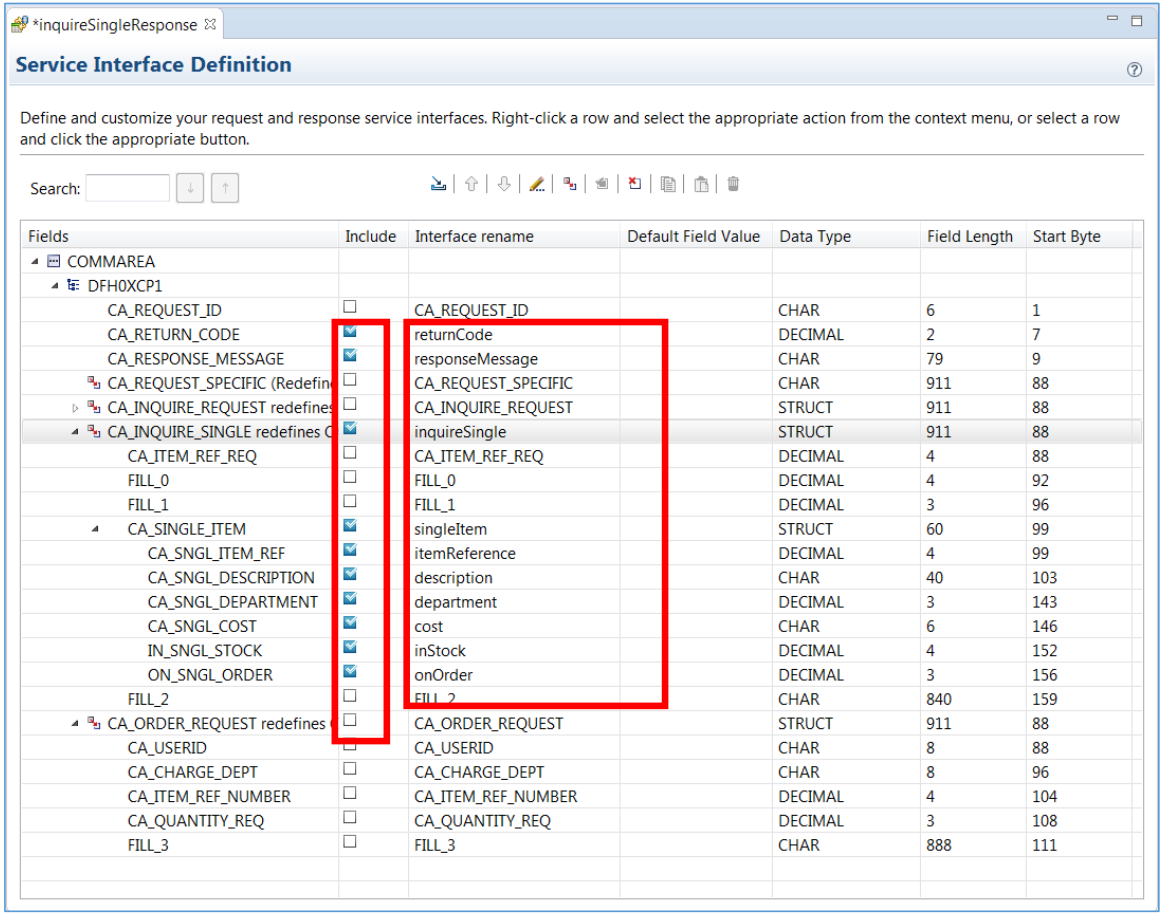


mitchj@us.ibm.com

API toolkit – Creating Services for CICS, IMS TM, IMS DB and MQ

And editing a response message service interface definition

z/OS Connect EE



Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID		CHAR	6	1
CA_RETURN_CODE	☑	returnCode		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☑	responseMessage		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefin	☐	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines	☐	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines C	☑	inquireSingle		STRUCT	911	88
CA_ITEM_REF_REQ	☐	CA_ITEM_REF_REQ		DECIMAL	4	88
FILL_0	☐	FILL_0		DECIMAL	4	92
FILL_1	☐	FILL_1		DECIMAL	3	96
CA_SINGLE_ITEM	☑	singleItem		STRUCT	60	99
CA_SNGL_ITEM_REF	☑	itemReference		DECIMAL	4	99
CA_SNGL_DESCRIPTION	☑	description		CHAR	40	103
CA_SNGL_DEPARTMENT	☑	department		DECIMAL	3	143
CA_SNGL_COST	☑	cost		CHAR	6	146
IN_SNGL_STOCK	☑	inStock		DECIMAL	4	152
ON_SNGL_ORDER	☑	onOrder		DECIMAL	3	156
FILL_2	☐	FILL_2		CHAR	840	159
CA_ORDER_REQUEST redefines	☐	CA_ORDER_REQUEST		STRUCT	911	88
CA_USERID	☐	CA_USERID		CHAR	8	88
CA_CHARGE_DEPT	☐	CA_CHARGE_DEPT		CHAR	8	96
CA_ITEM_REF_NUMBER	☐	CA_ITEM_REF_NUMBER		DECIMAL	4	104
CA_QUANTITY_REQ	☐	CA_QUANTITY_REQ		DECIMAL	3	108
FILL_3	☐	FILL_3		CHAR	888	111

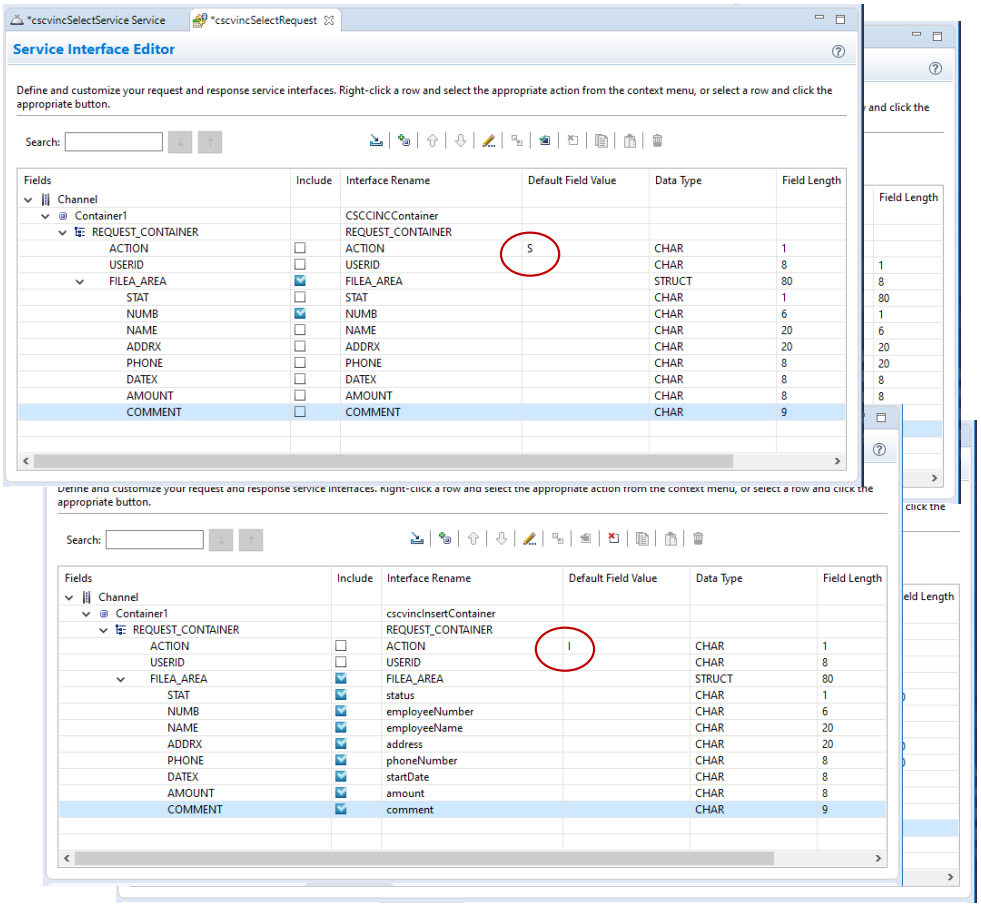
See the imported data structure and can **redact fields** and **rename fields**

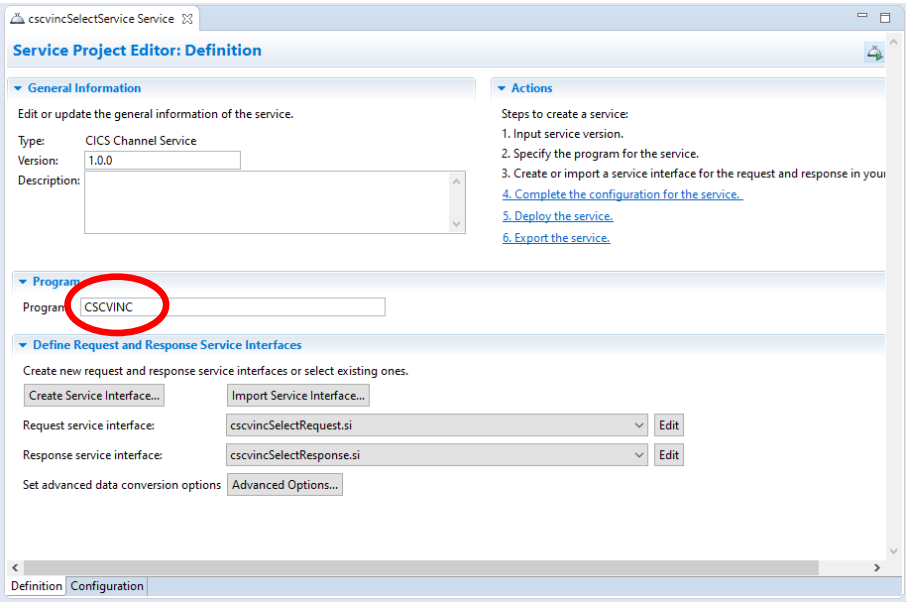
mitchj@us.ibm.com

API toolkit – Creating Services for CICS

Creating multiple services definitions to the same resource

z/OS Connect EE






```
EVALUATE ACTION of Request-Container
    WHEN 'D'
      PERFORM Delete-Record
    WHEN 'I'
      PERFORM Insert-Record
    WHEN 'U'
      PERFORM Update-Record
    WHEN 'S'
      PERFORM Select-Record
 END-EVALUATE.
```

The service developer creates distinct services for each function by setting the ACTION field to S for select, I for insert, U for update or D for delete

mitchj@us.ibm.com

API toolkit – Creating Services for IMS

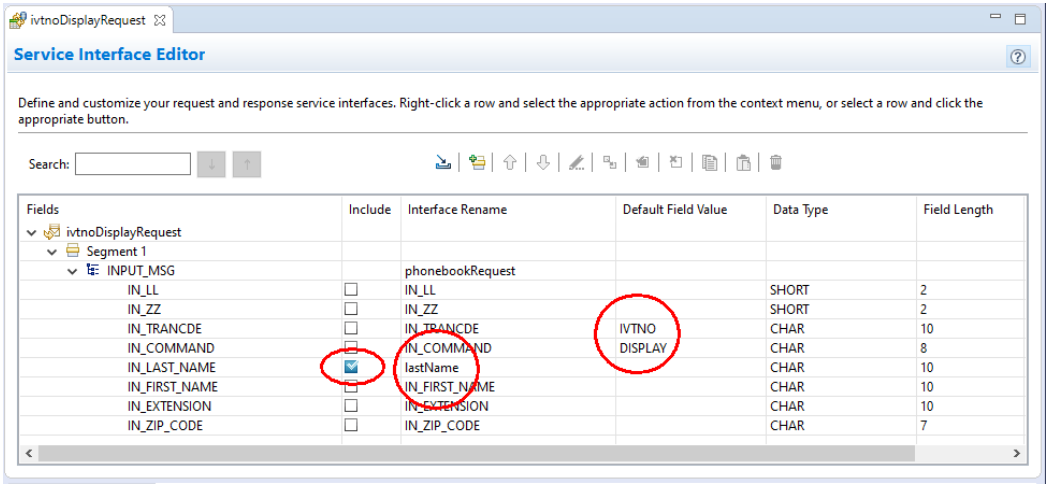
Creating a “GET” service interface request definition

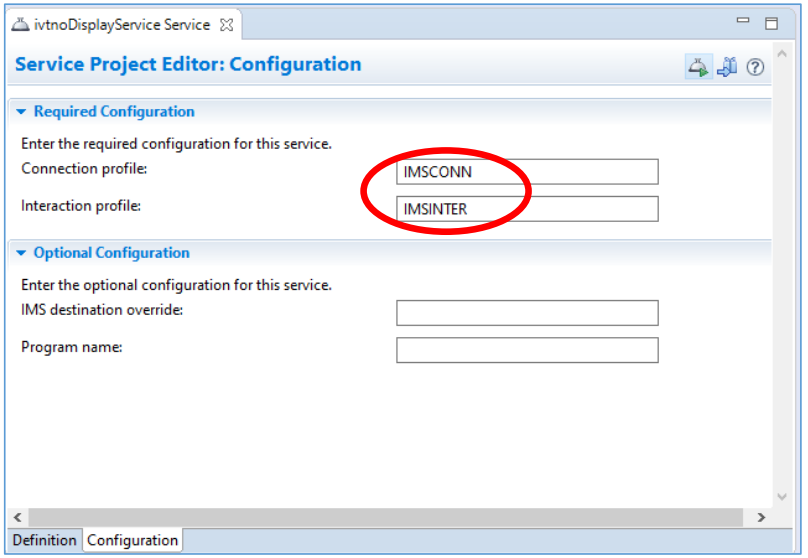
z/OS Connect EE

```
*--------------------------------------------------------------------*
*       ROUTE TO REQUEST HANDLER                                     *
*--------------------------------------------------------------------*
          SPACE 1
          CLC    KADD,IOCMD      IF COMMAND ADD ENTERED ?
          BE     TOADD           ...THEN, GOTO INSERT ENTRY
          CLC    KUPD,IOCMD      IF COMMAND UPDATE ENTERED ?
          BE     TOUPD           ...THEN, GOTO UPDATE ENTRY
          CLC    KDEL,IOCMD      IF COMMAND DEL ENTERED ?
          BE     TODEL           ...THEN, GOTO DELETE ENTRY
          CLC    KDIS,IOCMD      IF COMMAND DIS ENTERED ?
          BE     TODIS           ...THEN, GOTO DISPLAY ENTRY
          CLC    KTAD,IOCMD      IF TEST ADD WITH REPLY ?
          BE     TOTAD           ...THEN,
          B      INVREQ1         INVALID REQUEST
```

The service developer creates distinct services for each function.

DISPLAY (GET)
DELETE (DELETE)
ADD (POST)
UPDATE (PUT)





IMS/TM Meta Data

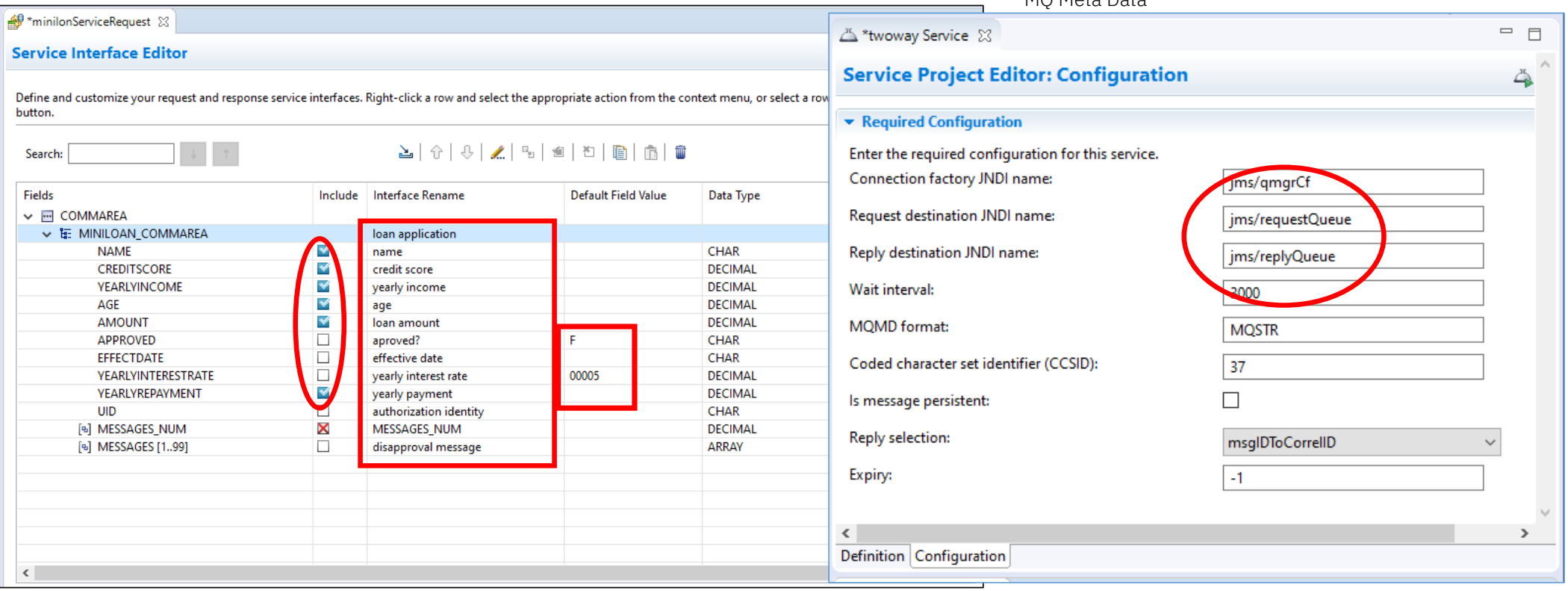
mitchj@us.ibm.com

API toolkit – Creating Services for MQ

z/OS Connect EE

Creating a “POST” service interface definition

MQ Meta Data



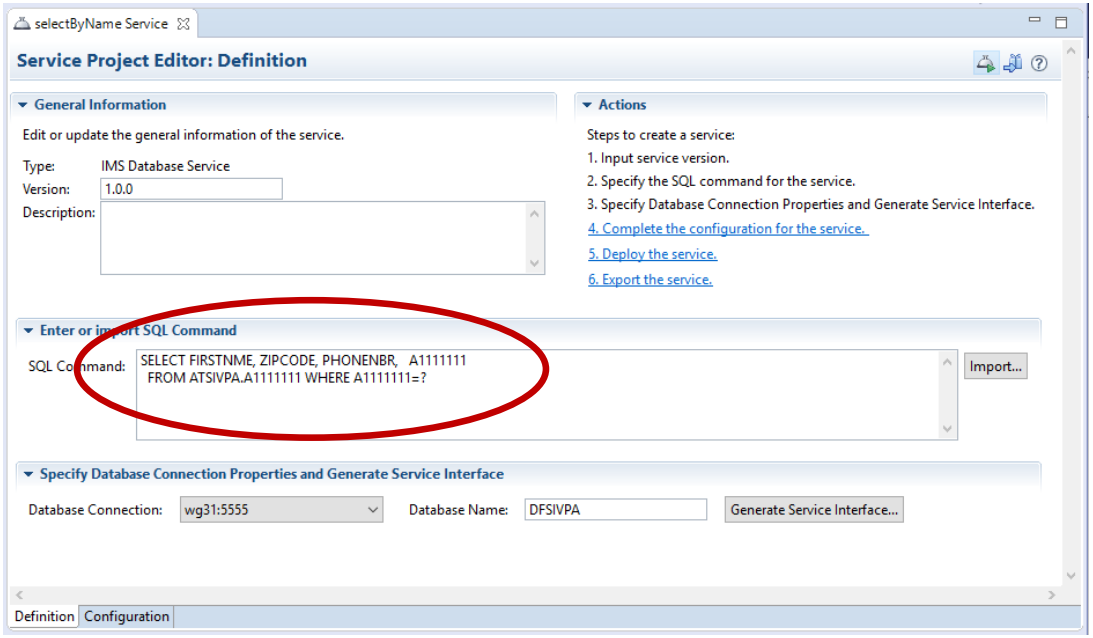
Again the service developer can then see the imported data structure and can **redact fields**, **rename fields**, and **add default values to fields** to make the service more consumable for an API developer.

mitchj@us.ibm.com

API toolkit – Creating Services for IMS DB

z/OS Connect EE

Creating a service project from the IMS Catalog



Use the IMS Catalog to assist with developing and testing SQL SELECT commands used for accessing IMS databases.

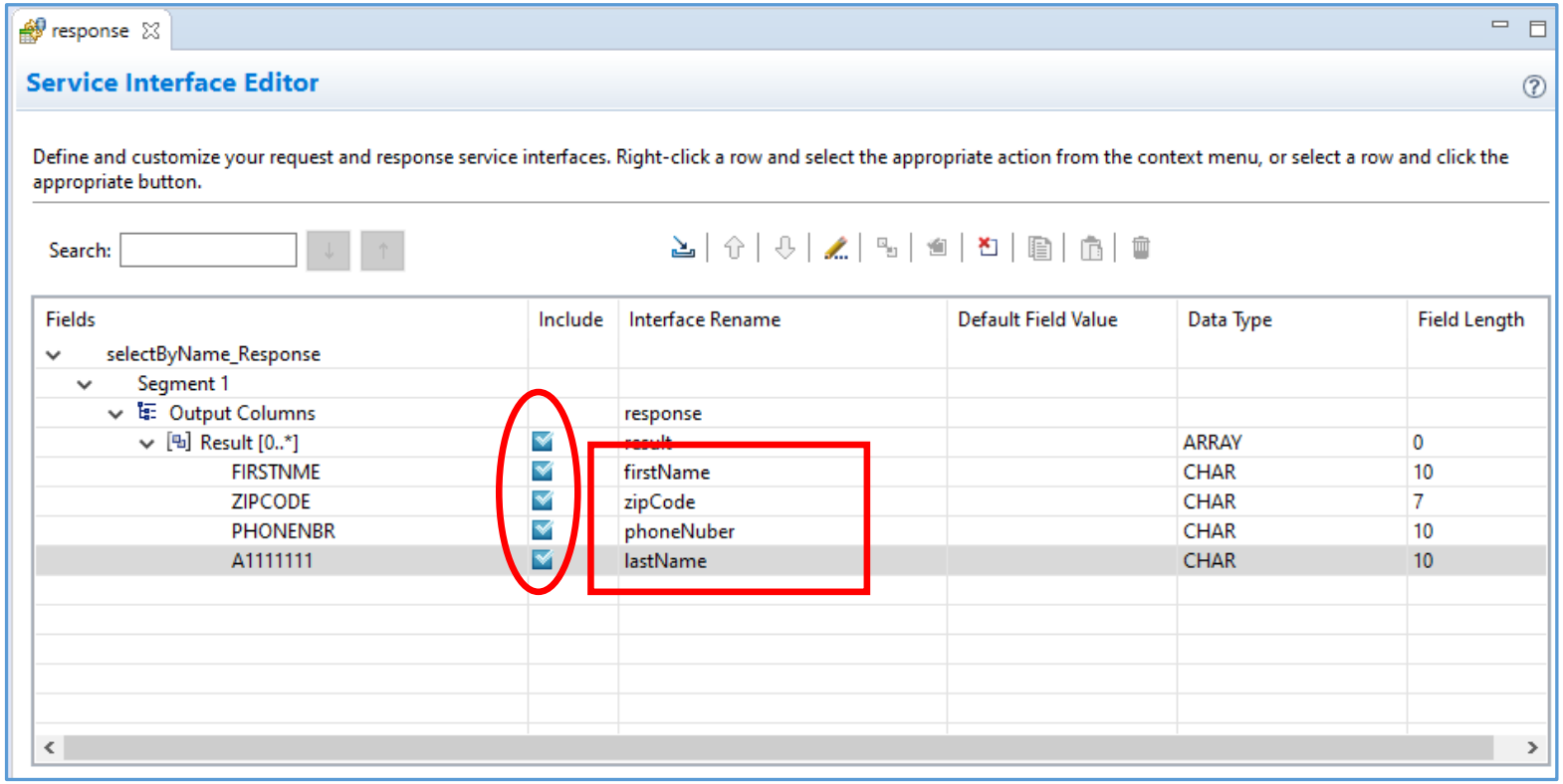
```
*-------------------------------------------------------------------*
*  SEGMENT DESCRIPTION                                              *
*    ROOT ONLY DATABASE                                             *
*      BYTES  1-10  LAST NAME (CHARACTER) - KEY                     *
*      BYTES  11-20 FIRST NAME (CHARACTER)                          *
*      BYTES  21-30 INTERNAL PHONE NUMBER (NUMERIC)                 *
*      BYTES  31-37 INTERNAL ZIP (CHARACTER)                        *
*      BYTES  38-40 RESERVED                                        *
*                                                                   *
*-------------------------------------------------------------------*
         DBD    NAME=IVPDB1,ACCESS=(HIDAM,OSAM)
         DATASET DD1=DFSIVD1,DEVICE=3380,SIZE=2048
         SEGM   NAME=A1111111,PARENT=0,BYTES=40,RULES=(LLV,LAST),      x
               PTR=(TB,CTR)
         FIELD  NAME=(A1111111,SEQ,U),BYTES=010,START=00001,TYPE=C
         FIELD  NAME=FIRSTNME,BYTES=010,START=00011,TYPE=C
         FIELD  NAME=PHONENBR,BYTES=010,START=00021,TYPE=C
         FIELD  NAME=ZIPCODE,BYTES=7,START=00031,TYPE=C
         LCHILD NAME=(A1,IVPDB1I),POINTER=INDX,RULES=LAST
         DBDGEN
         FINISH
         END
```

mitchj@us.ibm.com

API toolkit – Creating Services for IMS DB

z/OS Connect EE

The Toolkit allows editing a service interface definitions*

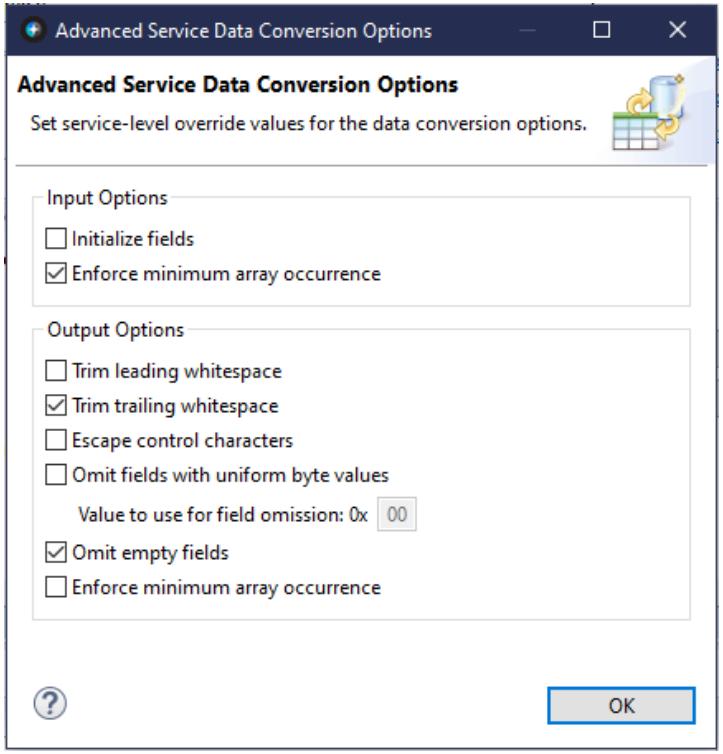


mitchj@us.ibm.com

*Using a slightly different process

API toolkit – Advanced Data Conversion Options

z/OS Connect EE



Request Messages:

