ZCINTRO - IBM z/OS Connect
An introduction and overview

Mitch Johnson
mitchj@us.ibm.com
John Brefach
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
  - Outbound REST APIs

- Accessing RESTful API from z/OS COBOL Applications
- A brief introduction to z/OS Connect Security

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

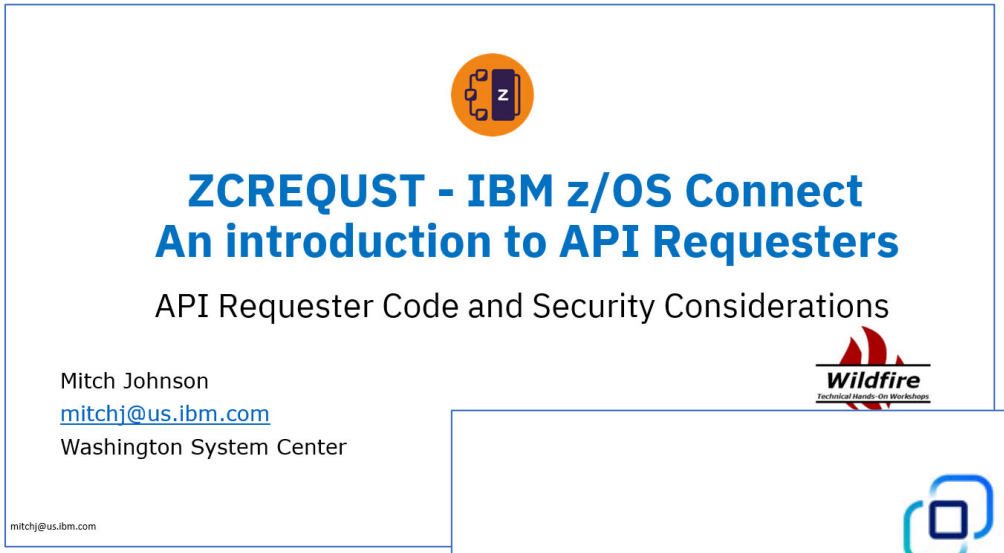
mitchj@us.ibm.com

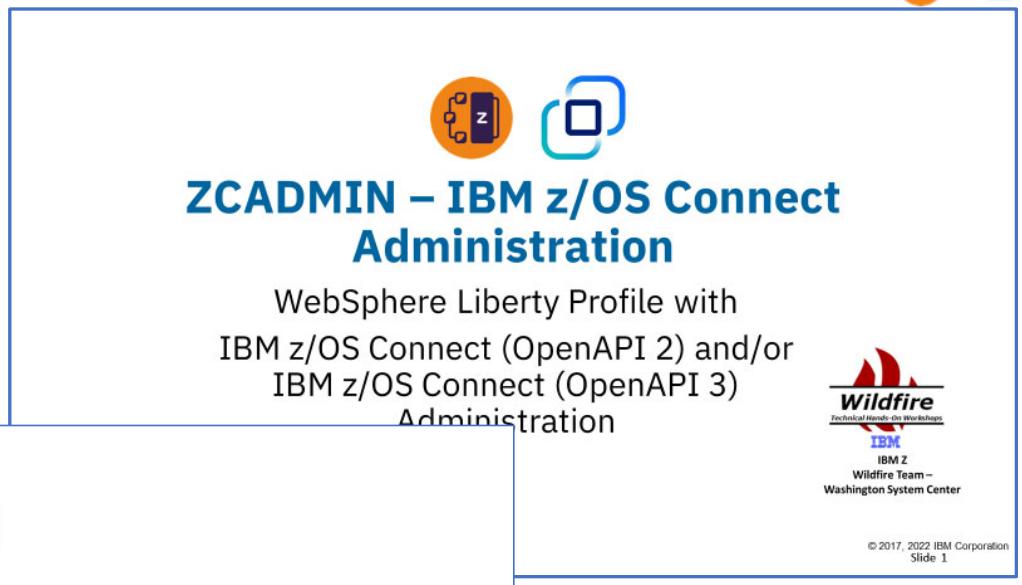
Notes and Disclaimers

- The information in this presentation was derived from various product documentation web sites.
- Additional information included in this presentation was distilled from years of experience implementing security using RACF with z/OS products like CICS, IMS, Db2, MQ, etc. as well as Java runtimes environments like WebSphere Application Server and WebSphere Application Server Liberty which is commonly called Liberty.
- There will be additional information on slides that will designated as Tech/Tips. These contain information that at perhaps at least interesting and hopefully, useful to the reader.
- **IBM z/OS Connect (OpenAPI 2)** refers the z/OS Connect EE product prior to service level V3.0.55. **IBM z/OS Connect (OpenAPI 3)** refers to the additional functions and features added with service level V3.0.55. Important - servers configured for OpenAPI 2 can will continue to operate as is with service level V3.0.55 and later.
- A z/OS Connect OpenAPI 2, or a z/OS Connect OpenAPI 3 icon will appear on slides where the information is specific to these products. Don't hesitate to ask questions as to why the icon does or does not appear on certain slides.
- The examples, tips, etc. present in this material are based on firsthand experiences and are not necessarily sanctioned by Liberty or z/OS Connect development.

mitchj@us.ibm.com

This session is part of a series of z/OS Connect workshops. . .