- Initialize fields
- Enforce minimum array occurrence

Response Messages:

- Trim leading whitespace
- Trim trailing whitespace
- Escape control characters
- Omit fields with uniform byte values
- Omit empty fields
- Enforce minimum array occurrence

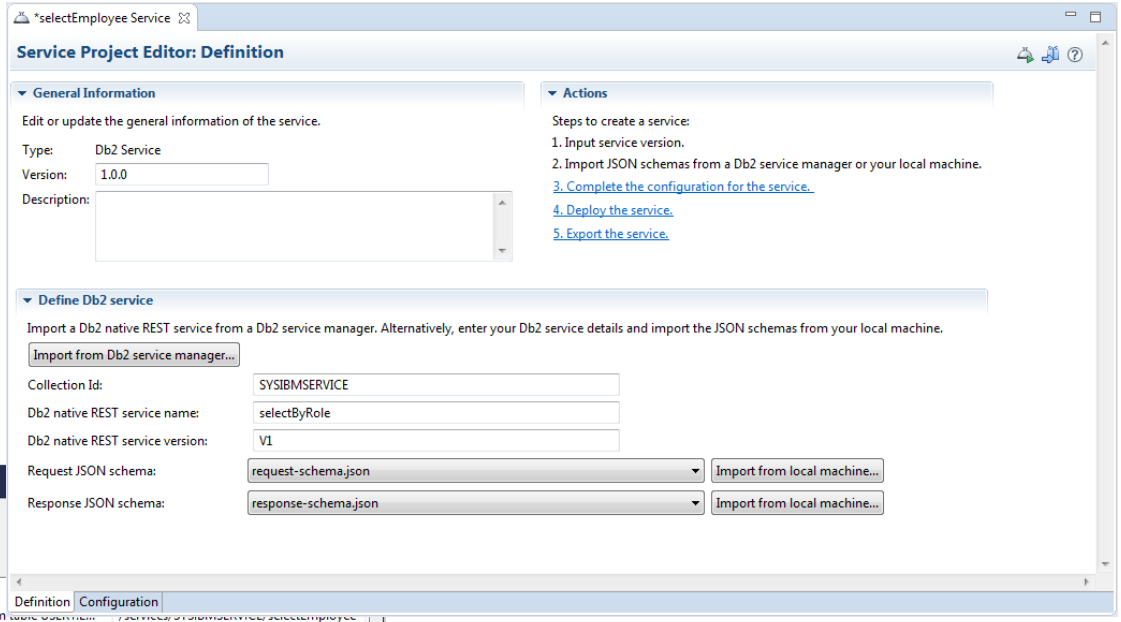
mitchj@us.ibm.com

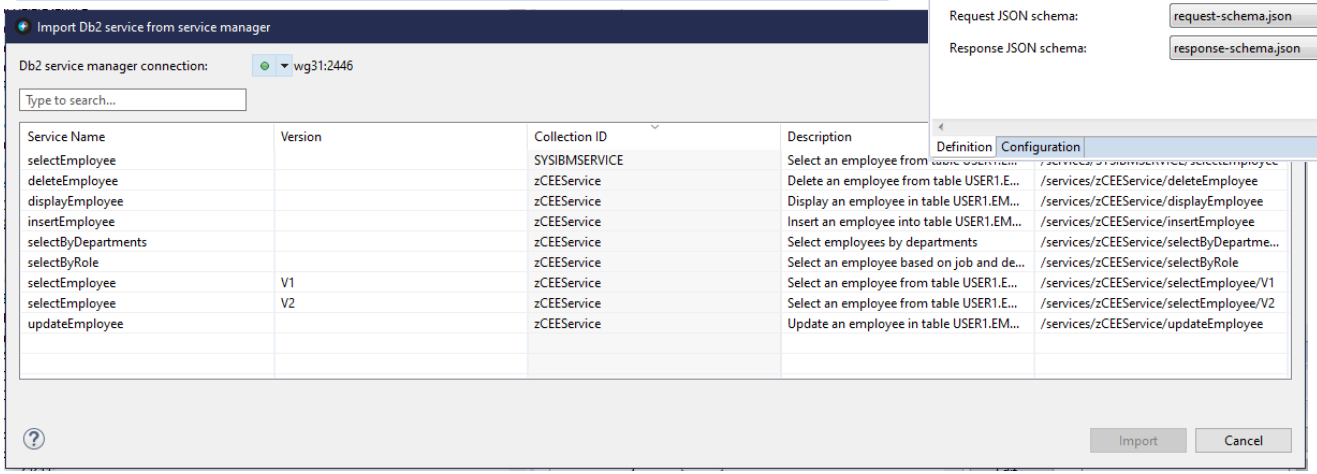
API toolkit – Creating Services for Db2

Creating a service project from Db2 REST service

z/OS Connect EE

```
//BIND EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD DSN=DSN1210.DB2.SDSNEXIT,DISP=SHR
//         DD DSN=DSN1210.DB2.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DSNSTMT  DD *
  SELECT EMPNO AS "employeeNumber", FIRSTNME AS "firstName",
        MIDINIT AS "middleInitial", LASTNAME as "lastName",
        WORKDEPT AS "department", PHONENO AS "phoneNumber",
        JOB AS "job"
  FROM USER1.EMPLOYEE WHERE EMPNO = :employeeNumber
//SYSTSIN  DD *
DSN SYSTEM(DSN2)
BIND SERVICE(SYSIBMSERVICE) -
NAME("selectEmployee") -
SQLENCODING(1047) -
DESCRIPTION('Select an employee from table USER1.EMPLOYEE')
```





The service developer retrieves details about the Db2 REST services

Note there is no service interface editor available

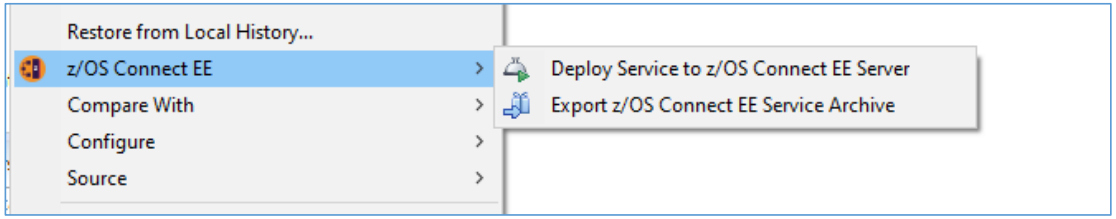
mitchj@us.ibm.com

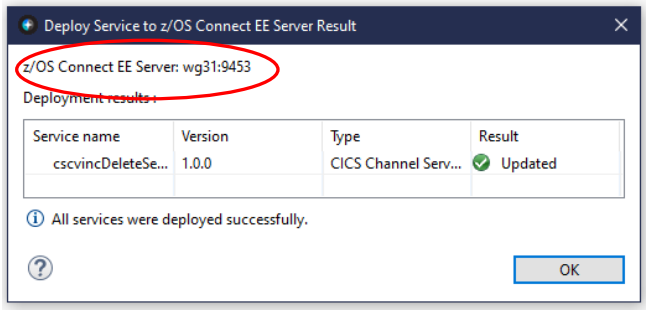
API toolkit – Services Editor

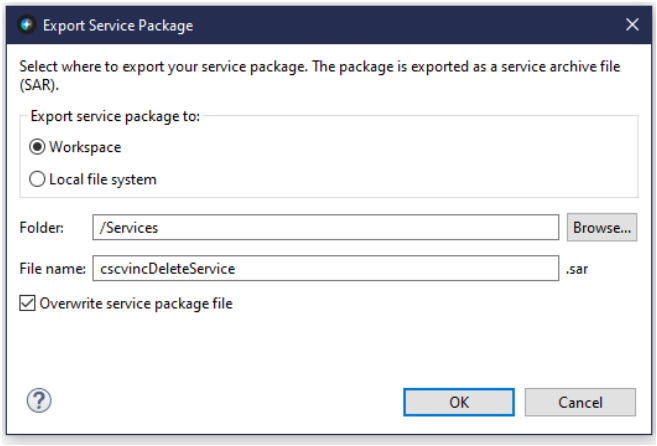
Server connection and Services deployment

z/OS Connect EE

Manage z/OS Connect EE server connections in the **Host Connections** view:





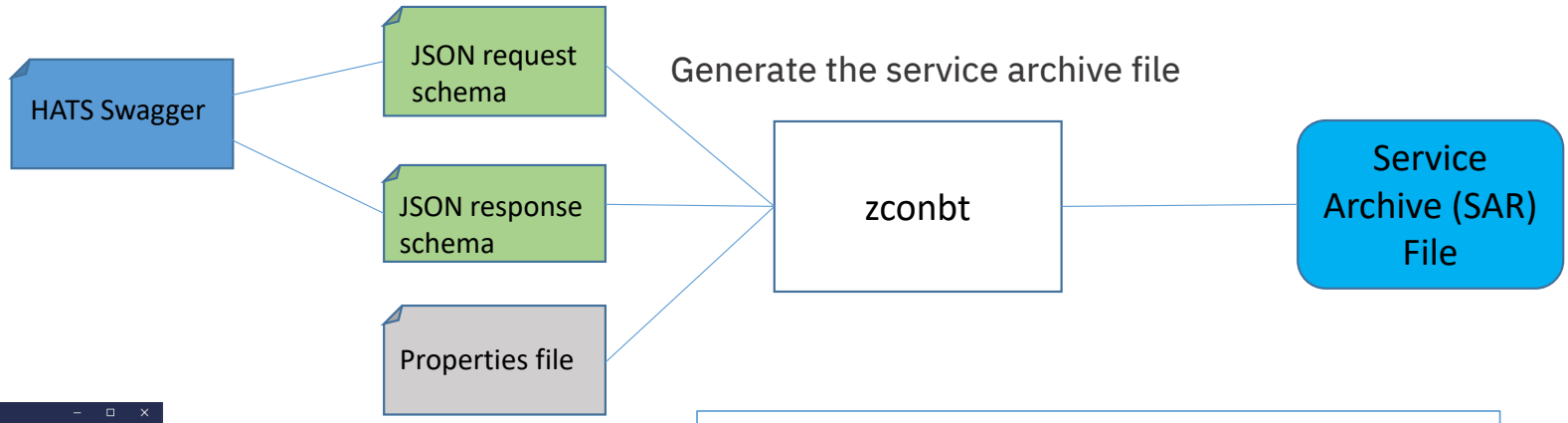


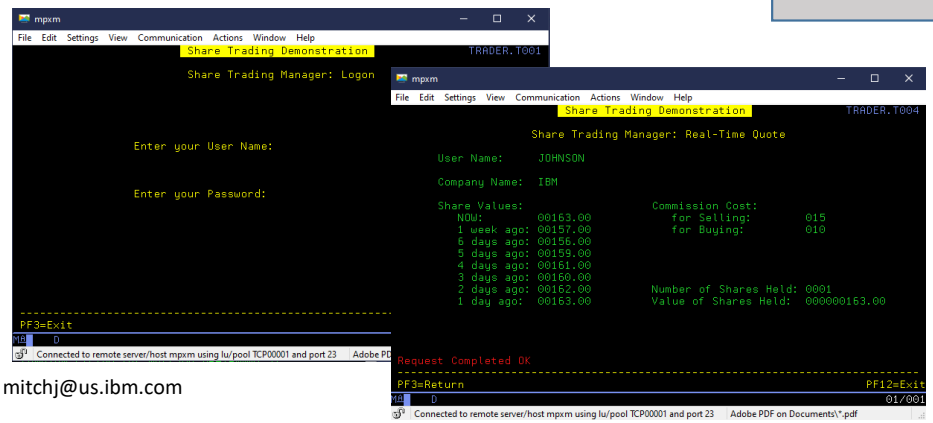
mitchj@us.ibm.com

Command line(zconbt) – REST Services

z/OS Connect EE

For HATS REST Services use the z/OS Connect Build toolkit (zconbt)






```
provider=rest
name=getCompany
version=1.0
description=Obtain a list of companies
requestSchemaFile=getCompanyRequest.json
responseSchemaFile=getCompanyResponse.json
verb=POST
uri=/Trader/rest/GetCompany
connectionRef=HatsConn
```

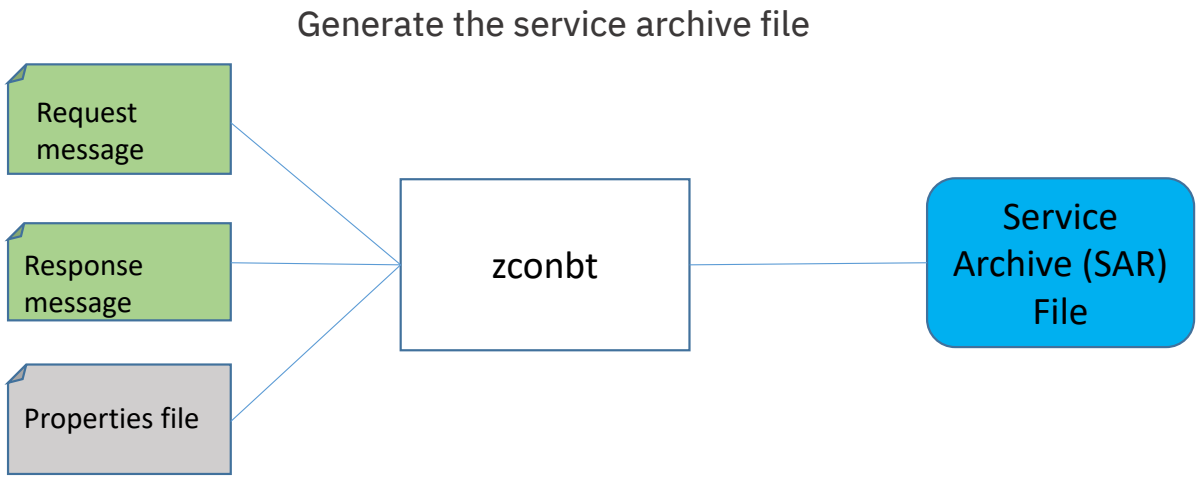
mitchj@us.ibm.com

Command line(zconbt) – MVS Batch

z/OS Connect EE

For batch WOLA services use the z/OS Connect Build toolkit (zconbt)

```
name=Filea
version=1.0
provider=wola
description=COBOL batch program
language=COBOL
program=ATSFILEA
register=FILEAZCON
connectionRef=wolaCF
requestStructure=fileareq.cpy
responseStructure=filearsp.cpy
```



WebSphere **O**ptimized **L**ocal **A**dapter – a protocol for cross memory communications between address spaces

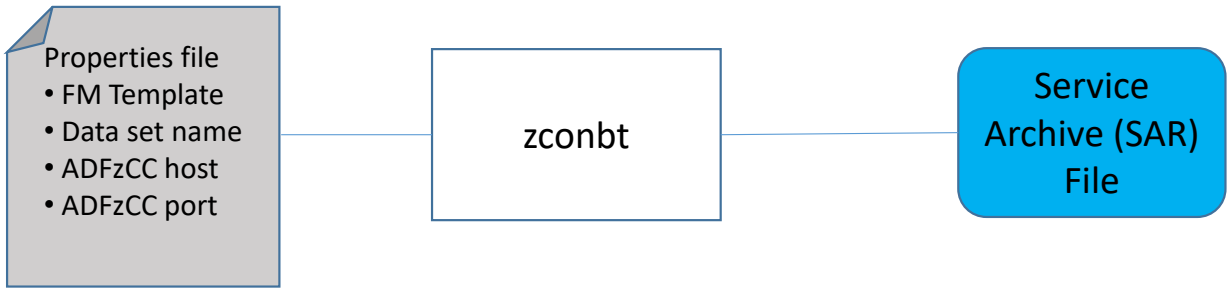
mitchj@us.ibm.com

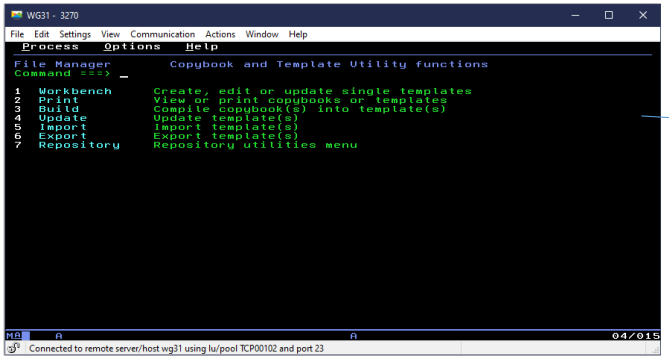
Command line(zconbt) – File Manager

z/OS Connect EE

For File Manager Services use the z/OS Connect Build toolkit (zconbt)

Generate the service archive file





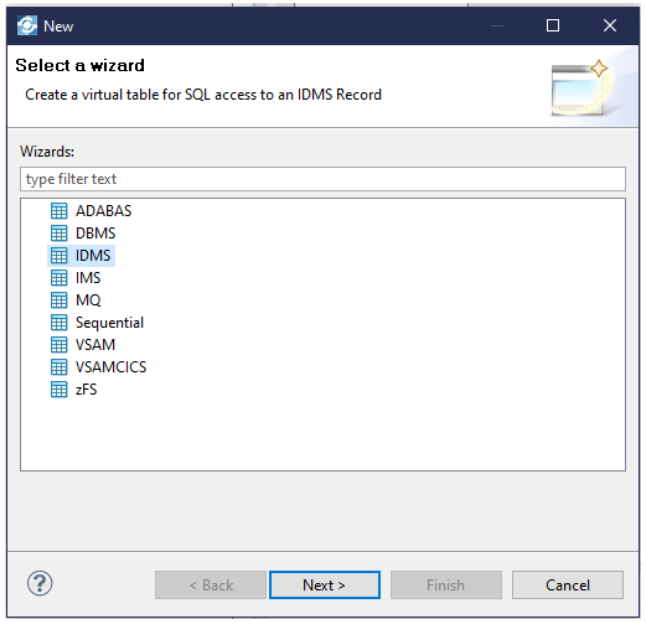

```
name=filea
provider=filemanager
host=wg31.washington.ibm.com
version=1.0
port=2800
file=USER1.ZCEE.FILEA
template=USER1.TEMPLATE(FILEA)
connid=default
userid=USER1
passwd=USER1
```

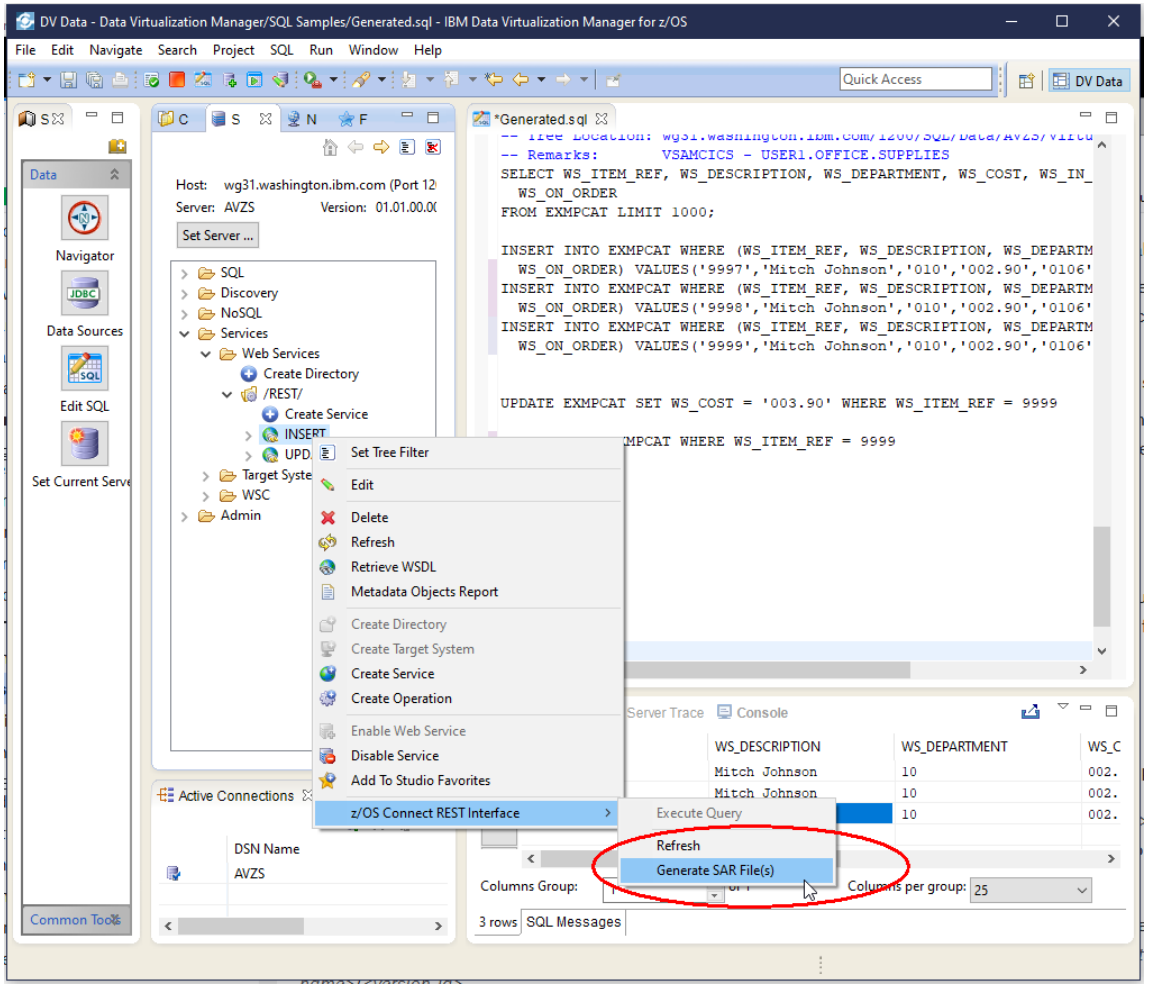
mitchj@us.ibm.com

DVM Studio

z/OS Connect EE

For DVM use the DVM Studio





mitchj@us.ibm.com

Once we have a Service Archive (SAR) What's next?

Quick and easy **API mapping.**

Remember: All service archives files are functionally equivalent regardless of how there are created

mitchj@us.ibm.com

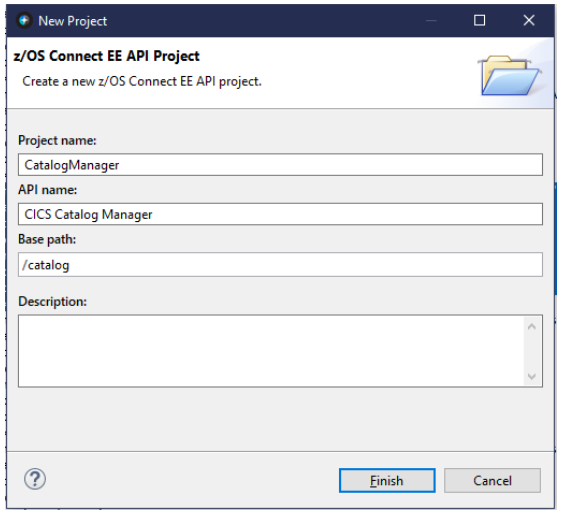
/api_toolkit/api_editor

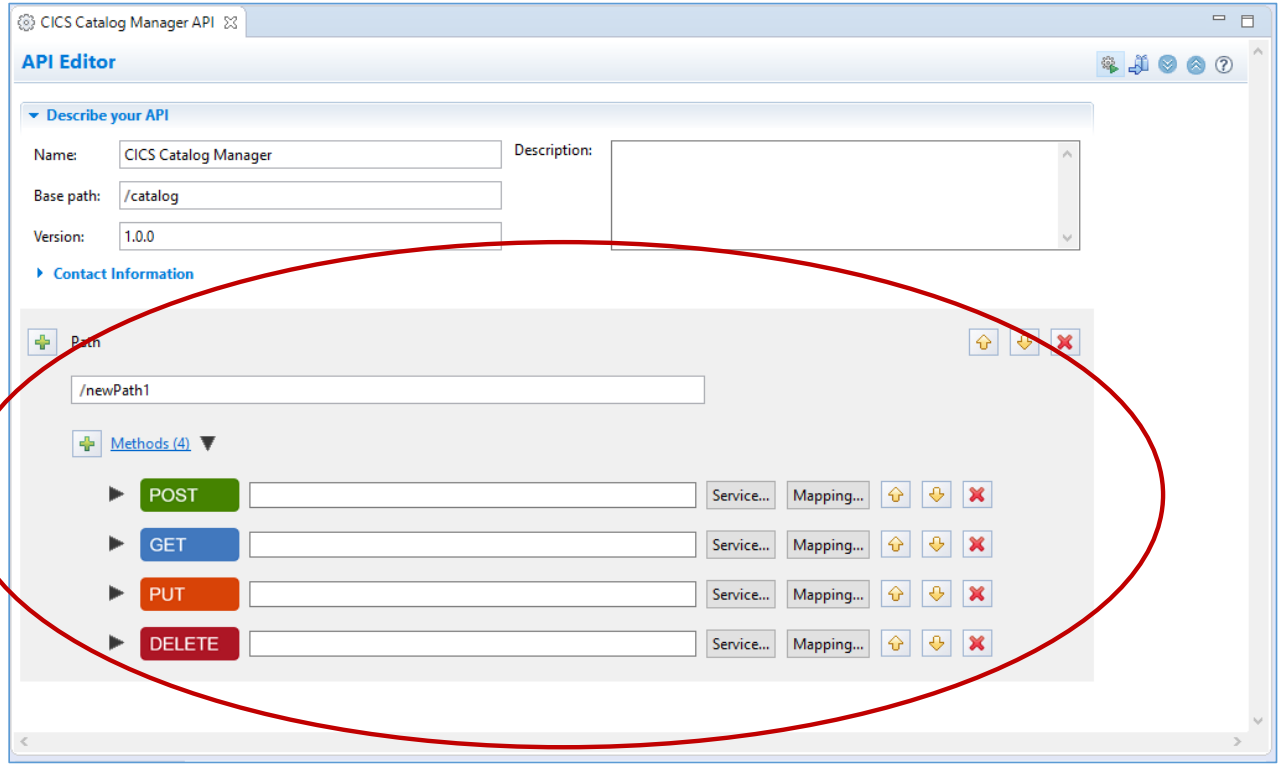
Quick and easy **API mapping.**

mitchj@us.ibm.com

API toolkit – API Editor

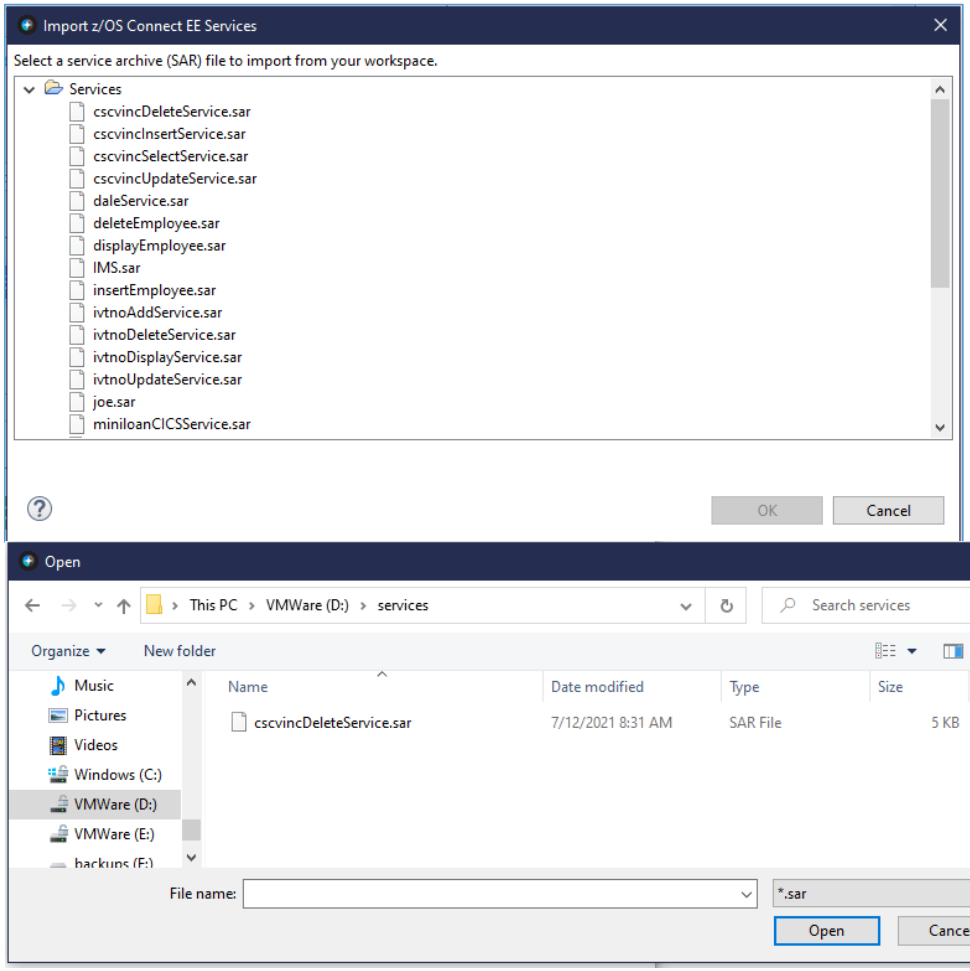
z/OS Connect EE

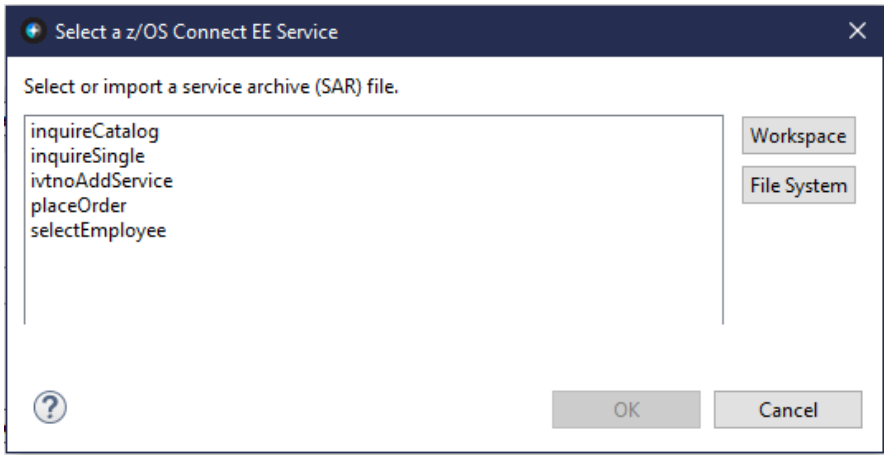




mitchj@us.ibm.com

Importing the service archives files

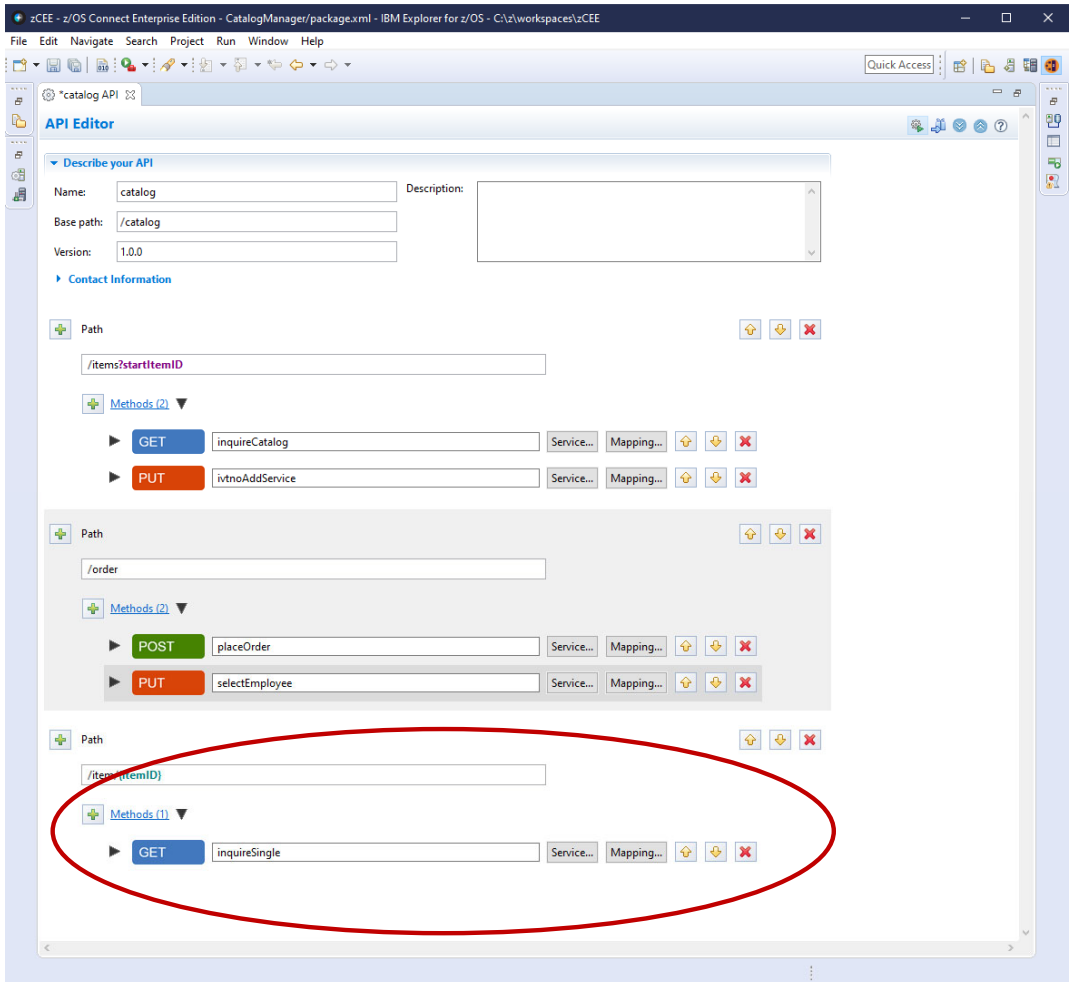




mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE



The **API toolkit** is designed to encourage RESTful API design.

Once you define your API, you can map backend services to each request.

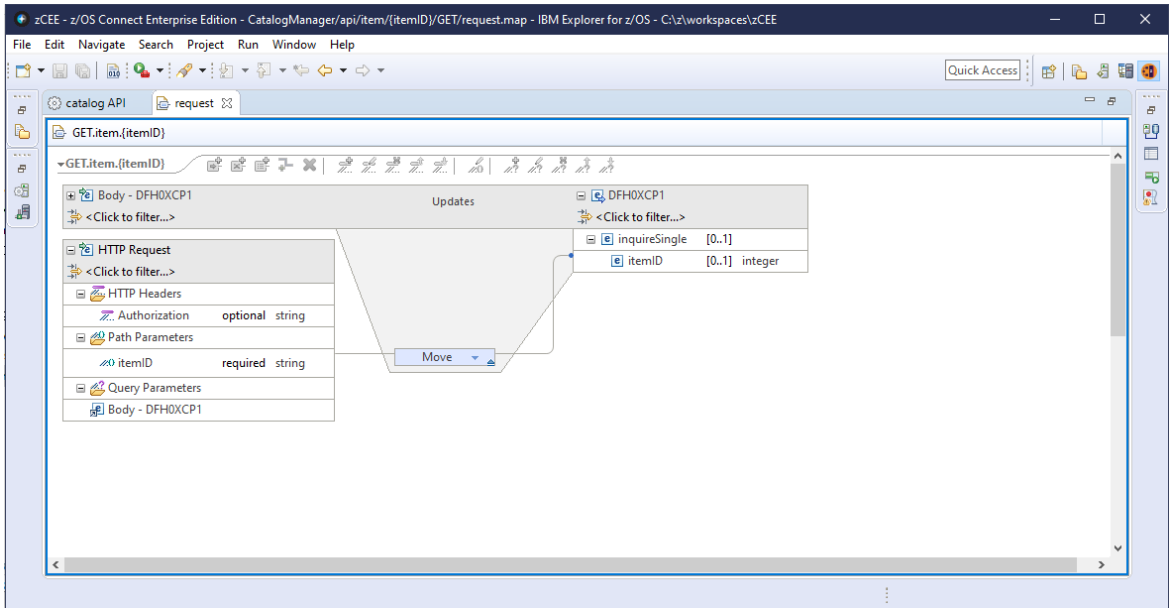
Your services are represented by a `.sar` files, which you import into the **API toolkit**, regardless of how the service archive file was generated.

mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

API mapping: Assign values to the interface fields exposed by the service developer



Map both the request and response for each API.

Map path and query parameters to native data structures.

Assign static values to fields, useful for Op codes.

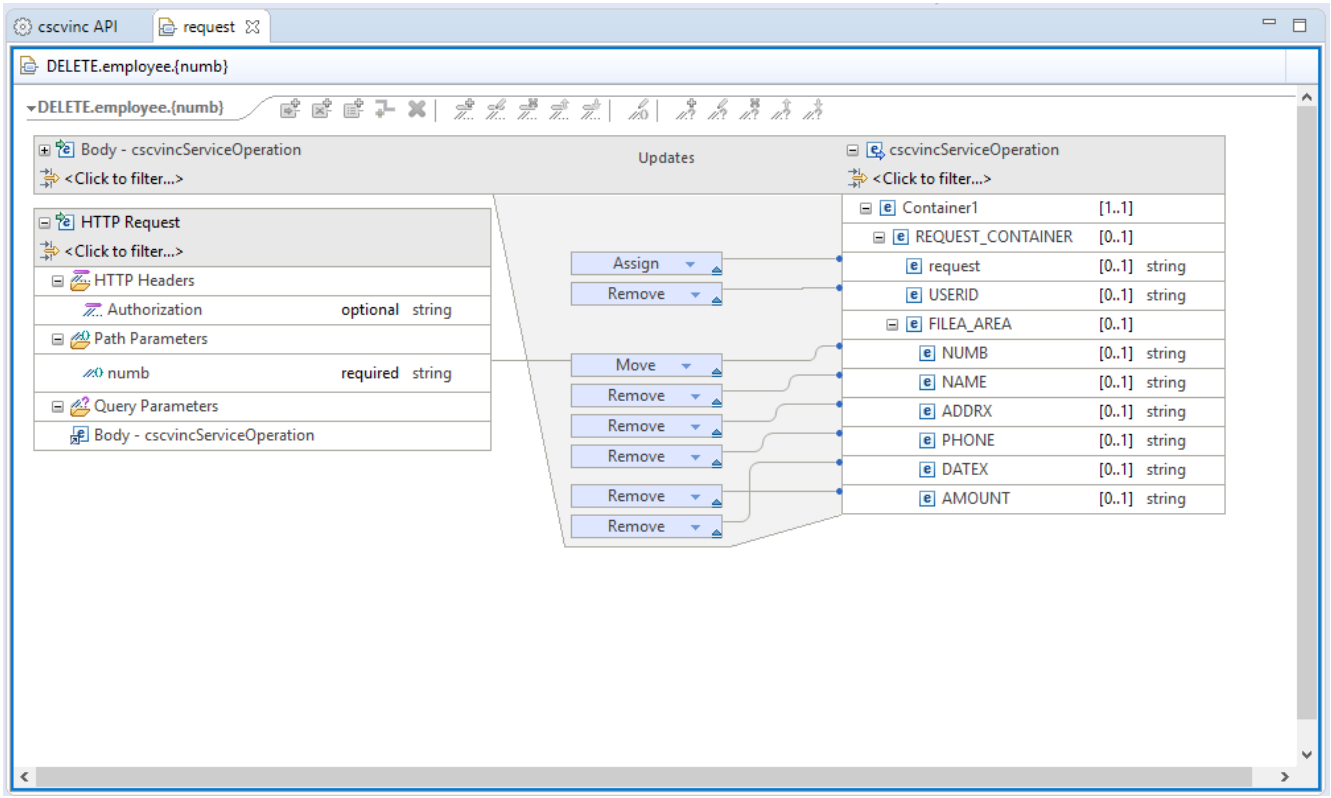
Remove unwanted fields to simplify the API (remember request was set to 01INQC in the SAR).

mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

API mapping: Remove or assign values to the fields exposed by service developer

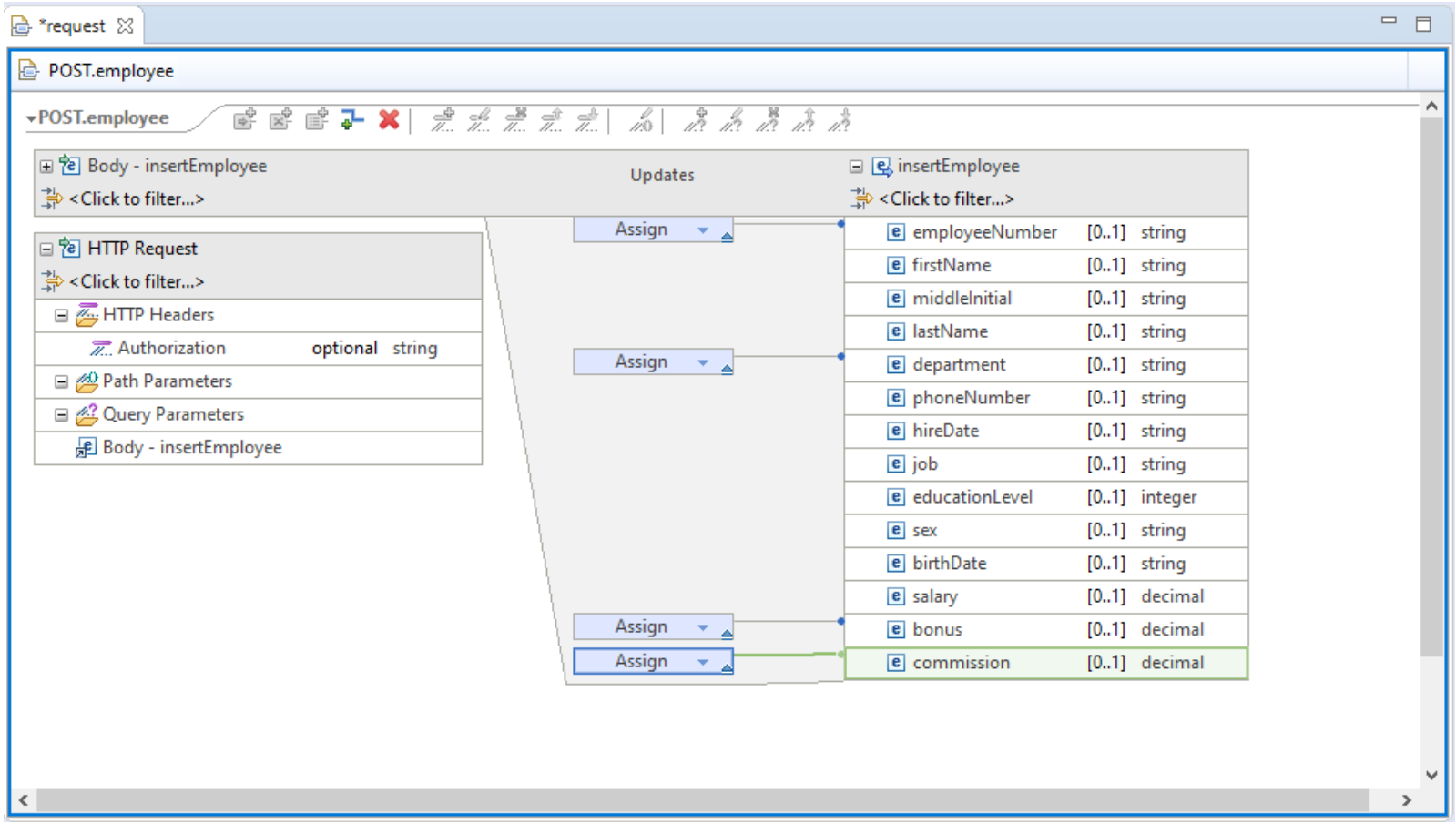


mitchj@us.ibm.com

API toolkit – API Editor and Db2 REST service

z/OS Connect EE

API mapping: Remove/Assign values columns exposed in Db2 REST service

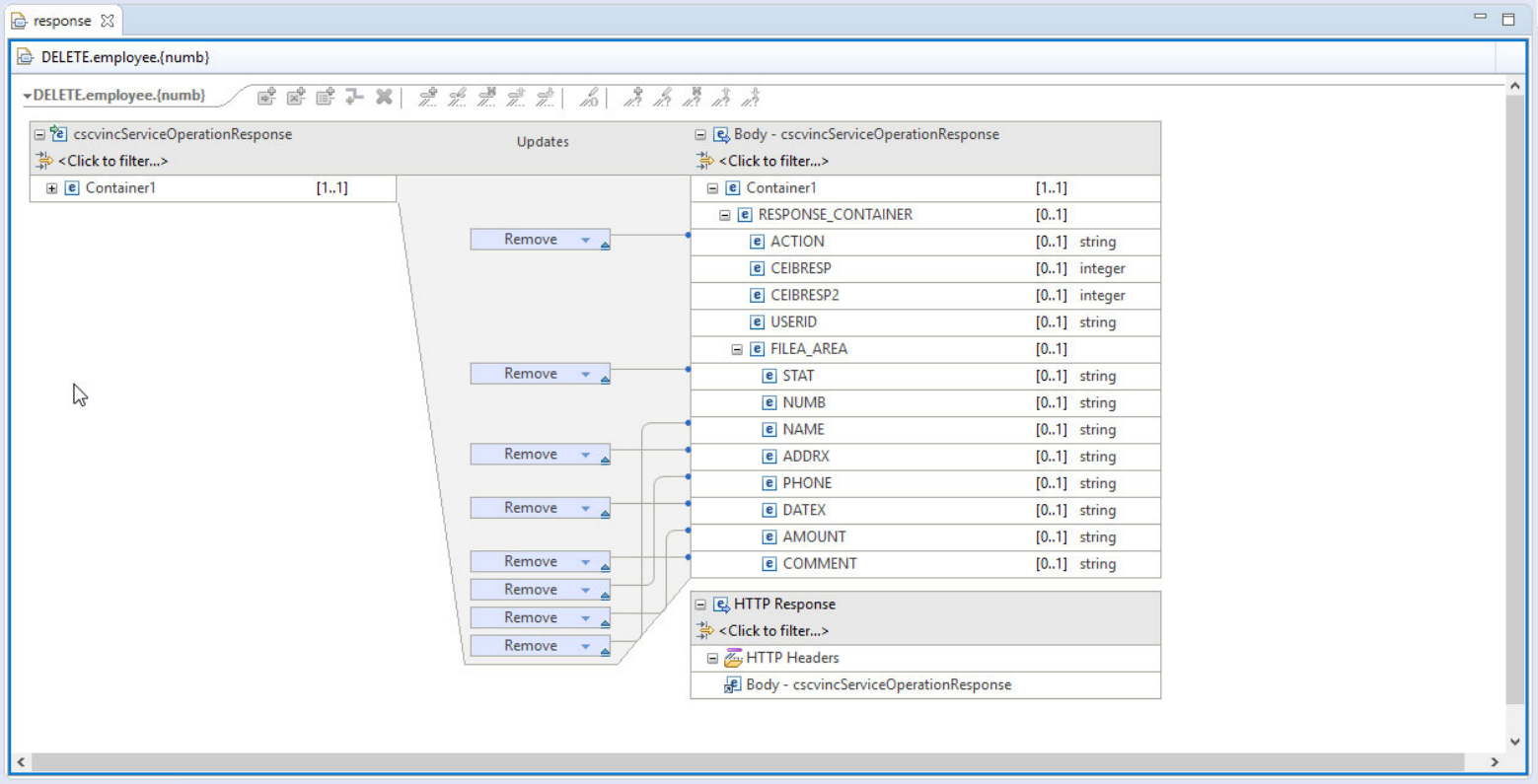


mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

API mapping: Allows the API Developer to remove fields from the response to tailor the API

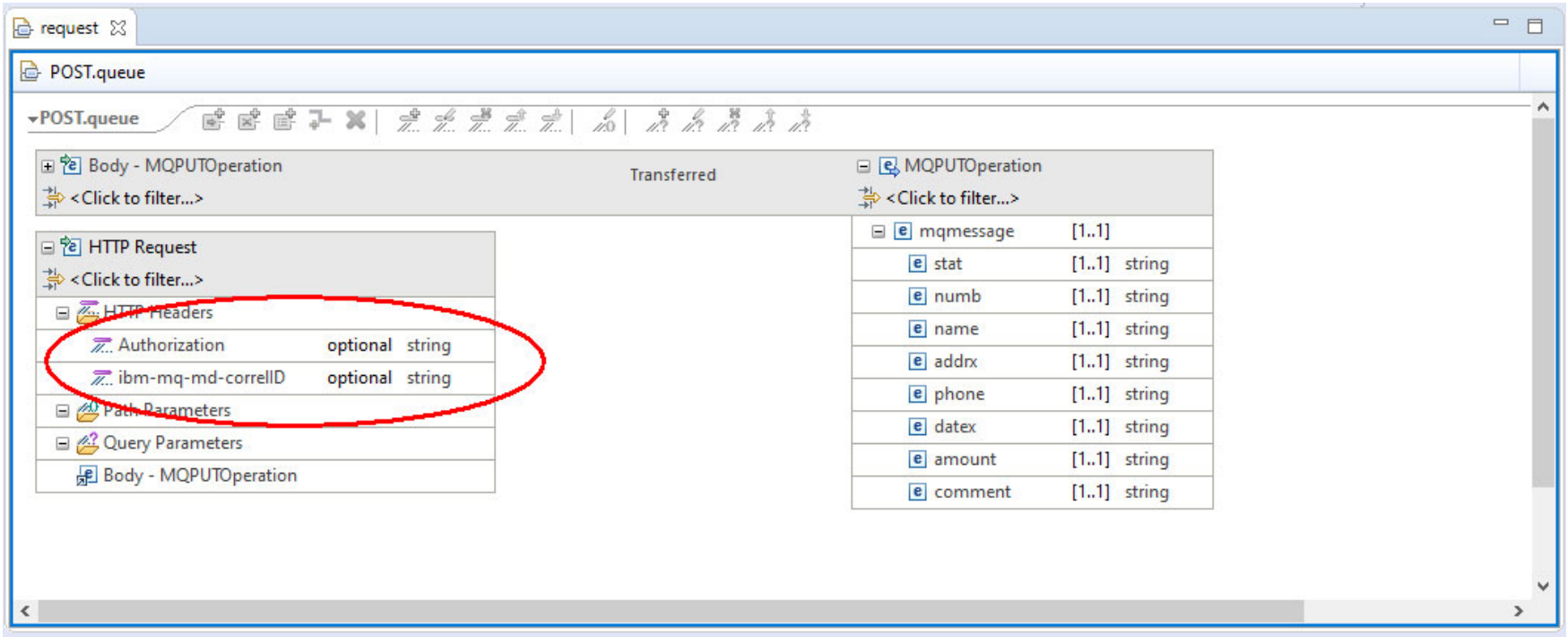


mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

API mapping: Allows adding HTTP header properties

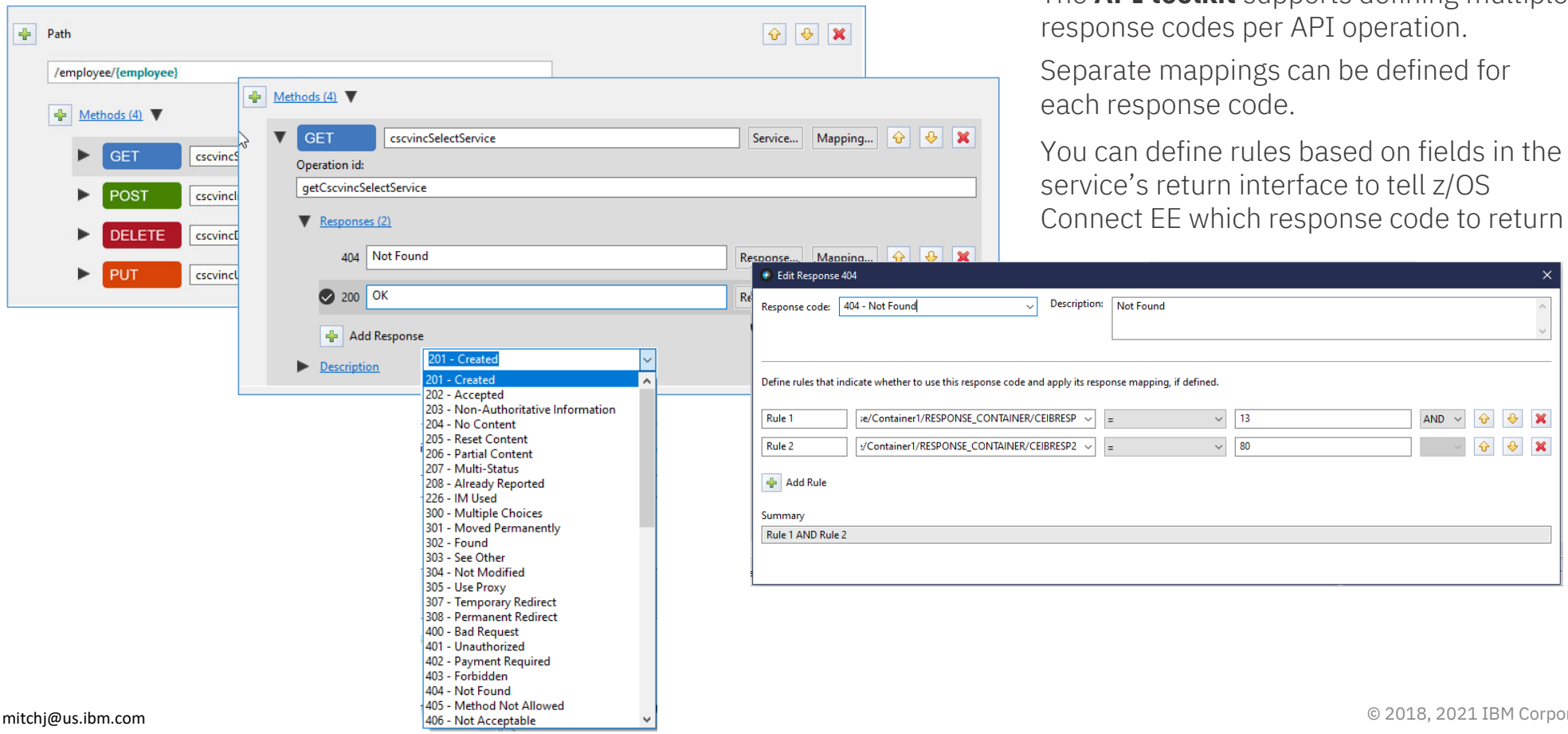


mitchj@us.ibm.com

API toolkit

z/OS Connect EE

API mapping: API definition with multiple response codes



The **API toolkit** supports defining multiple response codes per API operation.