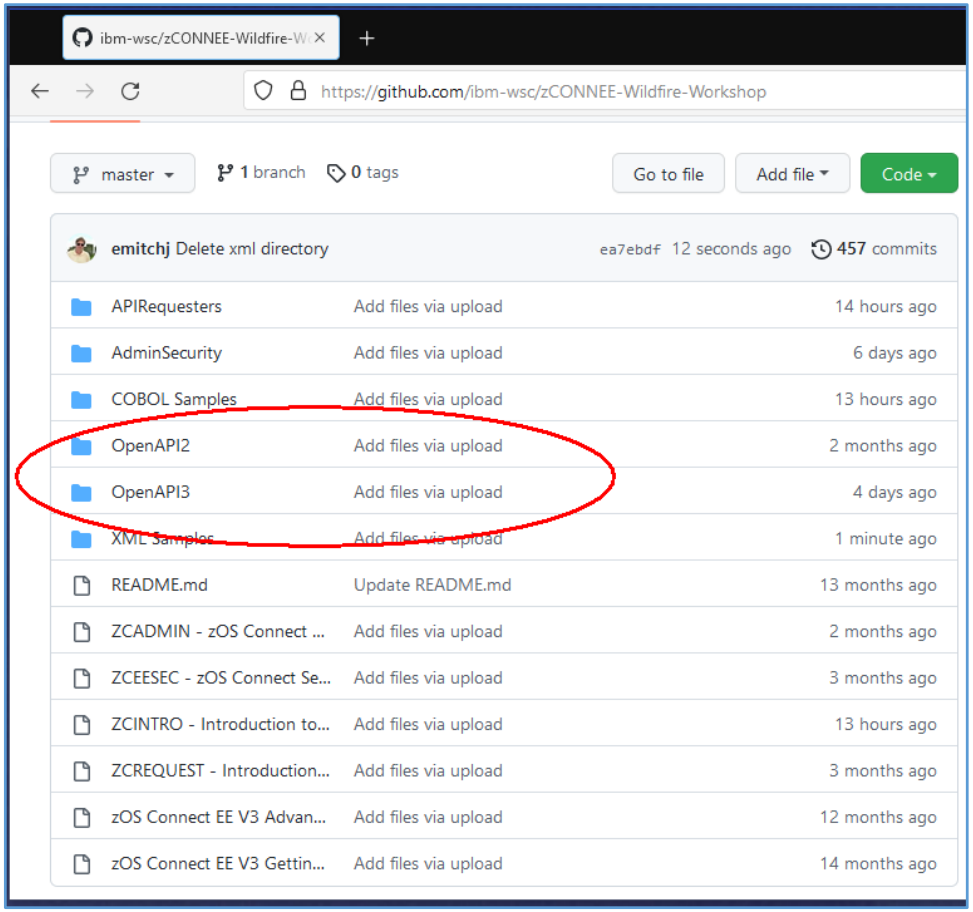


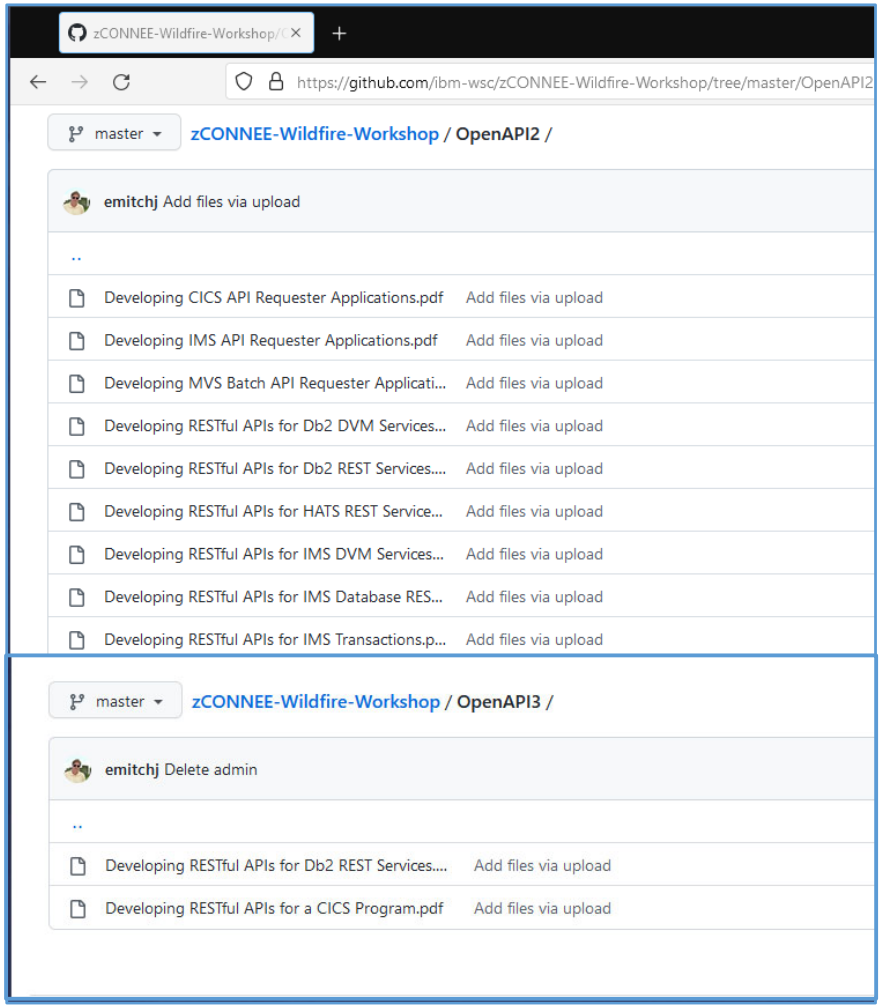


https://www.ibm.com/support/pages/mainframe-system-education-wildfire-workshops

z/OS Connect Wildfire Github Site

https://ibm.biz/zCEEWorkshopMaterial





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

mitchj@us.ibm.com

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