Separate mappings can be defined for each response code.

You can define rules based on fields in the service's return interface to tell z/OS Connect EE which response code to return

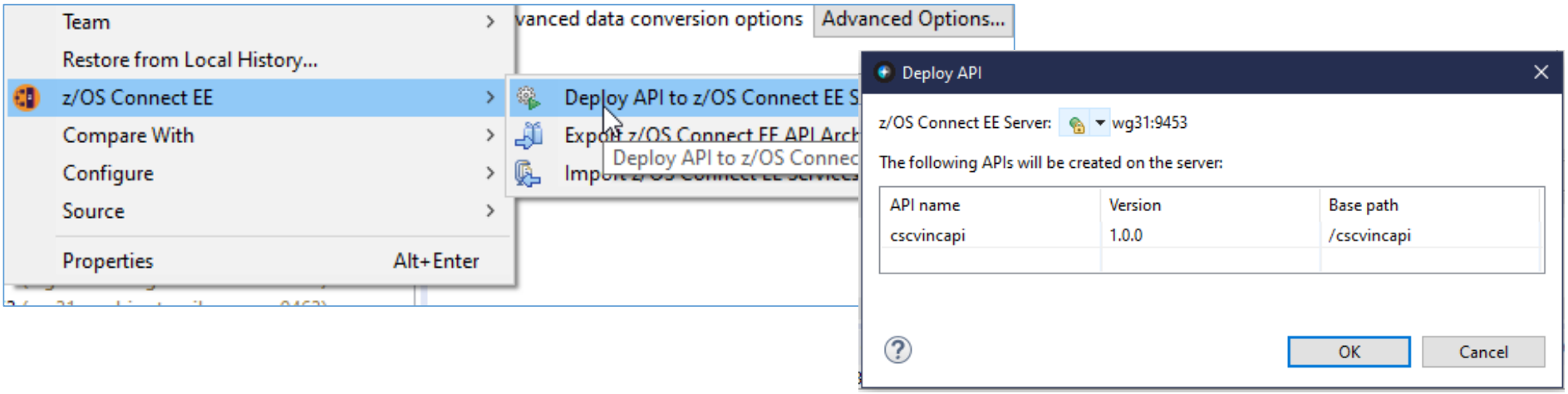
mitchj@us.ibm.com

API toolkit – API Editor

Server connection and API deployment

z/OS Connect EE

Manage z/OS Connect EE server connections in the **Host Connections** view:



Right-click deploy to server enables developers to quickly deploy, test, and iterate on their APIs.

z/OS Connect EE servers view allows you to start, stop, and remove APIs from a running server.

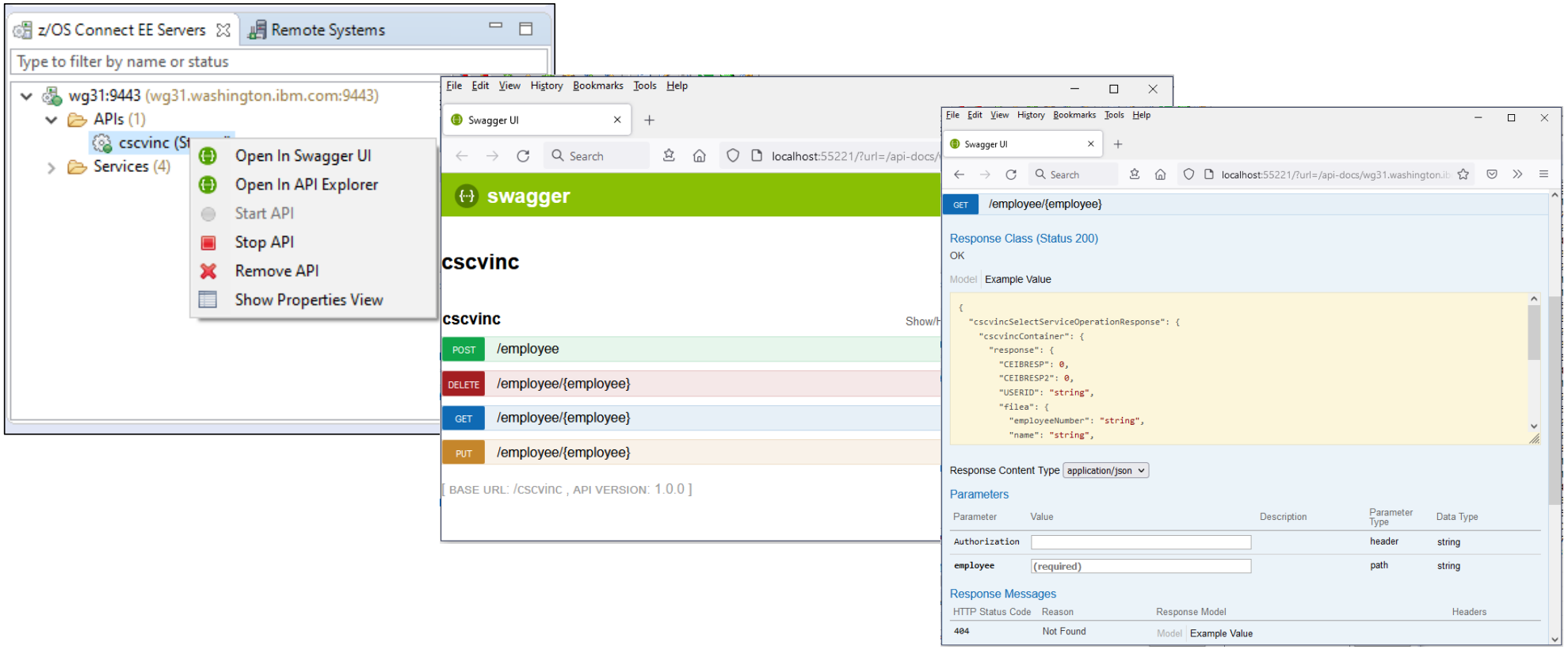
mitchj@us.ibm.com

API toolkit – API Editor

Testing with Swagger UI

z/OS Connect EE

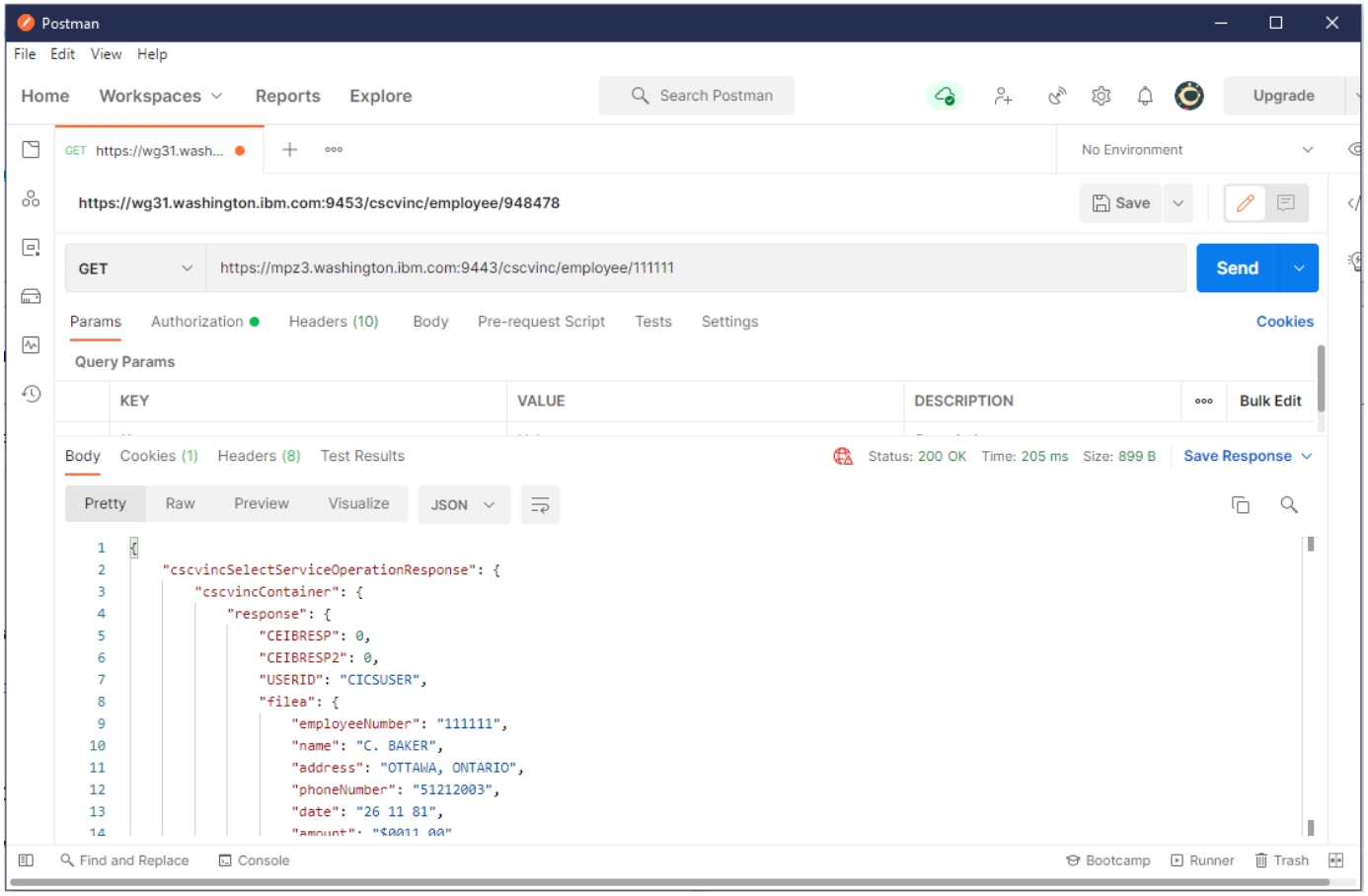
Test your deployed APIs directly with **Swagger UI** inside the editor.
No need to export the Swagger doc to a separate tool.



mitchj@us.ibm.com

API Testing with Postman

z/OS Connect EE

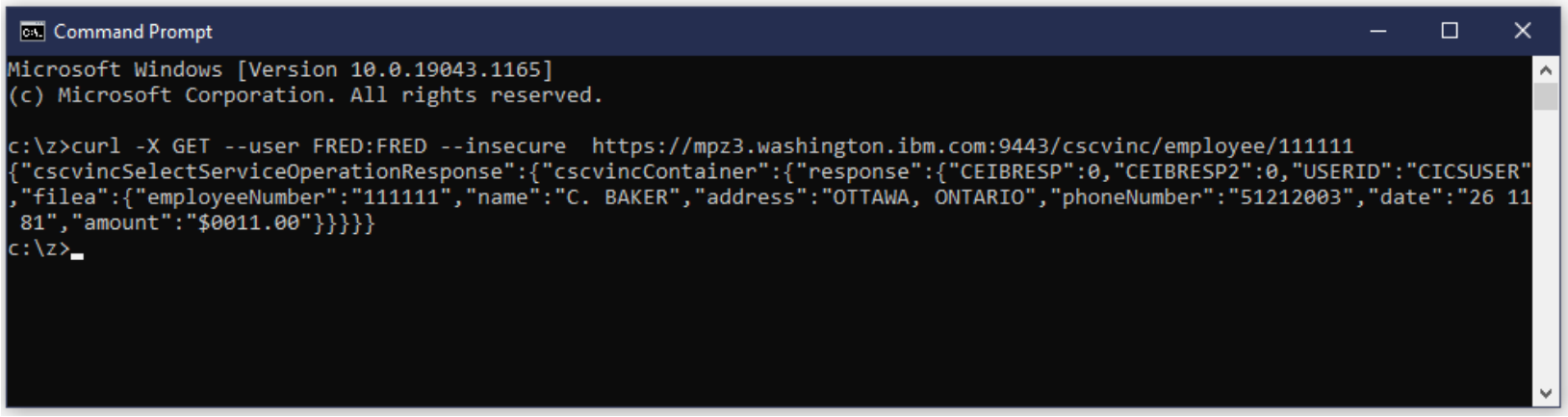


https://www.postman.com/downloads/

mitchj@us.ibm.com

API Testing with cURL

z/OS Connect EE

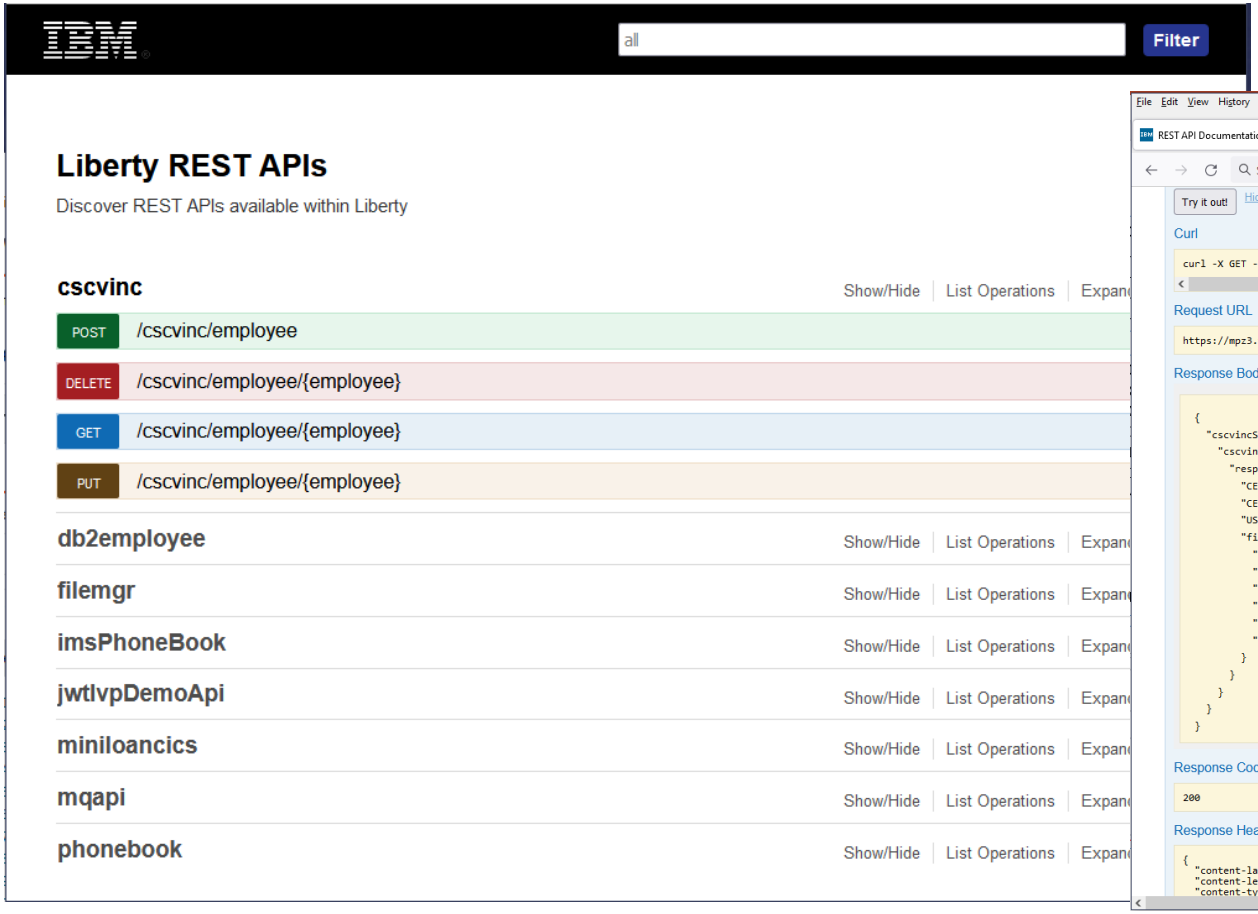


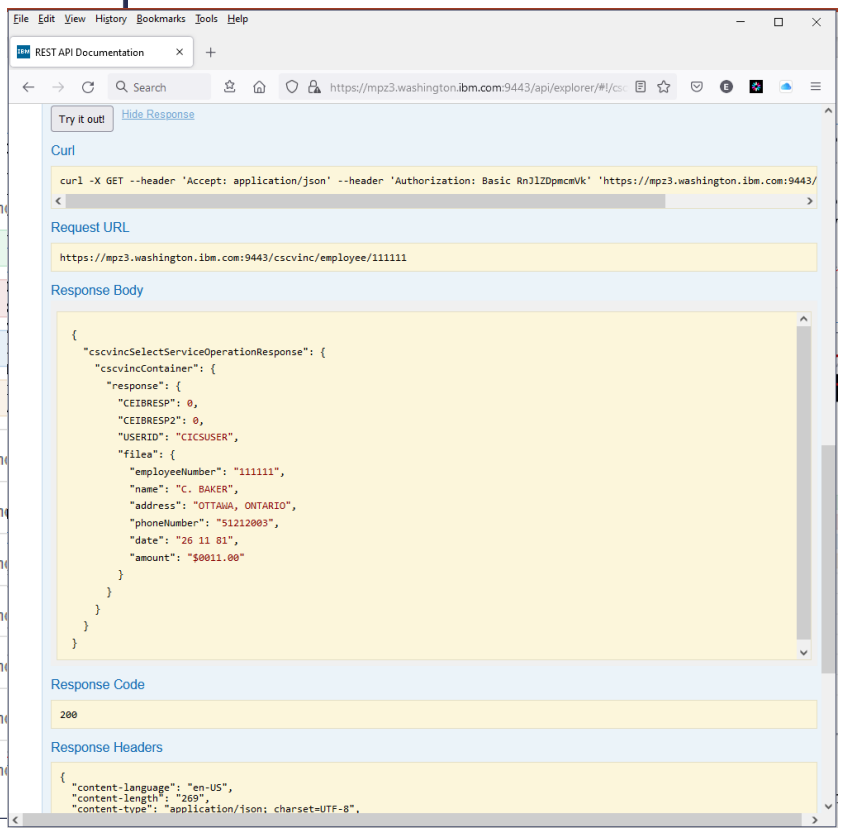
https://curl.se/download.html

mitchj@us.ibm.com

API Testing with the API Explorer (zCEE V3.0.48)

z/OS Connect EE





mitchj@us.ibm.com

/common_scenarios

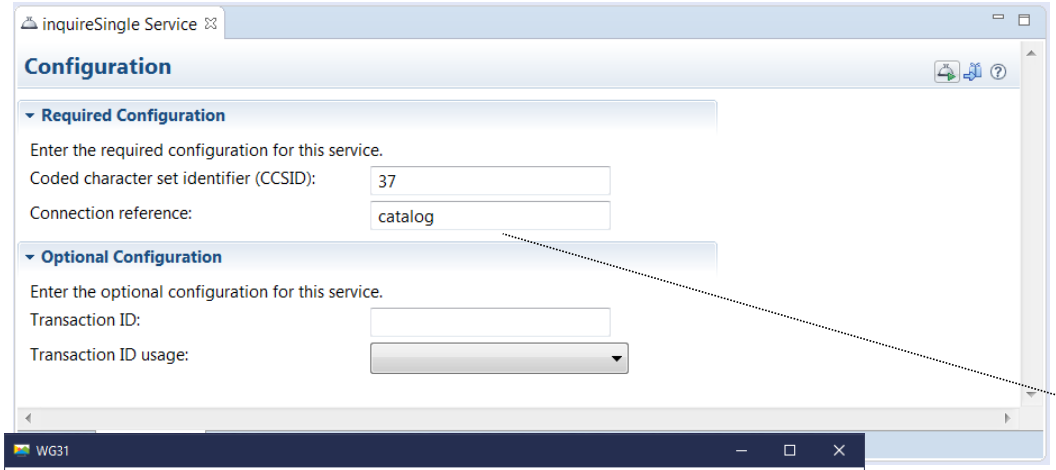
Typical server provider connections patterns

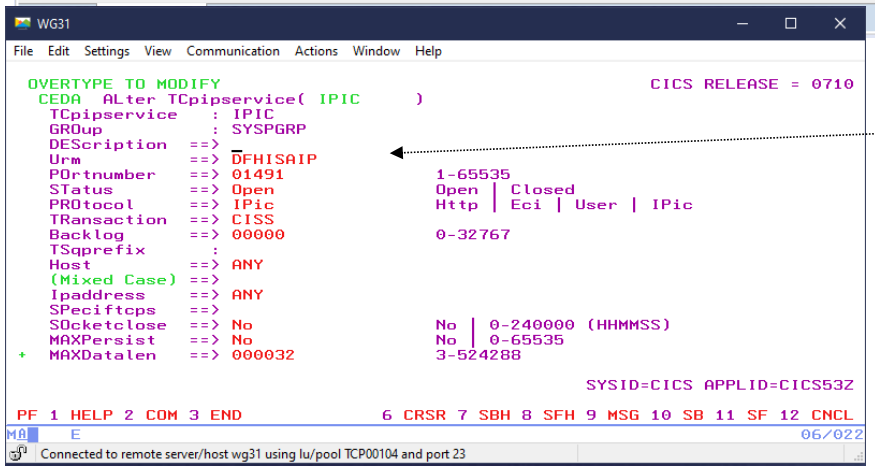
mitchj@us.ibm.com

Accessing a CICS program using IPIC

z/OS Connect EE

The server.xml file is the key configuration file:





Features are functional building blocks. When configured here, that function becomes available to the Liberty server

catalog.xml

Design Source

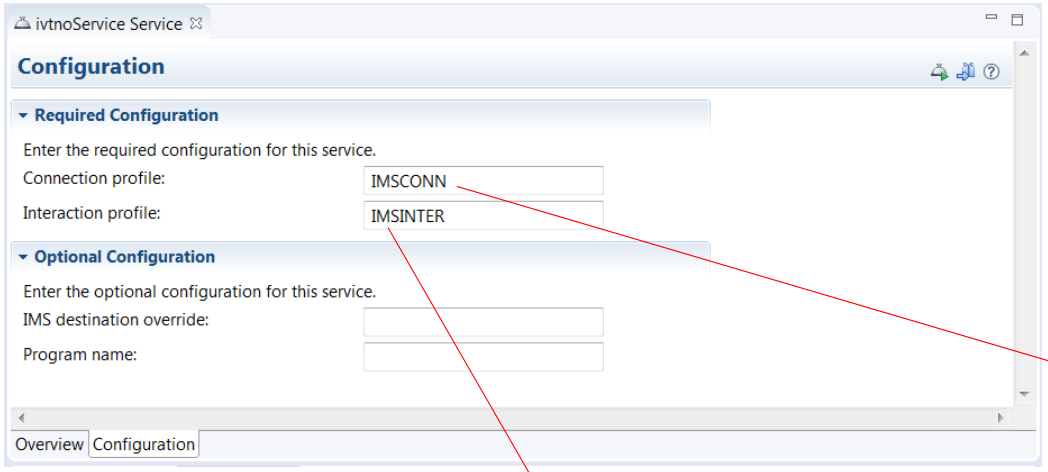
```
1 <server description="CICS IPIC - catalog">
2
3 <!-- Enable features -->
4 <featureManager>
5   <feature>zosconnect:cicsService-1.0</feature>
6 </featureManager>
7
8 <zosconnect_cicsIpicConnection id="catalog"
9   host="wg31.washington.ibm.com"
10  port="1491"
11  transid="CSMI"
12  transidUsage="EIB_AND_MIRROR"/>
13
14 </server>
15
```

Define IPIC connection to CICS

mitchj@us.ibm.com

IMS Connections and Interactions (server XML)

z/OS Connect EE



IMS Connect HWSCFG

```
HWS=(ID=IMS14HWS,XIBAREA=100,RACF=Y,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=
5000)
DATASTORE=(GROUP=OTMAGRP,ID=IVP1,MEMBER=HWSMEM,TMEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS,TMEMBER=PLEX1)
ODACCESS=(ODBMAUTOCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000),ODBMTMOT=6000)
```

Connection

```
<server>
<imsmobile_imsConnection comment="" connectionFactoryRef="CF1" connectionTimeout="-1" connectionType="IMSCONNECT" id="IMSCONN"/>
<connectionFactory containerAuthDataRef="Connection1_Auth" id="CF1">
          <properties.gmoa hostName="wg31.washington.ibm.com" portNumber="4000"/>
</connectionFactory>

<authData id="Connection1_Auth" password="encryptedPassword1" user="userName1"/>
</server>
```

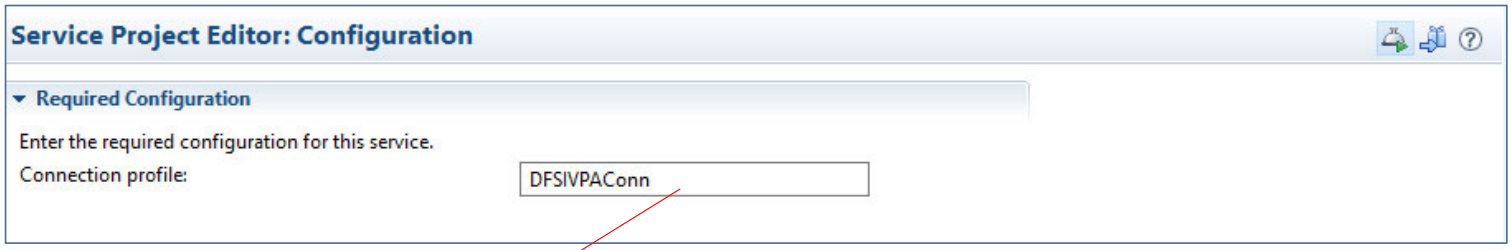
Interaction

```
<server>
<imsmobile_interaction comment="" commitMode="1" id="IMSINTER" imsConnectCodepage="Cp1047" imsConnectTimeout="0"
          imsDatastoreName="IVP1" interactionTimeout="-1" ltermOverrideName="" syncLevel="0"/>
</server>
```

mitchj@us.ibm.com

IMS Connection Factory in the server XML

z/OS Connect EE



ConnectionFactory

```
<connectionFactory id="DFSIVPAConn">
<properties.imsudbJLocal
   databaseName="DFSIVPA"
   datastoreName="IVP1"
   datastoreServer="wg31.washington.ibm.com"
   driverType="4"
   portNumber="5555"
   user="USER1"
   password="password"
   flattenTables="True"/>
</connectionFactory>
```

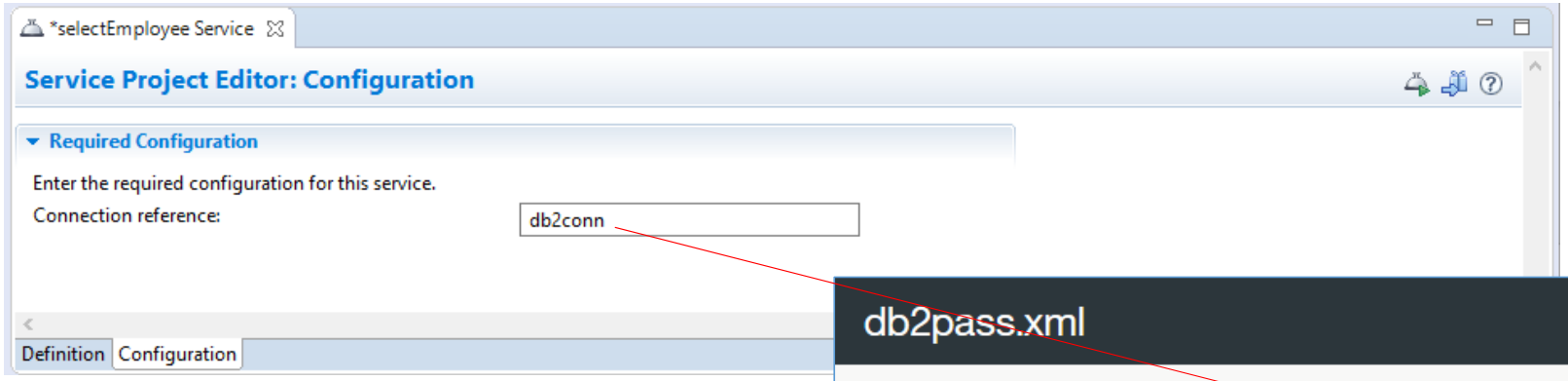
IMS Connect HWSCFG

```
HWS=(ID=IMS14HWS,XIBAREA=100,RACF=N,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)
DATASTORE=(GROUP=OTMAGRP,ID=IVP1,MEMBER=HWSMEM,TMEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS,TMEMBER=PLEX1)
ODACCESS=(ODBMAUTOCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000),ODBMTMOT=6000)
```

mitchj@us.ibm.com

Accessing a Db2 REST service resource

z/OS Connect EE




```
DSNL004I  -DSN2 DDF START
COMPLETE
          LOCATION  DSN2LOC
          LU
USIBMWZ.DSN2APPL
          GENERICLU -NONE
          DOMAIN
WG31.WASHINGTON.IBM.COM
          TCPPORT   2446
          SECPORT   2445
          RESPORT   2447
```

db2pass.xml

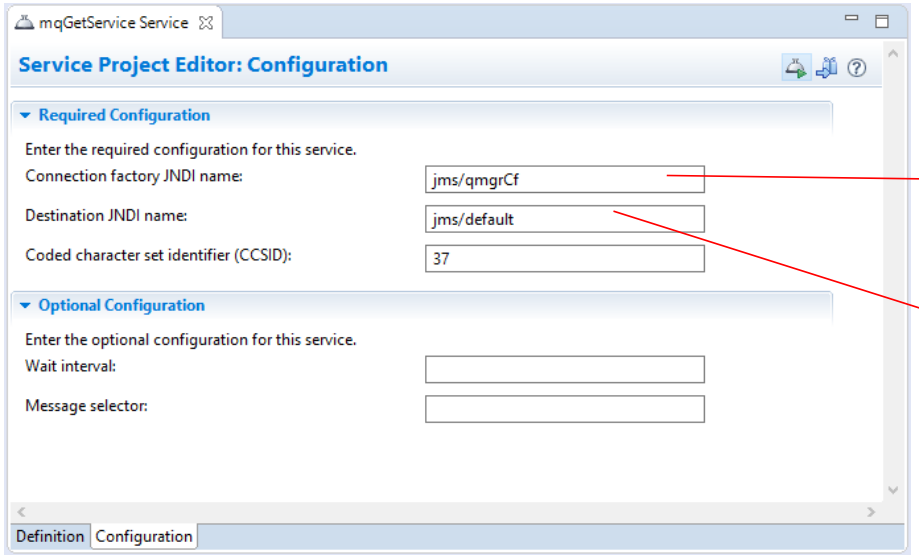
Design | Source

```
1  <server description="DB2 REST">
2
3   <zosconnect_zosConnectServiceRestClientConnection id="db2conn"
4         host="wg31.washington.ibm.com"
5         port="2446"
6         basicAuthRef="dsn2Auth" />
7
8   <zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
9         applName="DSN2APPL"/>
10
11 </server>
12
```

mitchj@us.ibm.com

Using JMS to access MQ (One-Way)

z/OS Connect EE



mq.xml

Read only | Close

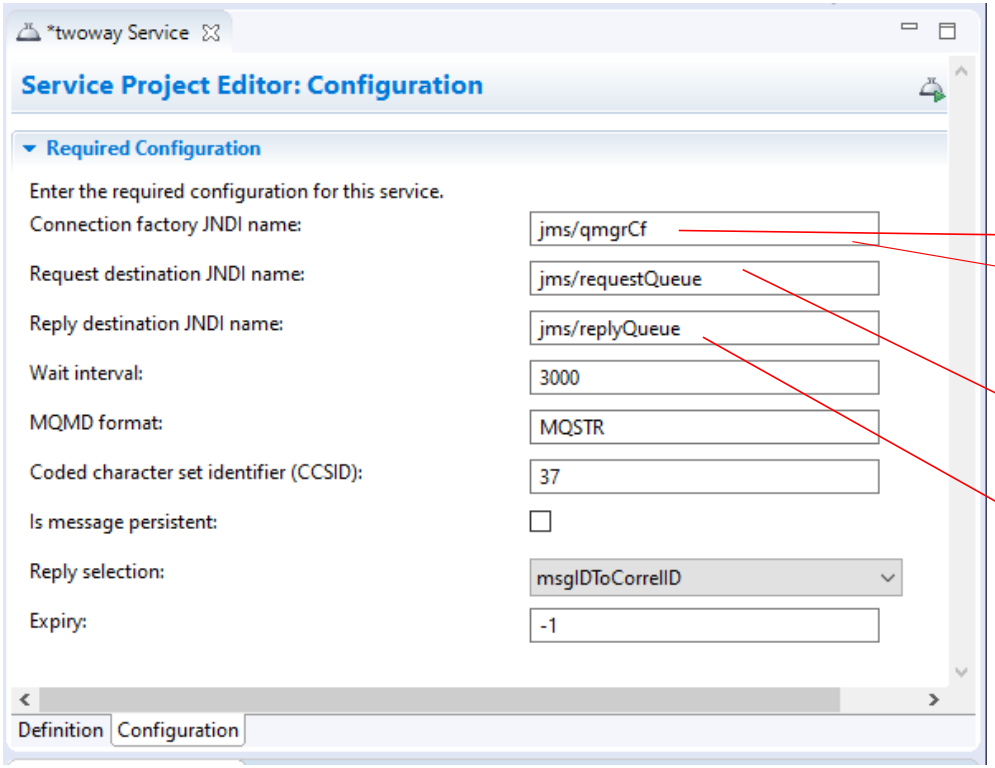
Design | Source

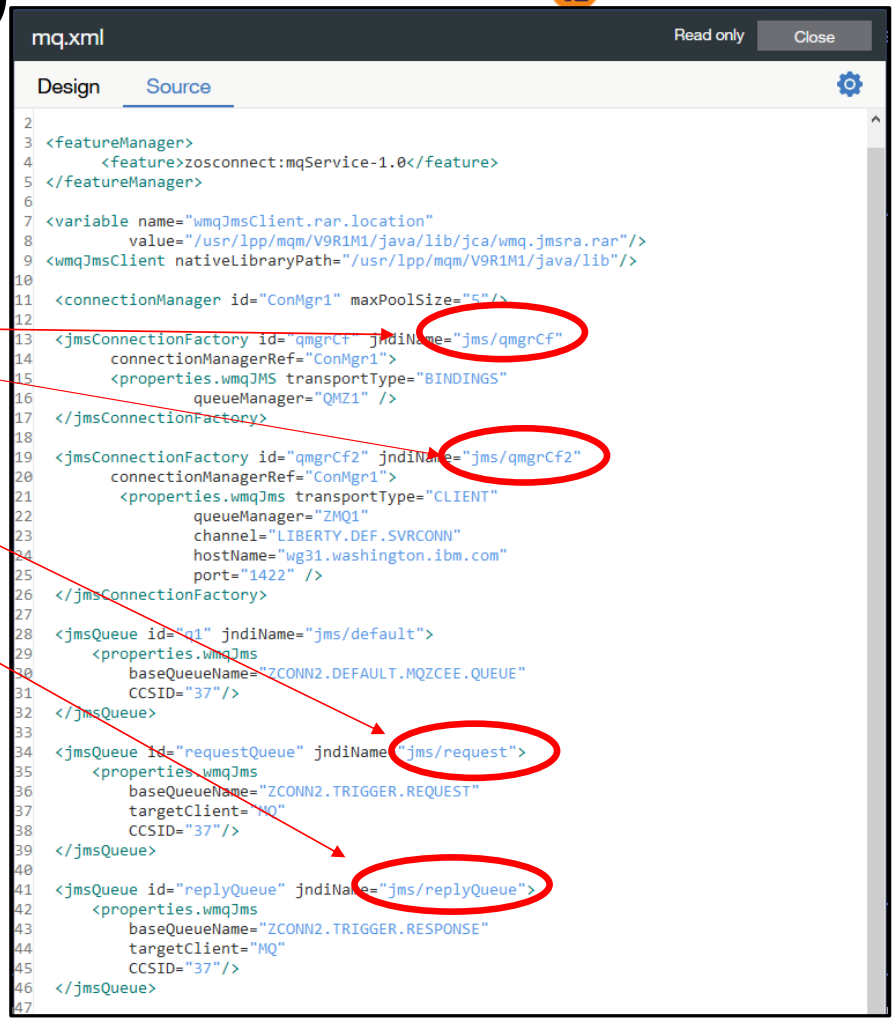
```
<featureManager>
      <feature>zosconnect:mqService-1.0</feature>
</featureManager>

<variable name="wmqJmsClient.rar.location"
        value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
<wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>

 <connectionManager id="ConMgr1" maxPoolSize="5"/>

 <jmsConnectionFactory id="qmgrCf" jndiName="jms/qmgrCf"
       connectionManagerRef="ConMgr1">
       <properties.wmqJMS transportType="BINDINGS"
                queueManager="QMZ1" />
 </jmsConnectionFactory>

 <jmsConnectionFactory id="qmgrCf2" jndiName="jms/qmgrCf2"
       connectionManagerRef="ConMgr1">
        <properties.wmqJms transportType="CLIENT"
                queueManager="ZMQ1"
                channel="LIBERTY.DEF.SVRCONN"
                hostName="wg31.washington.ibm.com"
                port="1422" />
 </jmsConnectionFactory>

 <jmsQueue id="q1" jndiName="jms/default">
     <properties.wmqJms
         baseQueueName="ZCONN2.DEFAULT.MQZCEE.QUEUE"
         CCSID="37"/>
 </jmsQueue>

 <jmsQueue id="requestQueue" jndiName="jms/request">
     <properties.wmqJms
         baseQueueName="ZCONN2.TRIGGER.REQUEST"
         targetClient="MQ"
         CCSID="37"/>
 </jmsQueue>

 <jmsQueue id="replyQueue" jndiName="jms/replyQueue">
     <properties.wmqJms
         baseQueueName="ZCONN2.TRIGGER.RESPONSE"
         targetClient="MQ"
         CCSID="37"/>
 </jmsQueue>
```

mitchj@us.ibm.com

Using JMS to access MQ (Two-Way)

z/OS Connect EE

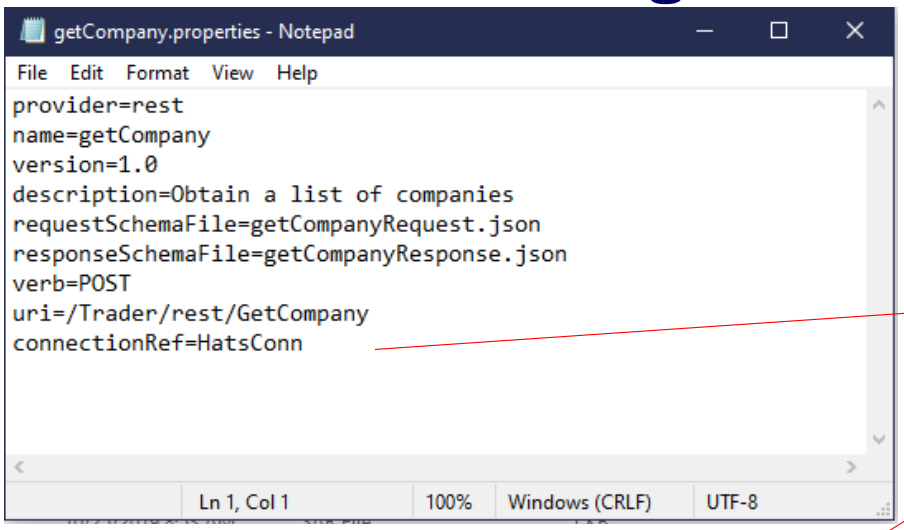




mitchj@us.ibm.com

HATS server XML configuration

z/OS Connect EE

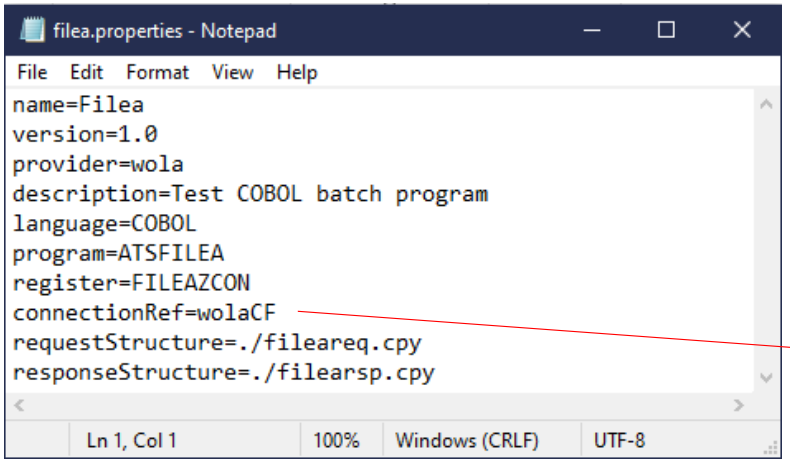


HATS Liberty server.xml

```
<!-- To access this server from a remote client, add a host attribute to the following element, e.g. host="*" -->
<httpEndpoint id="defaultHttpEndpoint"
              httpPort="29080"
              host="*"
              httpsPort="29443" />
```

mitchj@us.ibm.com

MVS batch server XML




```
      * SET THE VALUES FOR USE WITH WOLA REGISTRATION
           MOVE 'FILEAZCON'                    TO REG-REGNAME.
           MOVE 'ZCEESRVR'                     TO REG-DAEMONGRP.
           MOVE 'ZCEESRVR'                     TO REG-NODE.
           MOVE 'ZCEESRVR'                     TO REG-SVRNAME.
           MOVE 'ATSFILEA'                     TO SVC-SERVICE-NAME.
           INSPECT REG-DAEMONGRP CONVERTING ' ' to LOW-VALUES.
      * Register to a Local Liberty server
           CALL 'BBOA1REG' USING
                REG-DAEMONGRP,REG-NODE,REG-SVRNAME,REG-REGNAME,REG-MINCONN,REG-MAXCONN,REG-FLAGS,RSP-RC,RSP-RSN.
```

mitchj@us.ibm.com

DVM server XML

z/OS Connect EE

Server Config

dvs.xml — Read only — Close

Design | Source

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <server description="new server">
3
4    <!-- Enable features -->
5    <featureManager>
6        <feature>usr:dvsProvider</feature>
7        <feature>zosLocalAdapters-1.0</feature>
8    </featureManager>
9
10 <!--   Adapter Details with WOLA Group Name (ZCEEDVM) -->
11   <zosLocalAdapters wolaName3="NAME3"
12                     wolaName2="NAME2"
13                     wolaGroup="ZCEEDVM"/>
14
15 <!--   DVS Service Details with Register Name (ZCEEDVM) -->
16   <zosconnect_zosConnectService invokeURI="/dvs"
17    serviceDescription=""
18    serviceRef="dvsService"
19    serviceName="dvsService"
20    id="zosConnectDvsService"/>
21
22   <usr_dvsService invokeURI="/dvs"
23    serviceName="DVSS1"
24    registerName="ZCEEDVM"
25    connectionFactoryRef="wolaCF"
26    id="dvsService"/>
27
28   <connectionFactory jndiName="eis/ola" id="wolaCF">
29       <properties.ola/>
30   </connectionFactory>
31
32   <zosconnect_zosConnectService serviceRef="svc1"
33    serviceAsyncRequestTimeout="600s"
34    serviceName="dvs1" id="sdef1"/>
35
36     <zosconnect_localAdaptersConnectService
37      connectionWaitTimeout="7200"
38      connectionFactoryRef="wolaCF"
39      serviceName="DVSS1"
40      registerName="ZCEEDVM"
41      id="svc1"/>
42
43 </server>
44
```

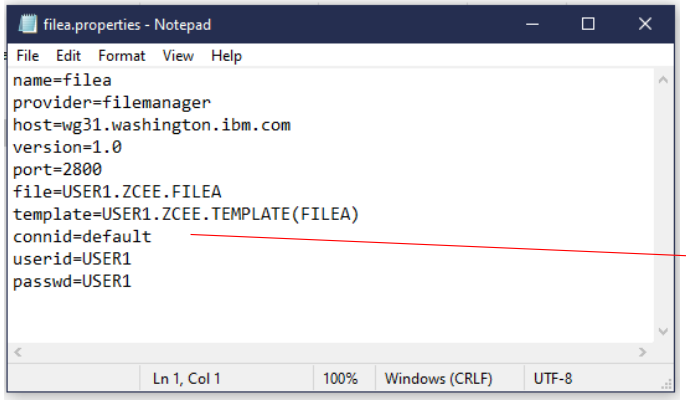
DVS.AVZS.SAVZEXEC(AVZSIN00)

```
/*-------------------------------------------------------------------*/
/* Enable z/OS Connect interface facility                            */
/*-------------------------------------------------------------------*/
if DoThis then
   do
    /*---------------------------------------------------------------*/
    /* The following parameter enables the z/OS Connect interface    */
    /* facility.                                                     */
    /*---------------------------------------------------------------*/
    "MODIFY PARM NAME(ZCONNECT)              VALUE(YES)"
    "MODIFY PARM NAME(NETWORKBUFFERSIZE)     VALUE(96K)"
    /*---------------------------------------------------------------*/
    /* The "DEFINE ZCPATH" command(s) can be used to define          */
    /* paths to z/OS Connect regions to handle requests.             */
    /* Use a separate '"DEFINE ZCPATH" command to define each        */
    /* path required (Note that a single path can handle             */
    /* several different requests)                                   */
    /* refer to the documentation for details about the parameters,  */
    /* and information about optional parameters.                    */
    /*---------------------------------------------------------------*/
    "DEFINE ZCPATH",
    "  NAME(ZCEE)                    ",
    "  RNAME(ZCEEDVM)                ",
    "  WNAME(ZCEEDVM)                ",
    ""
   end
```

mitchj@us.ibm.com

File Manager server XML

z/OS Connect EE



SYS1.PROCLIB(IPVSRV1)

```
//IPVSRV1  PROC PORT=2800,FAMILY='AF_INET',TRACE=N
//      SET ENV=''
//RUN       EXEC PGM=IPVSRV,REGION=40M,
//             PARM=('&ENV/&PORT &FAMILY &TRACE')
// SET IPV=SYSP.ADFZ.JCL                        <== Update HLQ
//* Common component authorised library
//STEPLIB  DD DISP=SHR,DSN=ADFZ.SIPVMODA         <== ADFzCC APF LIBRARY
//SYSPRINT DD SYSOUT=*
//IPVTRACE DD SYSOUT=*                           <== OUTPUT if Tracing
//STDOUT   DD SYSOUT=*
//* Server wide, then participating product configurations
//CONFIG   DD DISP=SHR,DSN=&IPV.(IPVCONFG)
```

mitchj@us.ibm.com

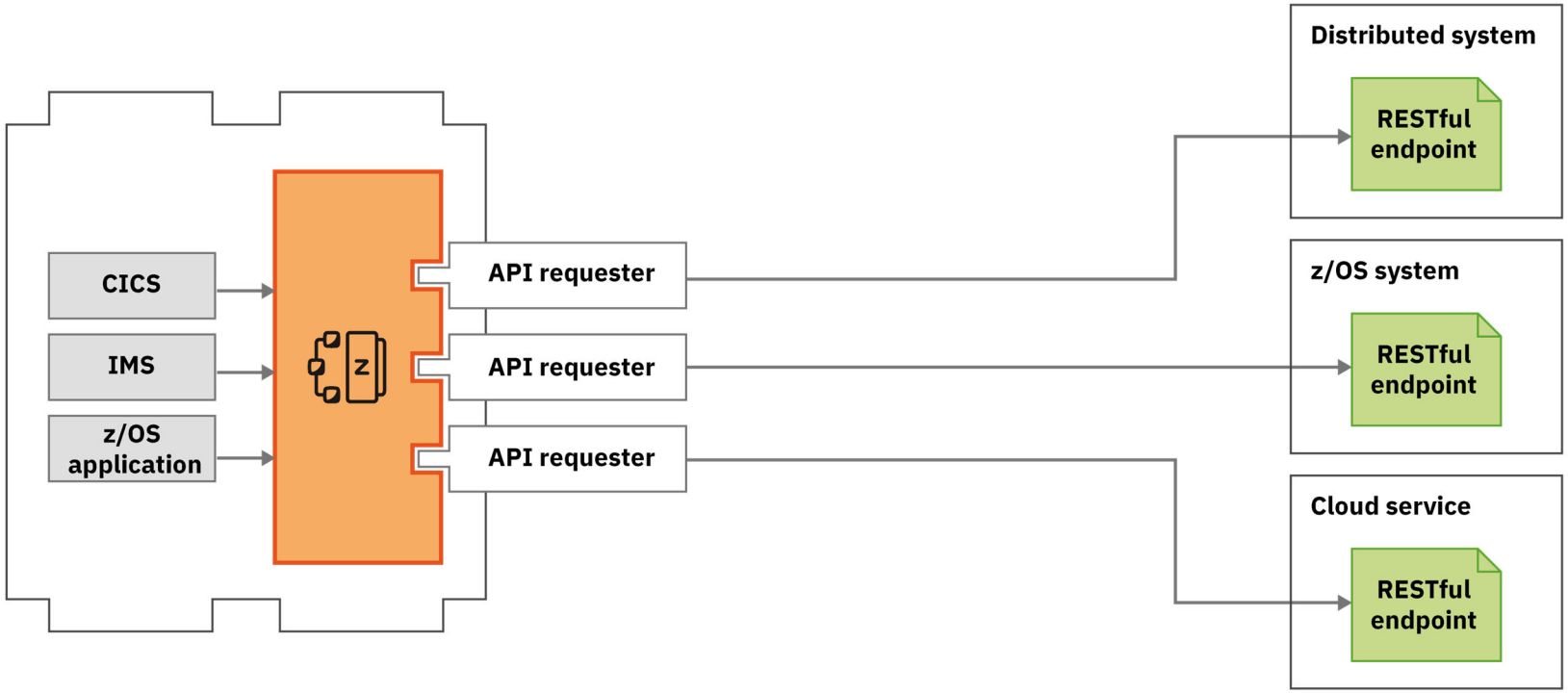
/api_toolkit/apiRequesters

Quick and easy **API mapping.**

mitchj@us.ibm.com

Use API requester to call external APIs from z/OS assets

z/OS Connect EE

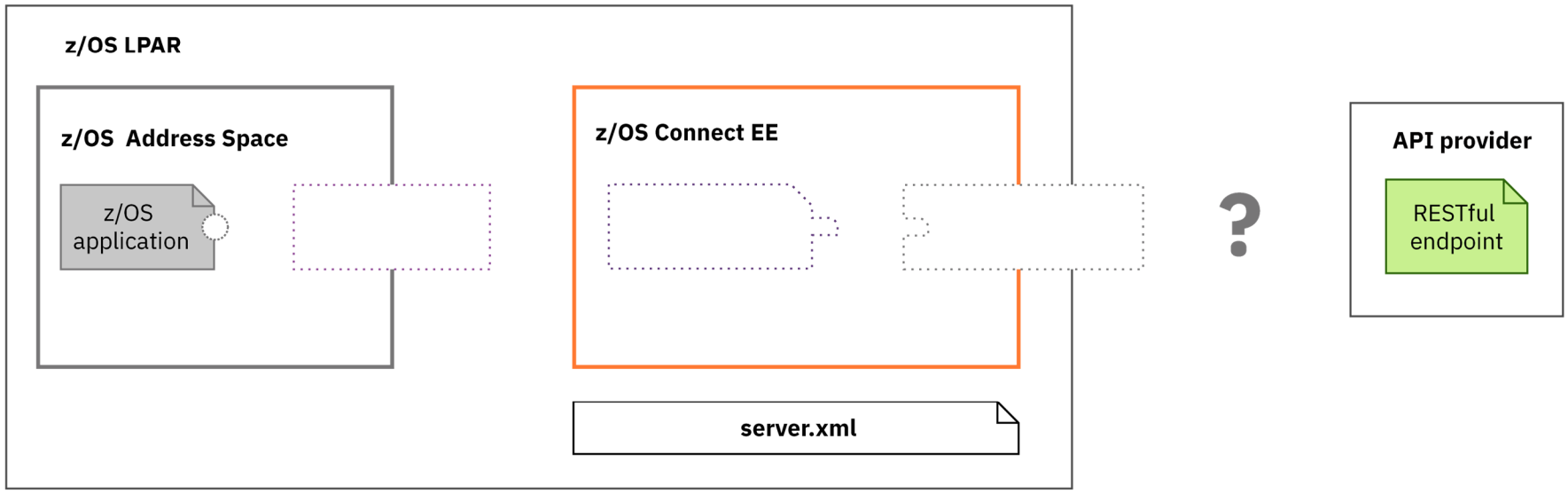


mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Starting point

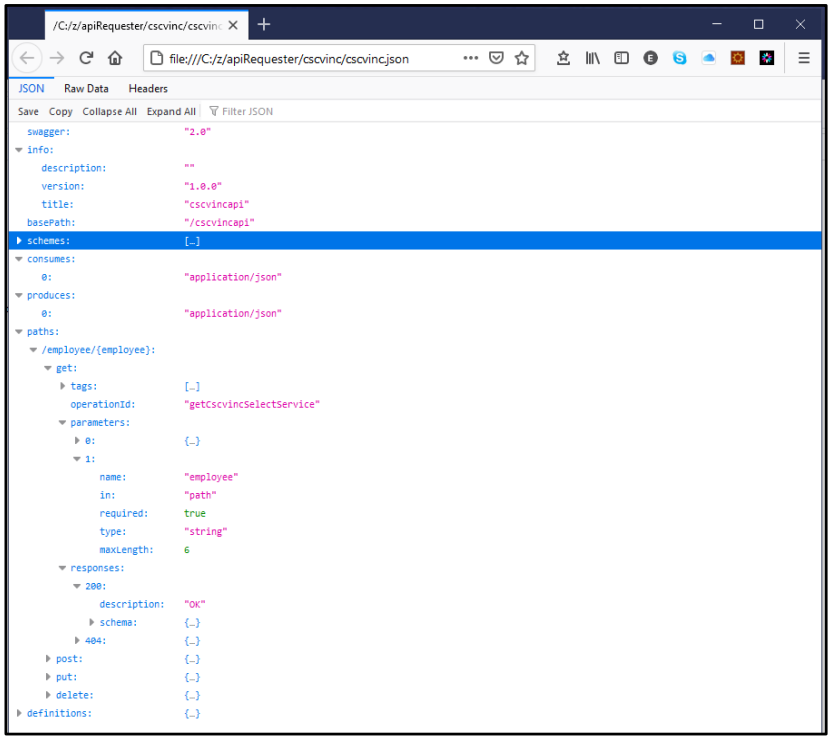


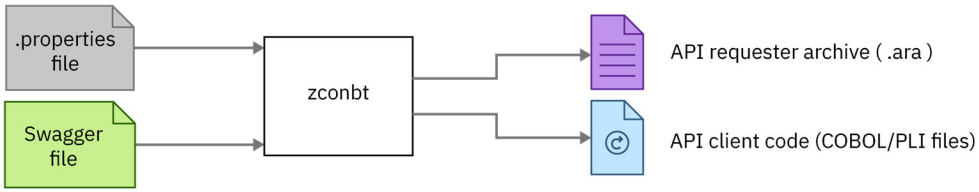
mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Generate API requester archive and API client code from Swagger





`.properties file`[#]

```
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

[#]Additional property file attributes, e.g., *defaultCharacterMaxLength*, *defaultArrayMaxItems*, etc. are described at **The build toolkit properties file** article at URL https://www.ibm.com/docs/en/zosconnect/3.0?topic=toolkit-build-properties-file

mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Using *zconbt* to generate API requester archive and API client code from Swagger

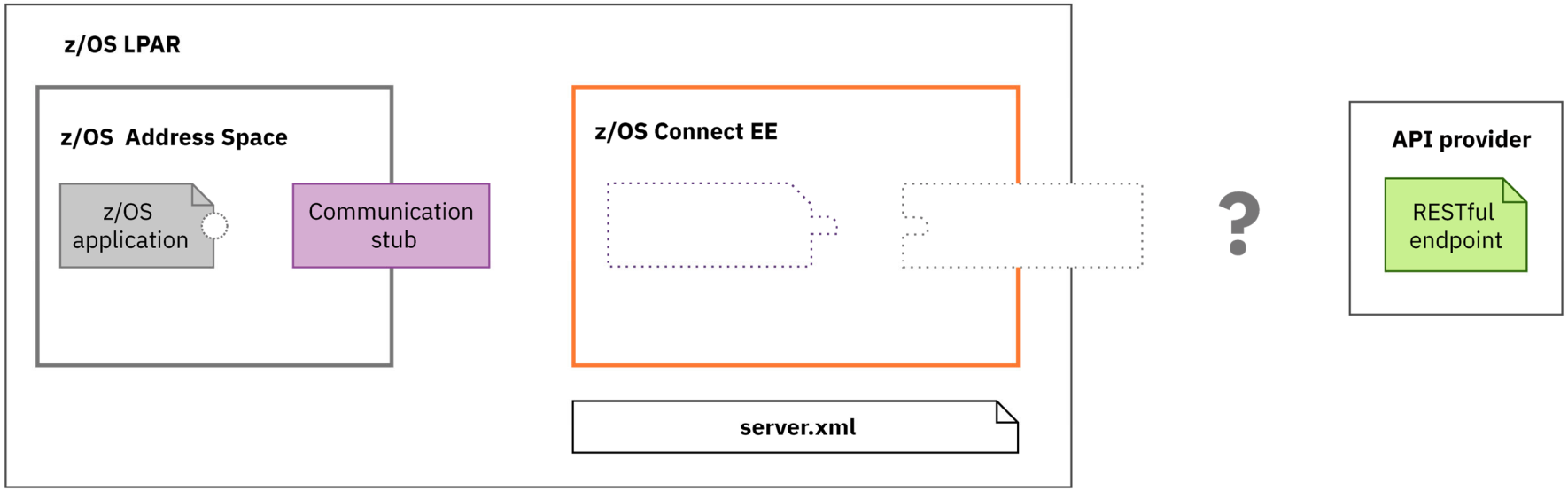
```
zconbt.bat -p=./cscvinc.properties -f=./cscvinc.ara
BAQB0000I: z/OS Connect Enterprise Edition 3.0 Build Toolkit Version 1.5 (20210816-0926).
BAQB0008I: Creating API requester archive from configuration file ./cscvinc.properties.
BAQB0040I: The generated API requester is automatically named cscvincapi_1.0.0 based on the title cscvincapi and version 1.0.0 of the API to be
called.
. . . .  .
Total 4 operation(s) (success: 4, ignored: 0) defined in api description file: ./cscvinc.json
------ Successfully processed operation(s) ------
operationId: getCscvincSelectService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: GET
- request data structure  : CSC00Q01
- response data structure : CSC00P01
- api info file           : CSC00I01

operationId: putCscvincUpdateService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: PUT
- request data structure  : CSC01Q01
- response data structure : CSC01P01
- api info file           : CSC01I01

operationId: postCscvincInsertService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: POST
- request data structure  : CSC02Q01
- response data structure : CSC02P01
- api info file           : CSC02I01

operationId: deleteCscvincDeleteService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: DELETE
- request data structure  : CSC03Q01
- response data structure : CSC03P01
- api info file           : CSC03I01


BAQB0009I: Successfully created API requester archive file ./cscvinc.ara.
```

mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Update the application by adding the copy books and a call to communication stub



Configure a communication stub.

- For CICS region systems using URIMAP resources
- For non CICS client the configuration is done via environment variables

ibm.biz/zosconnect-configure-comms-stub

mitchj@us.ibm.com

Steps to calling an external API

Include the generated copy books in a COBOL program

GETAPI

```
      * Error message structure
       01  ERROR-MSG.
           03 EM-ORIGIN                PIC X(8)  VALUE SPACES.
           03 EM-CODE                  PIC S9(9) COMP-5 SYNC VALUE 0.
           03 EM-DETAIL                PIC X(1024) VALUE SPACES.

      * Copy API Requester required copybook
       COPY BAQRINFO.

      * Request and Response
       01 API-REQUEST.
           COPY CSC02Q01.
       01 API_RESPONSE.
           COPY CSC02P01.
      * Structure with the API information
       01 API-INFO-OPER1.
           COPY CSC02I01.

      * Request and Response segment  used to store request and
```

API-REQUEST

CSC00I01 | CSC00Q01

```
      * JSON schema keyword 'minLength' value: '0'.
      * JSON schema keyword 'maxLength' value: '6'.
      * This field contains a varying length array of characters or
      *  binary data.
      *       09 employee-length              PIC S9999 COMP-5 SYNC.
      *       09 employee                     PIC X(6).
      *
      *
      * ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

           06 ReqPathParameters.
             09 employee-length              PIC S9999 COMP-5 SYNC.
             09 employee                     PIC X(6).
```

API-RESPONSE

CSC00I01 | CSC00Q01 | CSC00P01

```
      *
      * ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

           06 RespBody.

             09 cscvincSelectServiceOp-num    PIC S9(9) COMP-5 SYNC.

             09 cscvincSelectServiceOperatio.
               12 Container1.

                 15 RESPONSE-CONTAINER2-num       PIC S9(9) COMP-5
           SYNC.
```

API-INFO-OPER1

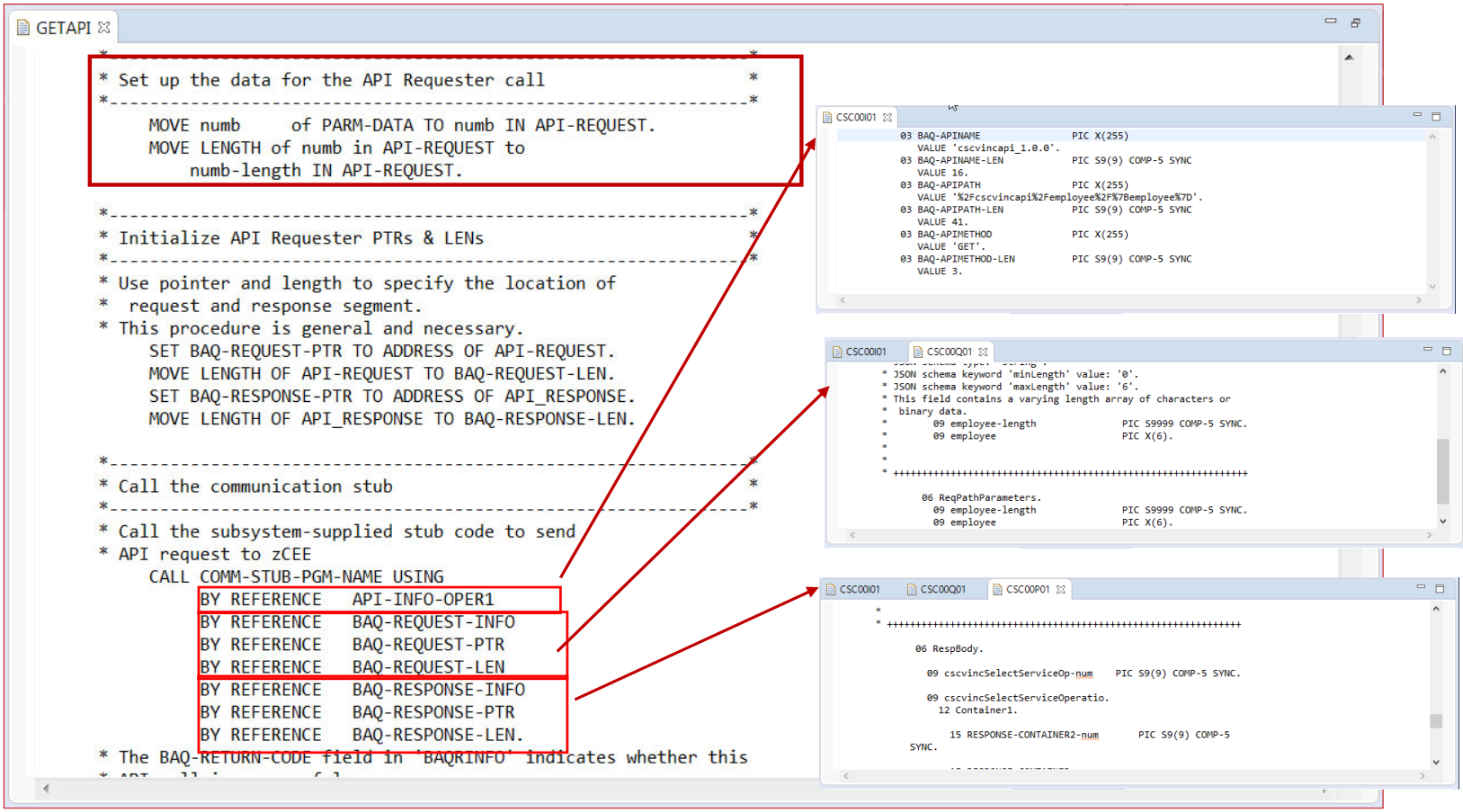
CSC00I01

```
           03 BAQ-APINAME                PIC X(255)
              VALUE 'cscvincapi_1.0.0'.
           03 BAQ-APINAME-LEN            PIC S9(9) COMP-5 SYNC
              VALUE 16.
           03 BAQ-APIPATH                PIC X(255)
              VALUE '%2Fcscvincapi%2Femployee%2F%7Bemployee%7D'.
           03 BAQ-APIPATH-LEN            PIC S9(9) COMP-5 SYNC
              VALUE 41.
           03 BAQ-APIMETHOD              PIC X(255)
              VALUE 'GET'.
           03 BAQ-APIMETHOD-LEN          PIC S9(9) COMP-5 SYNC
              VALUE 3.
```

mitchj@us.ibm.com

Steps to calling an external API

Add a call to the communication stub passing pointers to working storage of the copy books



mitchj@us.ibm.com

Steps to calling an external API

Access the results

GETAPI

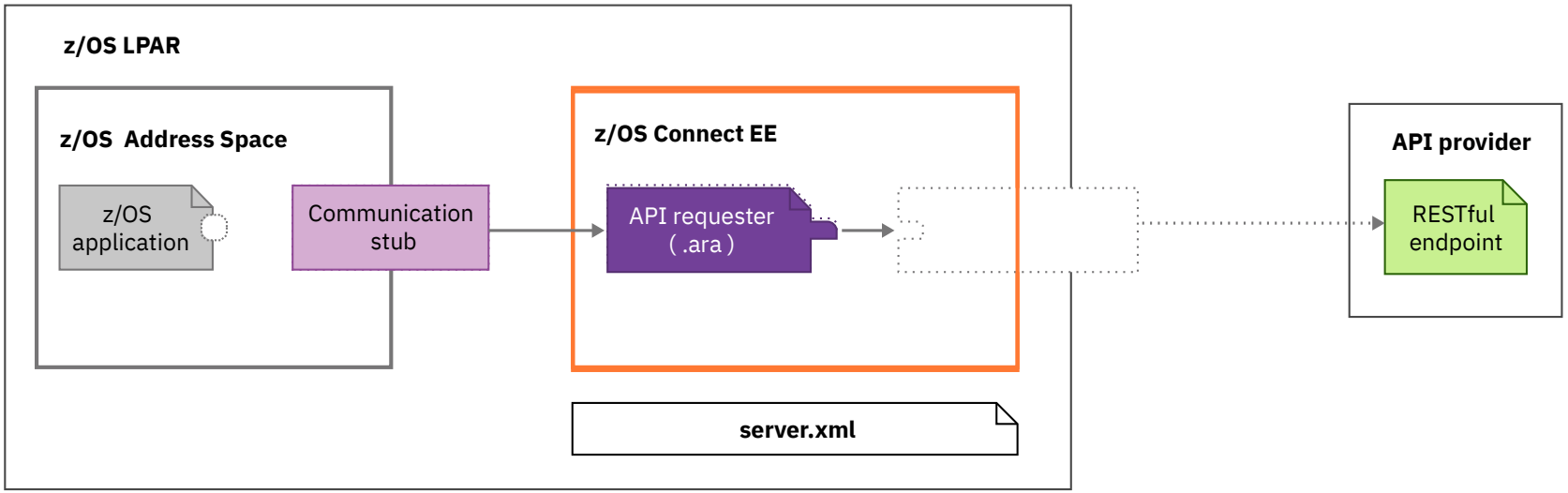
```
            BY REFERENCE   BAQ-RESPONSE-LEN.
      * The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
      * API call is successful.

      * When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is
      * successfully returned and fields in RESPONSE copybook
      * can be obtained. Display the translation result.
          IF BAQ-SUCCESS THEN
             DISPLAY "NUMB:   " numb2   of API_RESPONSE
             DISPLAY "NAME:   " name2   of API_RESPONSE
             DISPLAY "ADDRX:  " addrx2  of API_RESPONSE
             DISPLAY "PHONE:  " phone2  of API_RESPONSE
             DISPLAY "DATEX:  " datex2  of API_RESPONSE
             DISPLAY "AMOUNT: " amount2 of API_RESPONSE
             MOVE CEIBRESP of API_RESPONSE to EIBRESP
             MOVE CEIBRESP2 of API_RESPONSE to EIBRESP2
             DISPLAY "EIBRESP:    " EIBRESP
             DISPLAY "EIBRESP2:   " EIBRESP2
             DISPLAY "HTTP CODE:  " BAQ-STATUS-CODE

      * Otherwise, some error happened in API, z/OS Connect EE server
      * or communication stub. 'BAQ-STATUS-CODE' and
      * 'BAQ-STATUS-MESSAGE' contain the detailed information
      *  of this error.
          ELSE
             DISPLAY "Error code: " BAQ-STATUS-CODE
             DISPLAY "Error msg:" BAQ-STATUS-MESSAGE
             MOVE BAQ-STATUS-CODE TO EM-CODE
             MOVE BAQ-STATUS-MESSAGE TO EM-DETAIL
             EVALUATE TRUE
      * When error happens in API, BAQ-RETURN-CODE is BAQ-ERROR-IN-API.
      * BAQ-STATUS-CODE is the HTTP response code of API.
```

mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Deploy API requester (.ara) archive



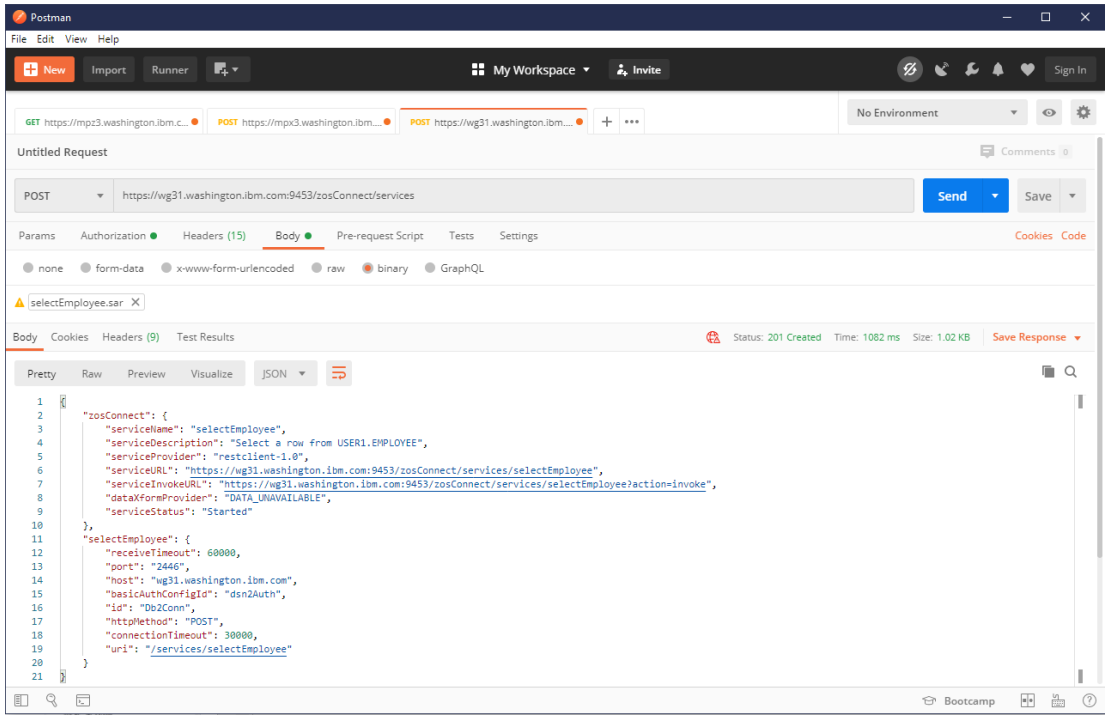
Deploy your API requester archive to the *apiRequesters* directory.

mitchj@us.ibm.com

Deploying Service Archive options

z/OS Connect EE

- Use SAR as request message and use HTTP POST
- Use URI path /zosConnect/services
- Postman or cURL



Command:

curl --data-binary @selectEmployee.sar --header "Content-Type: application/zip" https://mpxm:9453/zosConnect/services

Results:

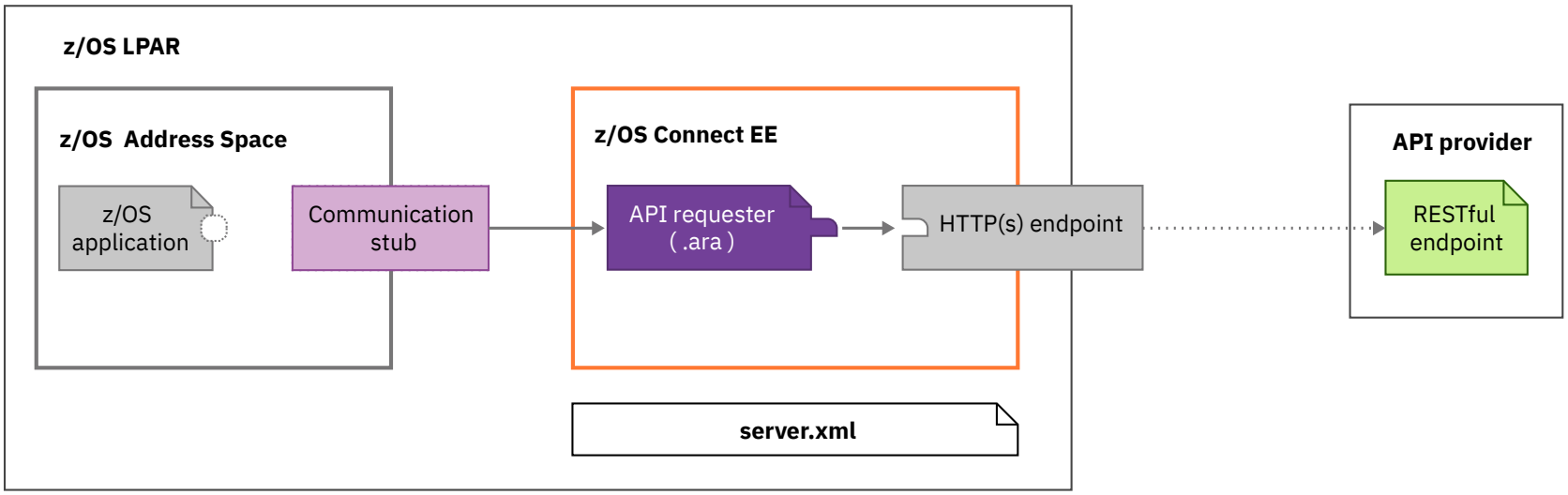

```
{"zosConnect":{"serviceName":"selectEmployee","serviceDescrip
tion":"Select a row from
USER1.EMPLOYEE","serviceProvider":"IBM_ZOS_CONNECT_SERVICE_RE
ST_CLIENT-1.0
","serviceURL":"https://mpxm:9453/zosConnect/services/selectE
mployee","serviceInvokeURL":"https://mpxm:9453/zosConnect/ser
vices/selectEmployee?action=invoke","dataXformProvider":"DATA
_UNAVAILABLE","serviceStatus":"Started"},"selectEmployee"
:{"receiveTimeout":0,"port":null,"host":null,"httpMethod":"PO
ST","connectionTimeout":0,"uri":"/services/selectEmployee"}}
```


mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Configure HTTP(S) endpoint configuration element



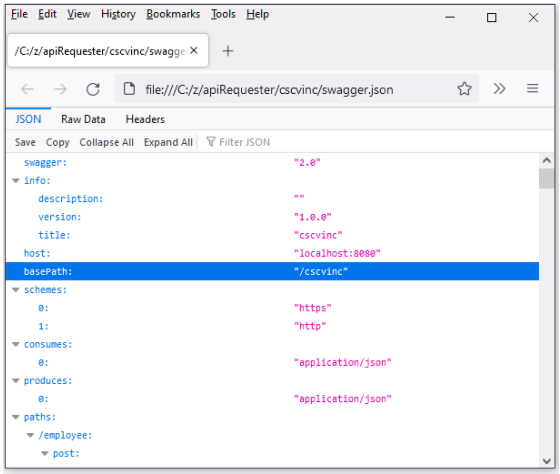
Configure the connection between z/OS Connect EE and the external API.

ibm.biz/zosconnect-configure-endpoint-connection

mitchj@us.ibm.com

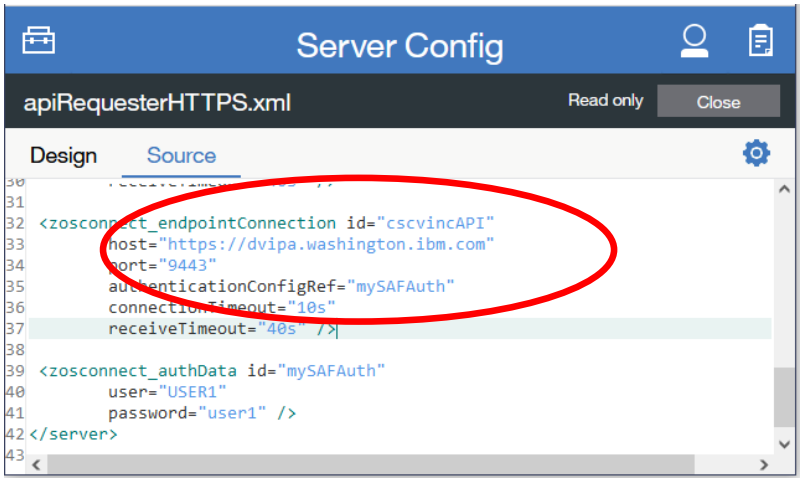
Steps to calling an external API

Update the server XML configuration for the endpoint

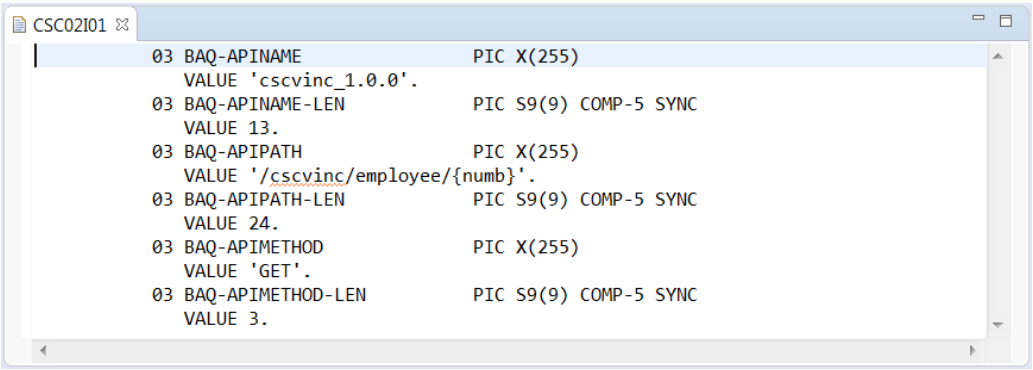


cscvinc.properties

```
connectionRef=cscvincAPI
```



http://dvipa.washington.ibm.com:9443/cscvincapi/employee/{numb}

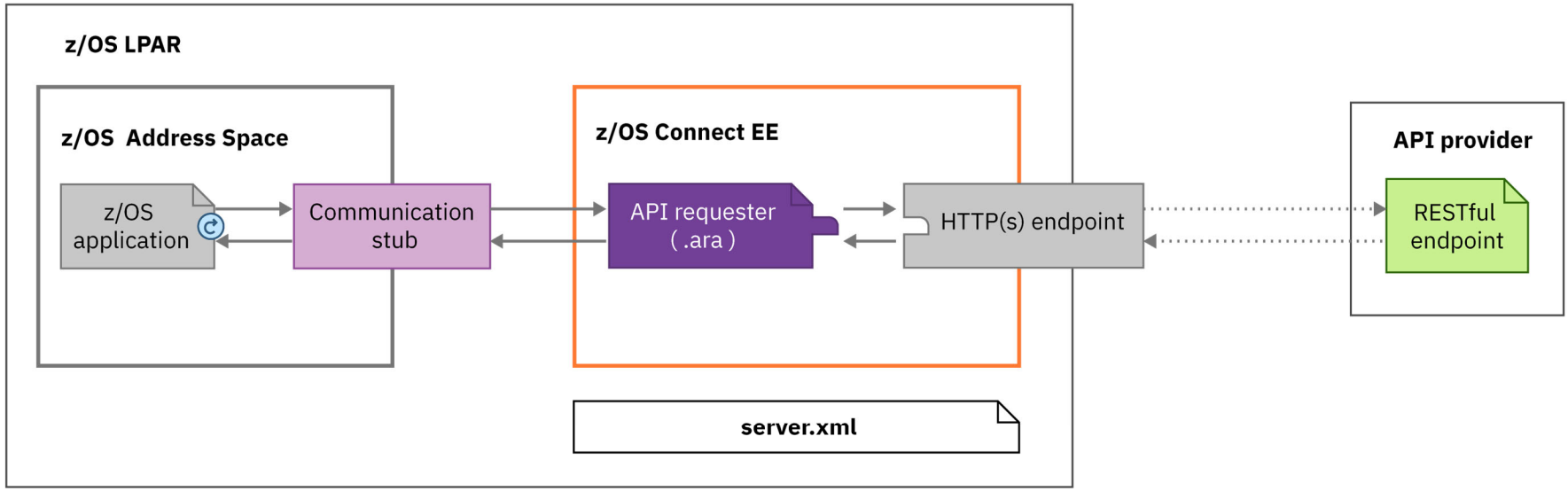


mitchj@us.ibm.com

Steps to calling an external API

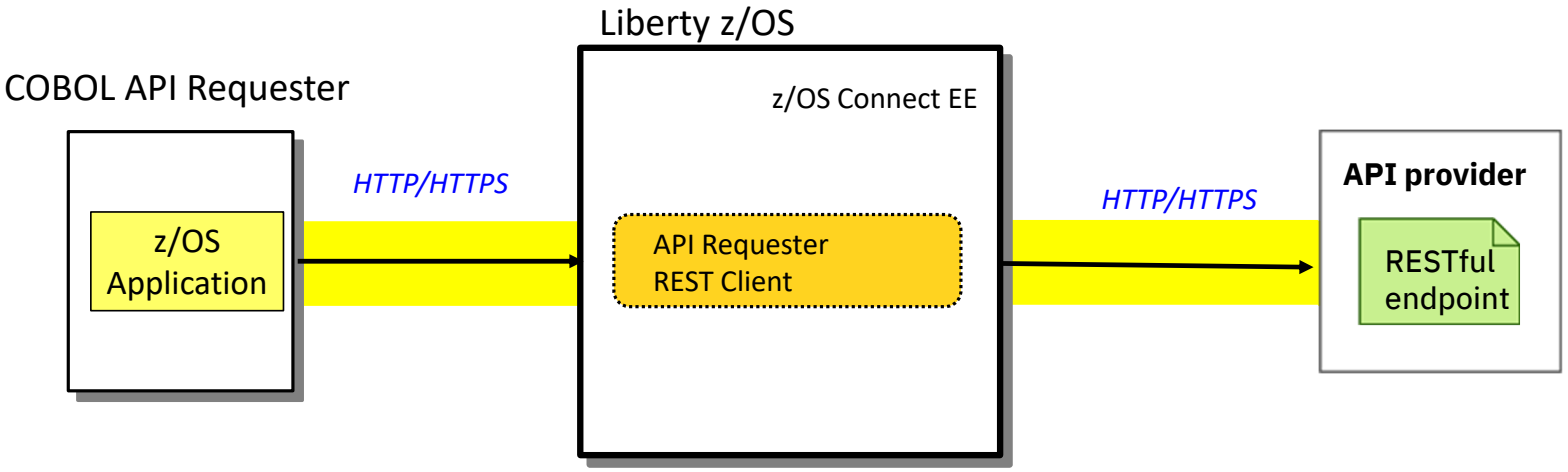
z/OS Connect EE

Done



mitchj@us.ibm.com

API requester to API Provider connection overview



MVS Batch and IMS HTTP connection details provided by:

- Environment Variables (BAQURI, BAQPORT)
  - Via JCL
  - LE Options (CEEROPTS)
  - Programmatically (CEEENV)
- HTTP or HTTPS

CICS HTTP connection details provided by:

- CICS URIMAP resource (default BAQURIMP)
  - HOST
  - PORT
  - SCHEME (HTTP/HTTPS)

mitchj@us.ibm.com

/miscellaneousTopics

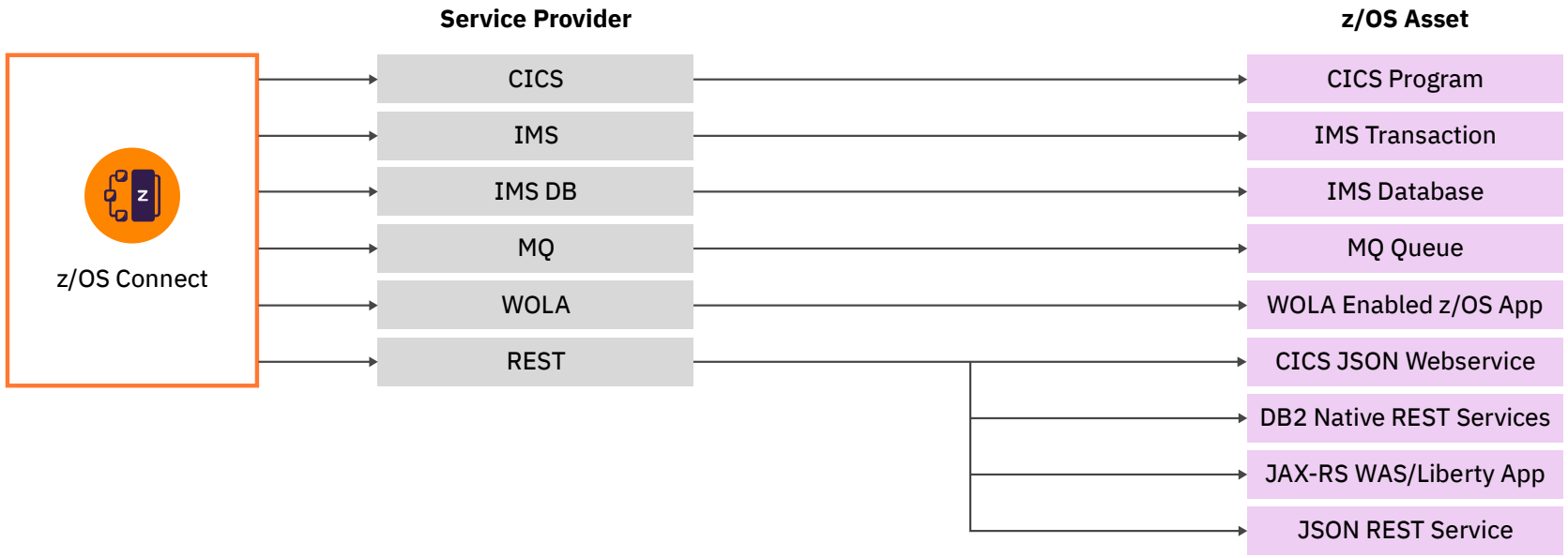
performance, high availability, Liberty

mitchj@us.ibm.com

What assets can z/OS Connect EE map to?

z/OS Connect EE

And which service provider could I use?



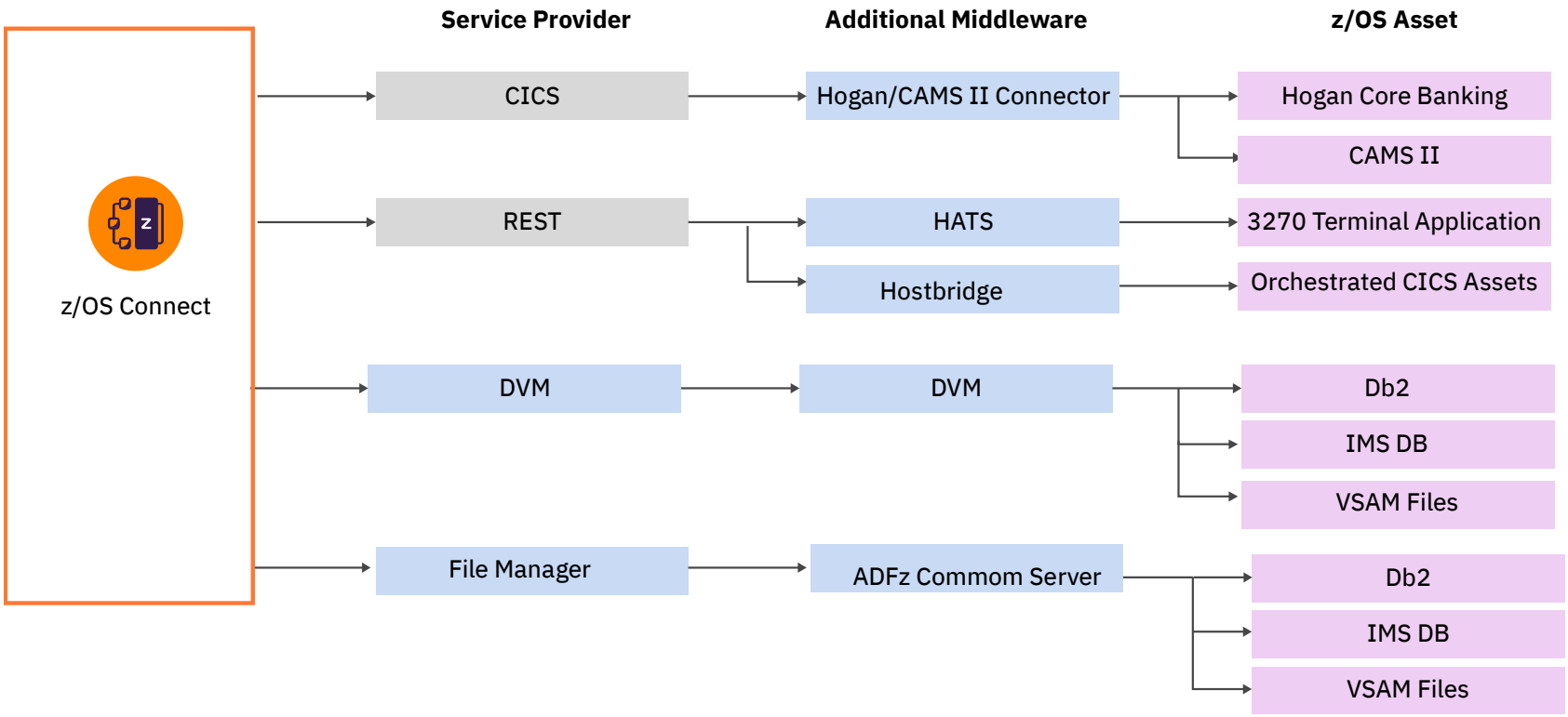
The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.

mitchj@us.ibm.com

Additional Middleware

z/OS Connect EE

Additional value from the ecosystem

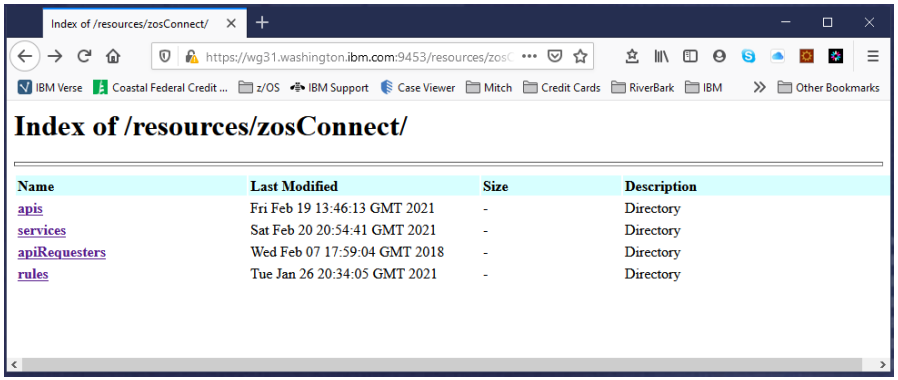


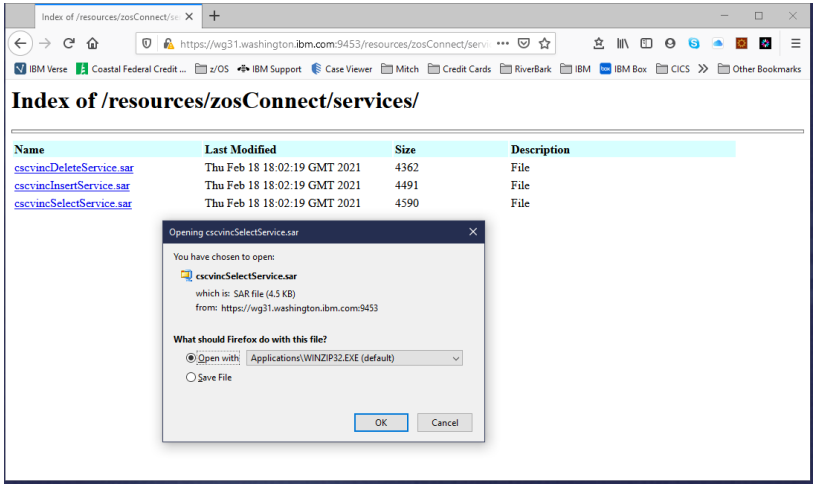
z/OS Connect EE is **pluggable** and **extensible** allowing the use of additional middleware to expand the list of z/OS assets you can expose as APIs

mitchj@us.ibm.com

Tech/Tip: Providing access to service archives files

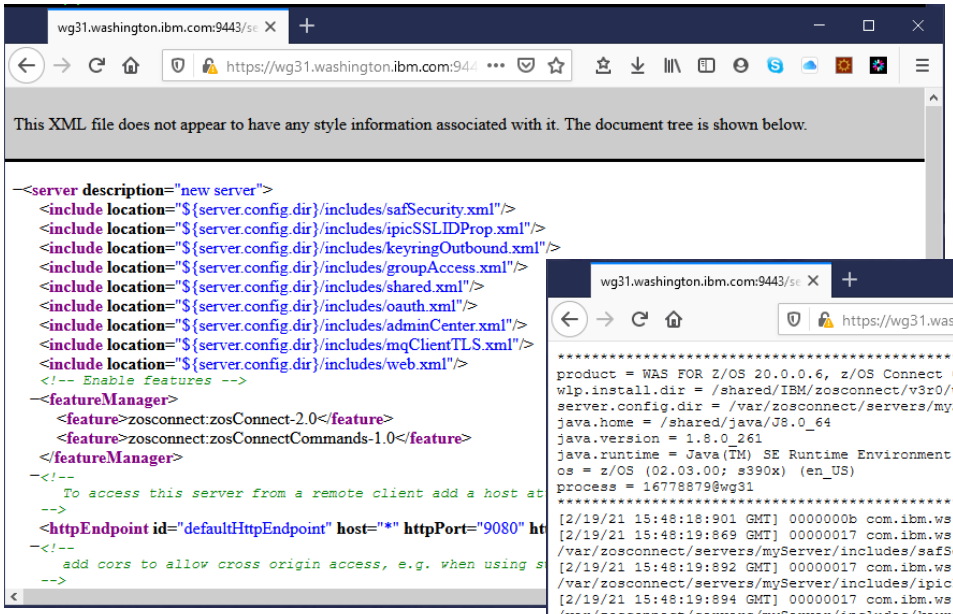
```
<webApplication id="resources-location" name="resources"
  location="${server.config.dir}/resources/zosconnect">
  <web-ext context-root="/resources/zosConnect"
    enable-file-serving="true" enable-directory-
browsing="true">
    <file-serving-attribute name="extendedDocumentRoot"
      value="${server.config.dir}/resources/zosconnect" />
  </web-ext>
</webApplication>
```



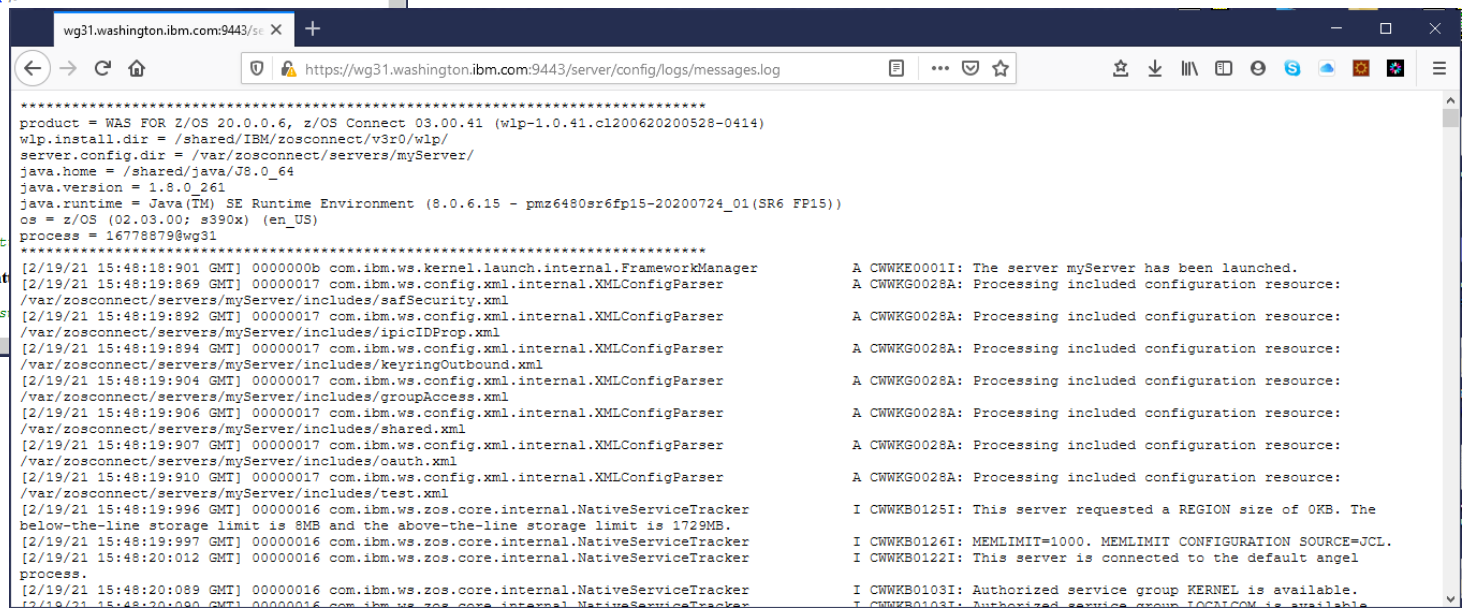


mitchj@us.ibm.com

Tech/Tip: Providing access to configuration/log information



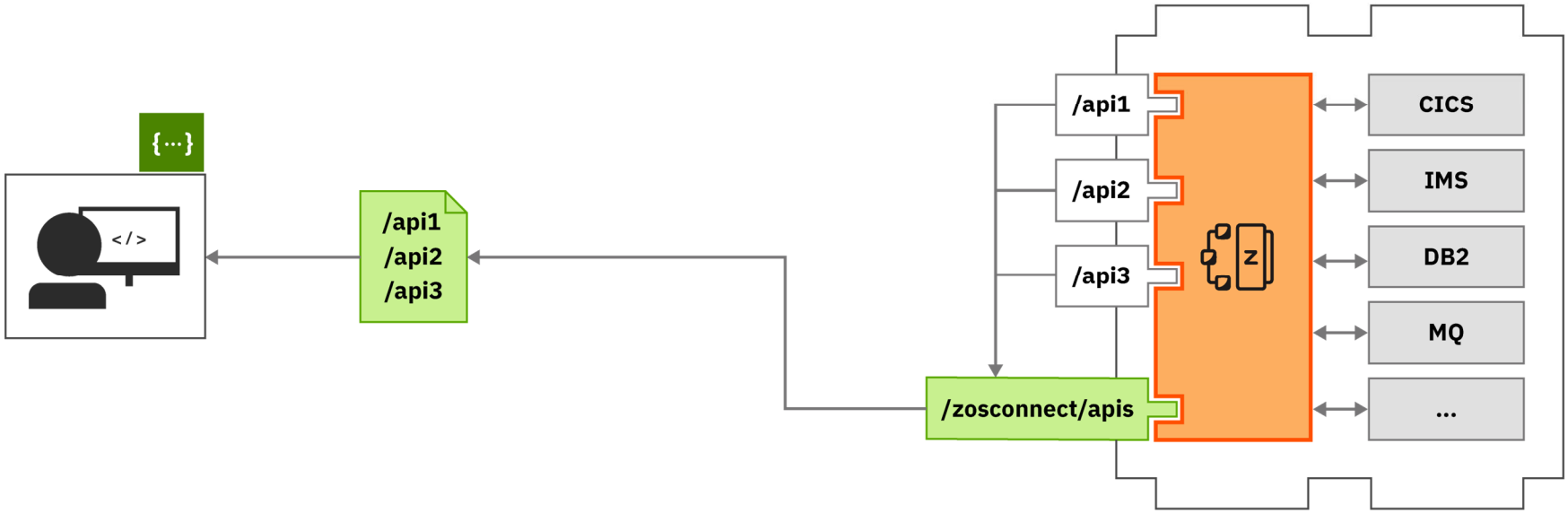

```
<webApplication id="serverConfig-location" name="serverConfig"
  location="${server.config.dir}">
  <web-ext context-root="/server/config"
    enable-file-serving="true" enable-directory-browsing="true">
    <file-serving-attribute name="extendedDocumentRoot"
      value="${server.config.dir}" />
  </web-ext>
</webApplication>
```



mitchj@us.ibm.com

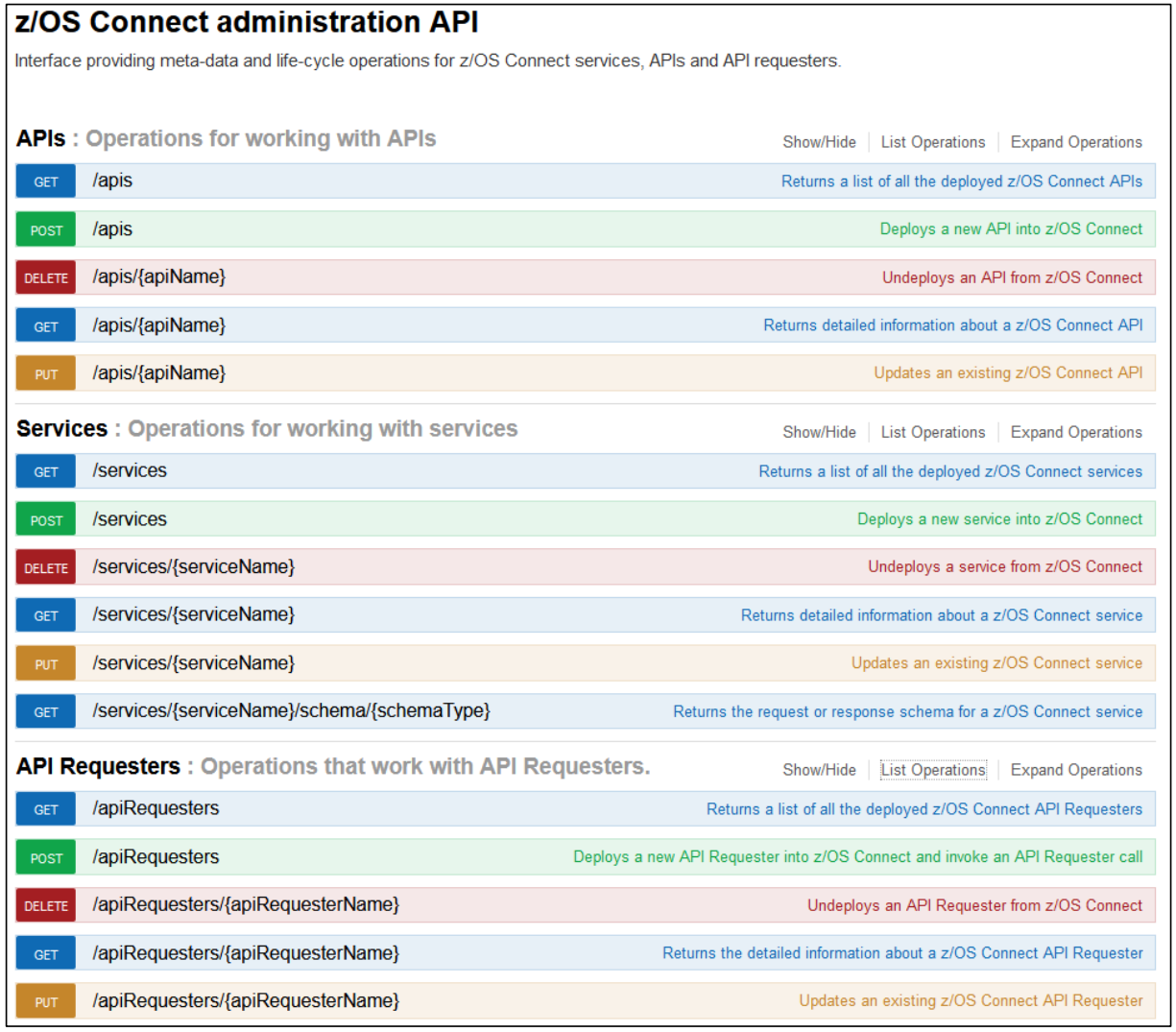
API Documentation

z/OS Connect EE



APIs are discoverable via Swagger docs served from **z/OS Connect EE**.

mitchj@us.ibm.com



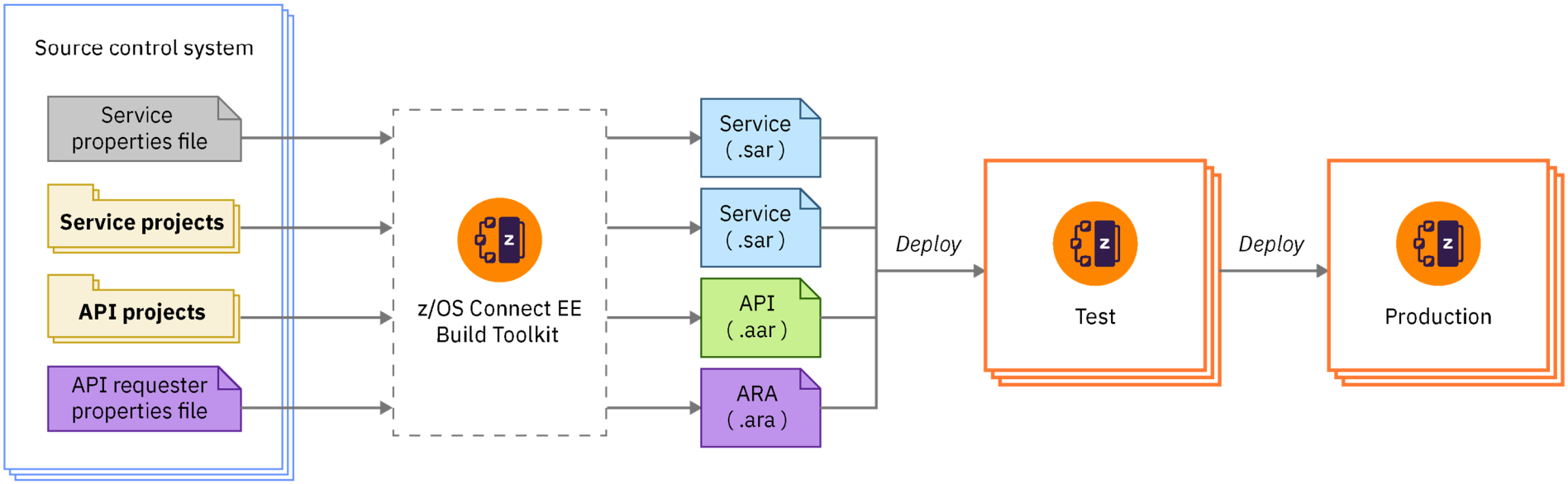
mitchj@us.ibm.com

DevOps using z/OS Connect EE

z/OS Connect EE

Automate the development and deployment of services, APIs, and API requesters for continuous integration and delivery.

- The build toolkit supports the generation of service archives and API archives from projects created in the z/OS Connect EE API toolkit
- The build toolkit also supports the use of properties files to generate API requester archives
- Run the build toolkit from a build script to generate these archive files
- Deploy them to z/OS Connect servers by copying them to their deployment directories or by using the REST Admin API



mitchj@us.ibm.com

ibm.biz/zosconnect-devops

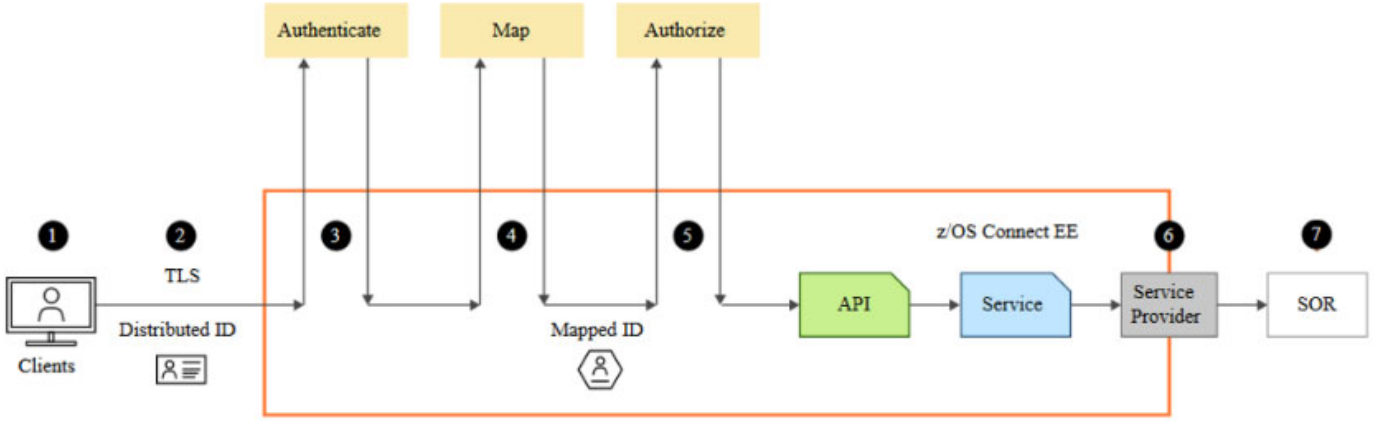
/security

How is security implement?

mitchj@us.ibm.com

Typical z/OS Connect EE API Provider security flow

z/OS Connect EE



1. The credentials provided by the client
2. Secure the connection to the z/OS Connect EE server
3. Authenticate the client. This can be within the z/OS Connect EE server or by requesting verification from a third-party server
4. Map the authenticated identity to a user ID in the user registry
5. Authorize the mapped user ID to connect to z/OS Connect EE and optionally authorize user to invoke actions on APIs
6. Secure the connection to the System of Record (SoR) and provide security credentials to be used to invoke the program or to access the data resource
7. The program or database request may run in the SoR under the mapped ID

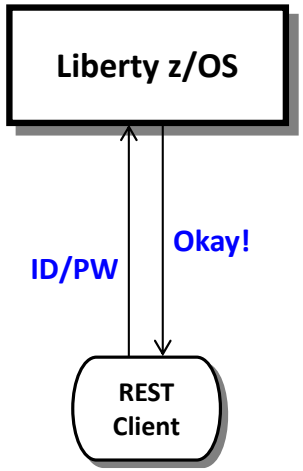
mitchj@us.ibm.com

API Provider Authentication

z/OS Connect EE

Several different ways this can be accomplished:

Basic Authentication



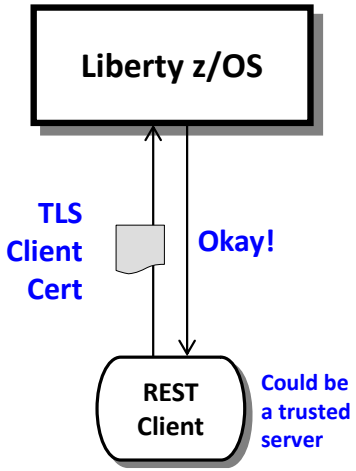
Server prompts for ID/PW

Client supplies ID/PW or ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

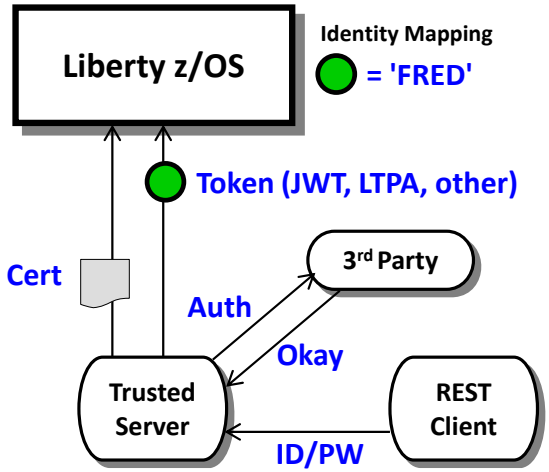
Client supplies certificate

Server validates cert and maps to an identity

Registry options:

- LDAP
- SAF

Third Party Authentication



Client authenticates to 3rd party sever

Client receives a trusted 3rd party token

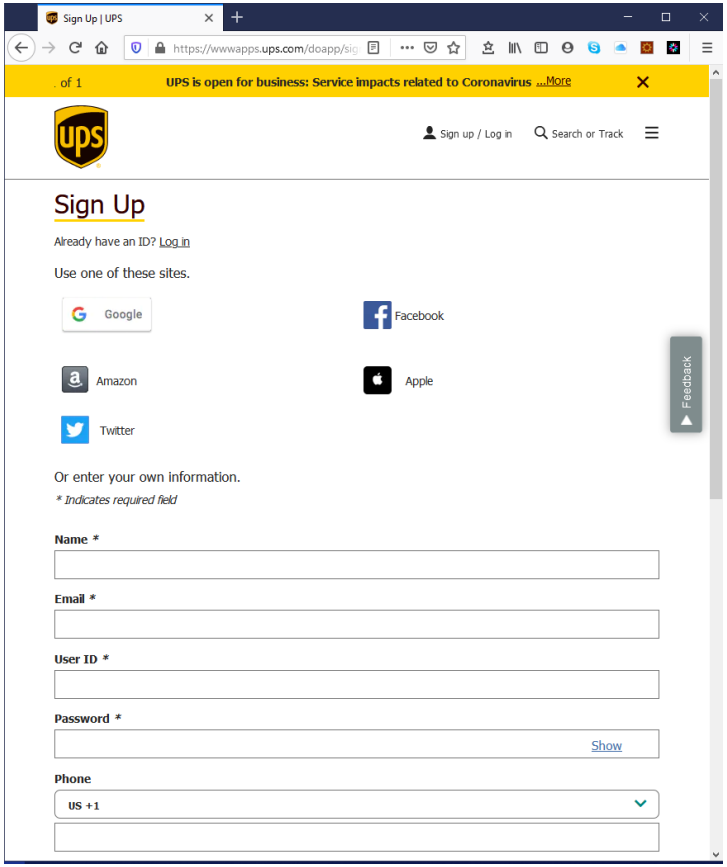
Token flows to Liberty z/OS and is mapped to an identity

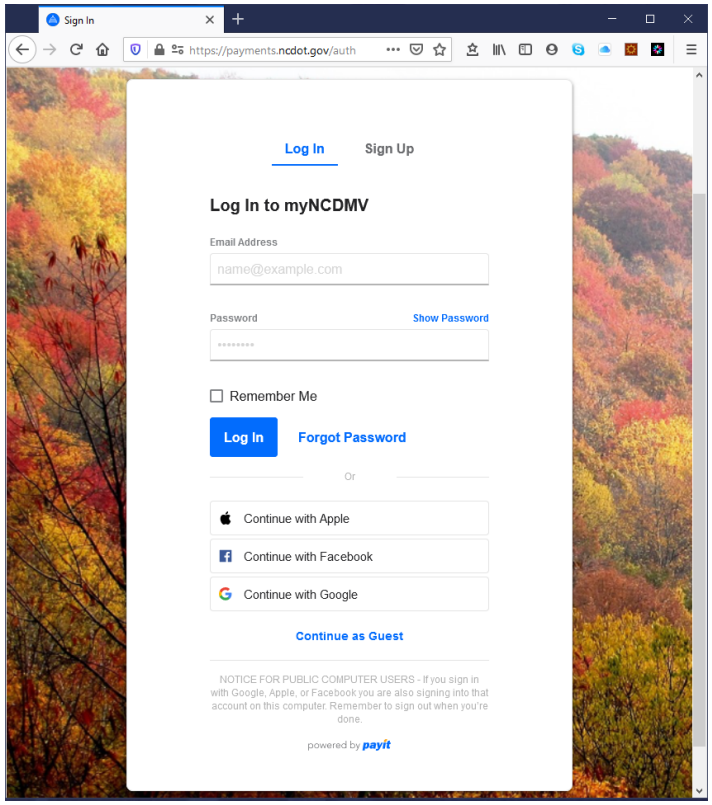
Registry options:

- LDAP
- SAF

mitchj@us.ibm.com

Third Party Authentication Examples

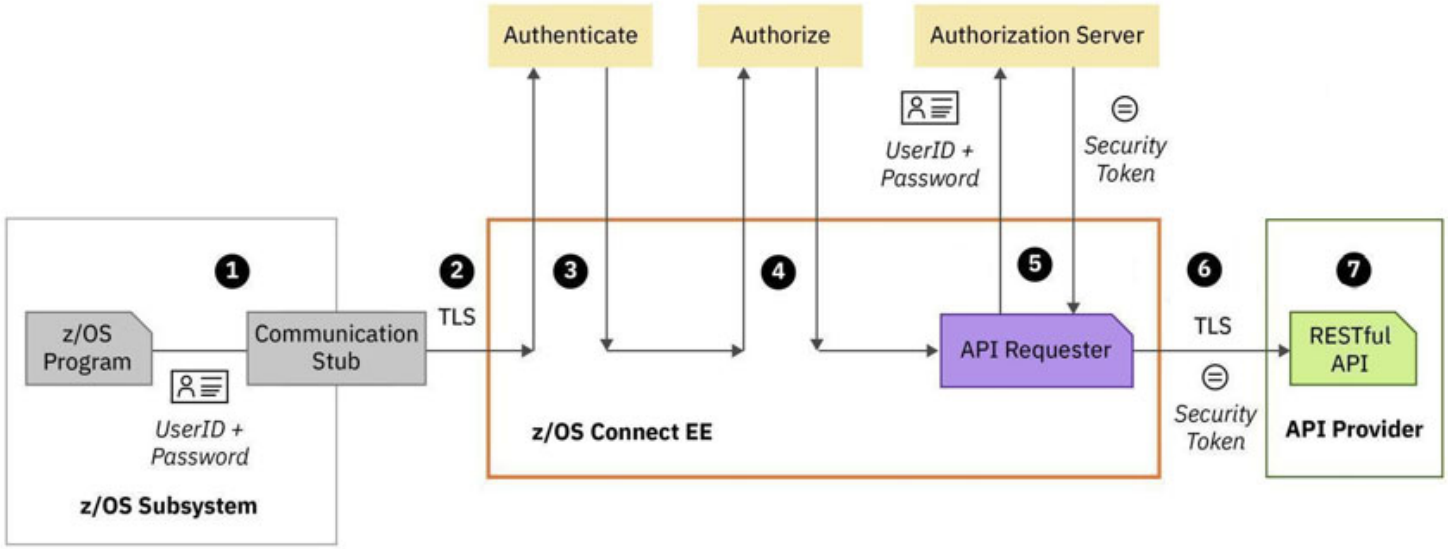




mitchj@us.ibm.com

Typical z/OS Connect EE API Requester security flow

z/OS Connect EE



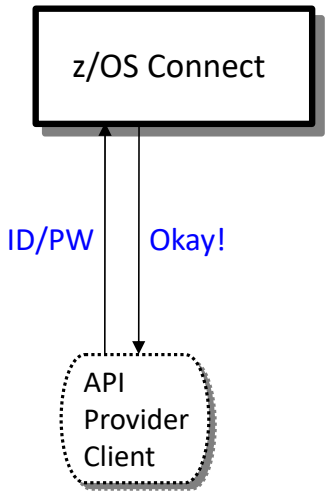
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS Connect EE and to perform specific actions on z/OS Connect EE API requesters
5. Pass the user ID and password credentials to an authorization server to obtain a security token.
6. Secure the connection to the external API provider, and provide security credentials such as a security token to be used to invoke the RESTful API
7. The RESTful API runs in the external API provider

mitchj@us.ibm.com

z/OS Application to z/OS Connect API Requester

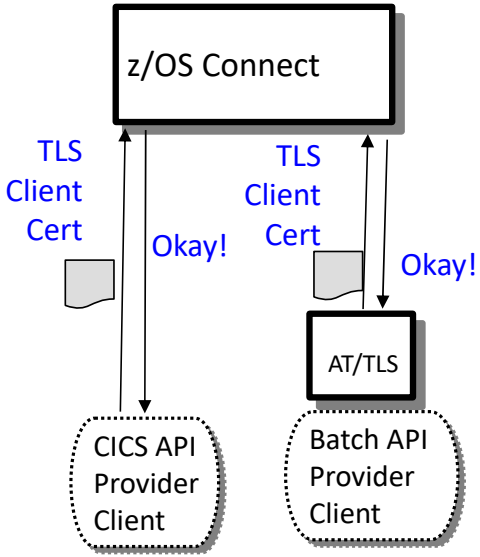
Two options for providing credentials for authentication

Basic Authentication



Application provides ID/PW or ID/PassTicket

Client Certificate



z/OS Connect requests a client certificate

CICS or AT/TLS supplies a client certificate

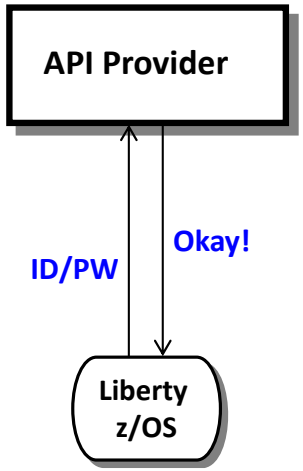
mitchj@us.ibm.com
mitchj@us.ibm.com

API Requester - API Provider Authentication

z/OS Connect EE

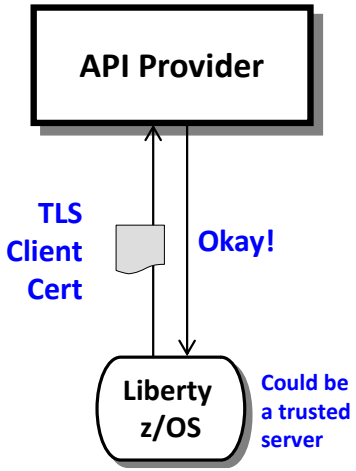
Several different ways this can be accomplished:

Basic Authentication



zCEE supplies ID/PW or ID/Passticket

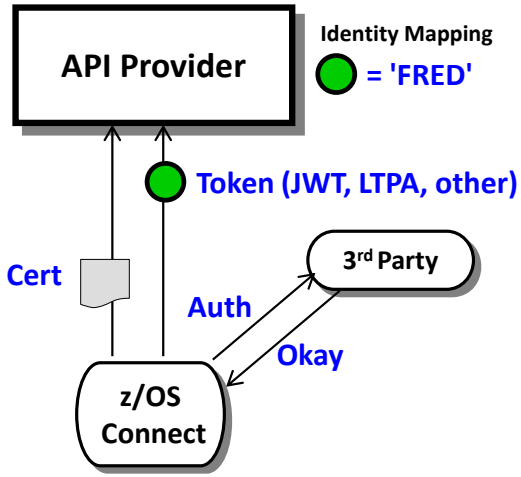
Client Certificate



Server prompts for certificate

zCEE supplies certificate

Third Party Authentication



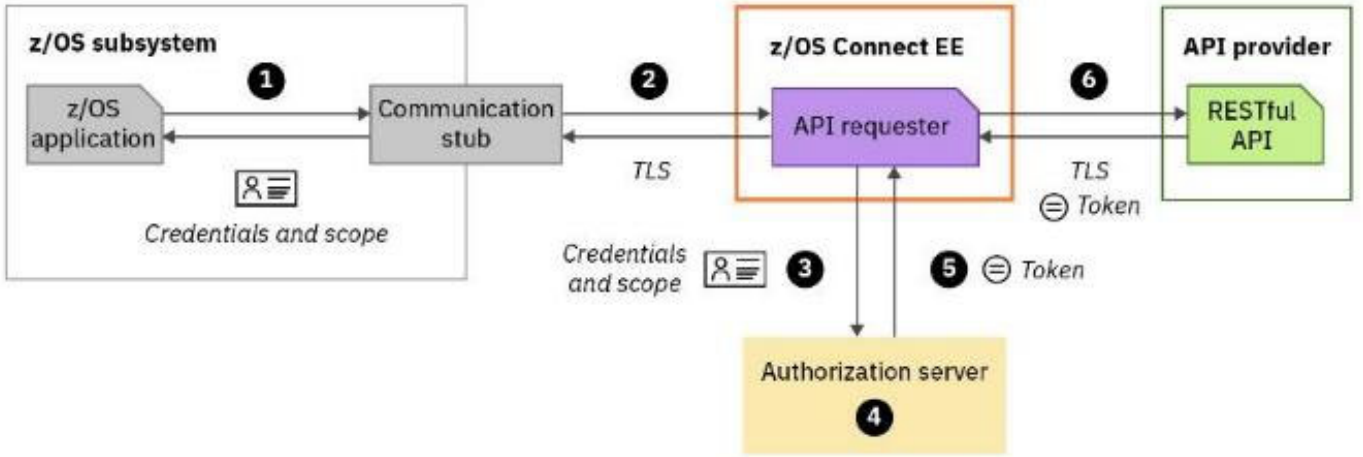
zCEE authenticates to 3rd party sever

zCEE receives a trusted 3rd party token

Token flows to API Provider

mitchj@us.ibm.com

Calling an API with OAuth 2.0 support



mitchj@us.ibm.com

Configuring OAuth support – BAQRINFO copy book

```
05 BAQ-OAUTH.
  07 BAQ-OAUTH-USERNAME            PIC X(256).
  07 BAQ-OAUTH-USERNAME-LEN        PIC S9(9) COMP-5 SYNC VALUE 0.
  07 BAQ-OAUTH-PASSWORD            PIC X(256).
  07 BAQ-OAUTH-PASSWORD-LEN        PIC S9(9) COMP-5 SYNC VALUE 0.
  07 BAQ-OAUTH-CLIENTID            PIC X(256).
  07 BAQ-OAUTH-CLIENTID-LEN        PIC S9(9) COMP-5 SYNC VALUE 0.
  07 BAQ-OAUTH-CLIENT-SECRET       PIC X(256).
  07 BAQ-OAUTH-CLIENT-SECRET-LEN   PIC S9(9) COMP-5 SYNC VALUE 0.
  07 BAQ-OAUTH-SCOPE-PTR           USAGE POINTER.
  07 BAQ-OAUTH-SCOPE-LEN           PIC S9(9) COMP-5 SYNC.
```

Grant Type: *client_credentials* - the identity associated with the combination of the CICS, IMS, or z/OS application, and the z/OS Connect EE server that calls the RESTful API on behalf of the CICS, IMS, or z/OS application

Grant Type: *password* - The identity of the user provided by the CICS, IMS, or z/OS application, or it might be another entity. Client_credentials can be supplied by the program or in the server XML configuration.

Scope is always required.

OAuth 2.0 specification entity	password	client_credentials	Where Set
Client ID	required	Required	server.xml or by application
Client Secret	optional	Required	server.xml or by application
Username	required	N/A	by application
Password	required	N/A	by application

mitchj@us.ibm.com

Agenda

- An Introduction and Overview of using REST API
- Enabling RESTful API to various z/OS resources, e.g.
  - CICS
  - Db2
  - IMS/TM
  - IMS/DB
  - MQ
  - MVS Batch
  - Outbound REST APIs
  - IBM DVM
  - Host Access Transformation Services (3270 screen-based applications)
  - IBM File Manager
- Accessing RESTful API from z/OS COBOL Applications
- A brief overview of z/OS Connect Security*

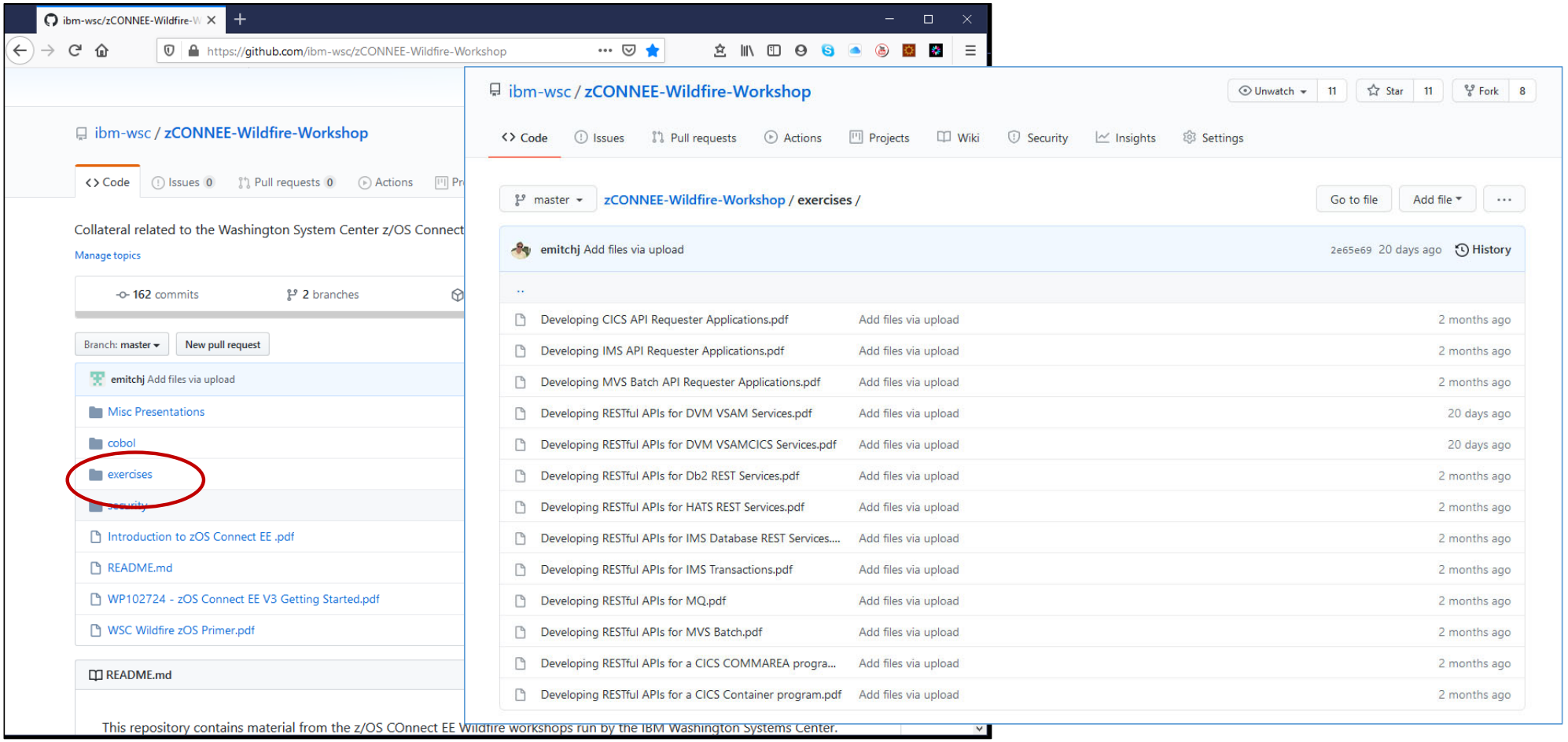
*For more on security, contact your local IBM rep regarding the schedule of workshop *zOSSEC1 IBM z/OS Connect Administration/Security Wildfire Workshop*

mitchj@us.ibm.com

z/OS Connect Wildfire Github Site

https://ibm.biz/Bdf8BZ

z/OS Connect EE



mitchj@us.ibm.com

- **Contact your IBM representative to schedule access to these exercises**

/questions?thanks=true

Thank you for listening.

- z/OS Connect EE Users Group: **https://www.linkedin.com/groups/8731382/**

mitchj@us.ibm.com