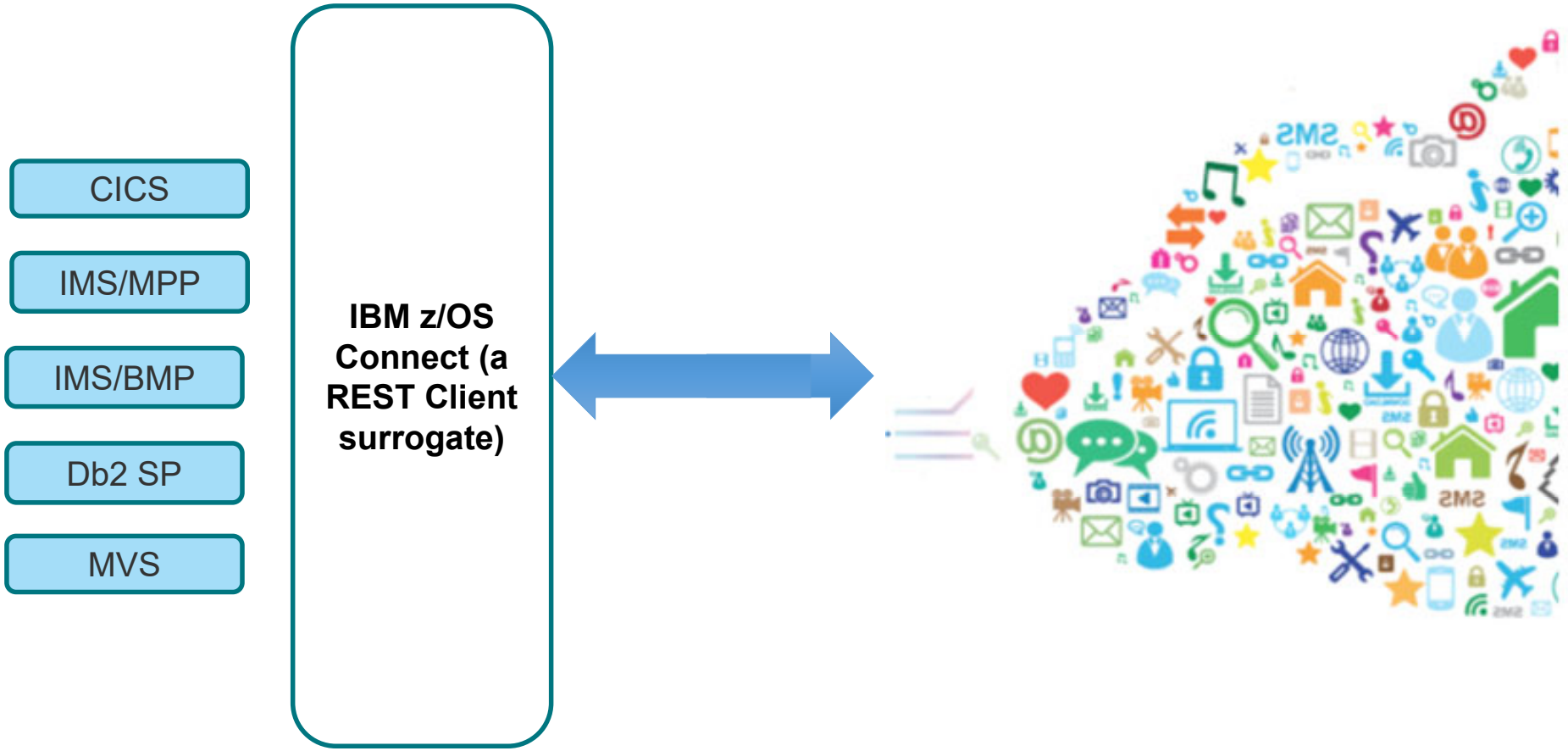
IBM z/OS Connect (an API Gateway)

- 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

z/OS Connect EE exposes external REST APIs in the “cloud” to z/OS applications



mitchj@us.ibm.com

Before we go any further, let ask

/what_is_REST?

And what makes an API “RESTful”?

mitchj@us.ibm.com

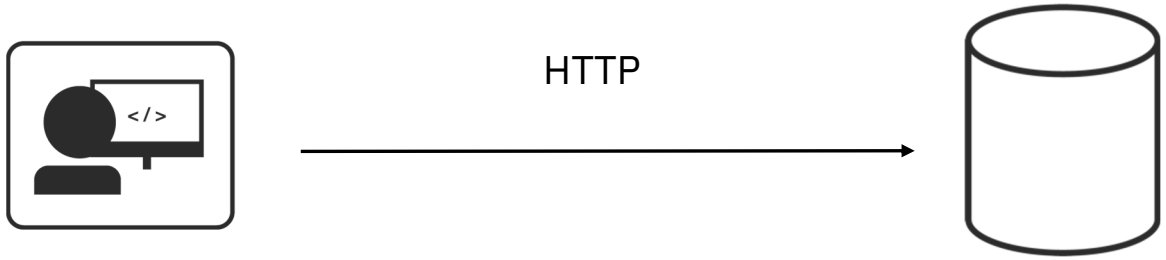
REST is architectural programming style

REST is acronym standing 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 the REST API

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

```
POST
GET     http://<host>:<port>/path/parameter?name=value&name=value
PUT
DELETE
```

Use Path and Query parameters to refine the request

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..

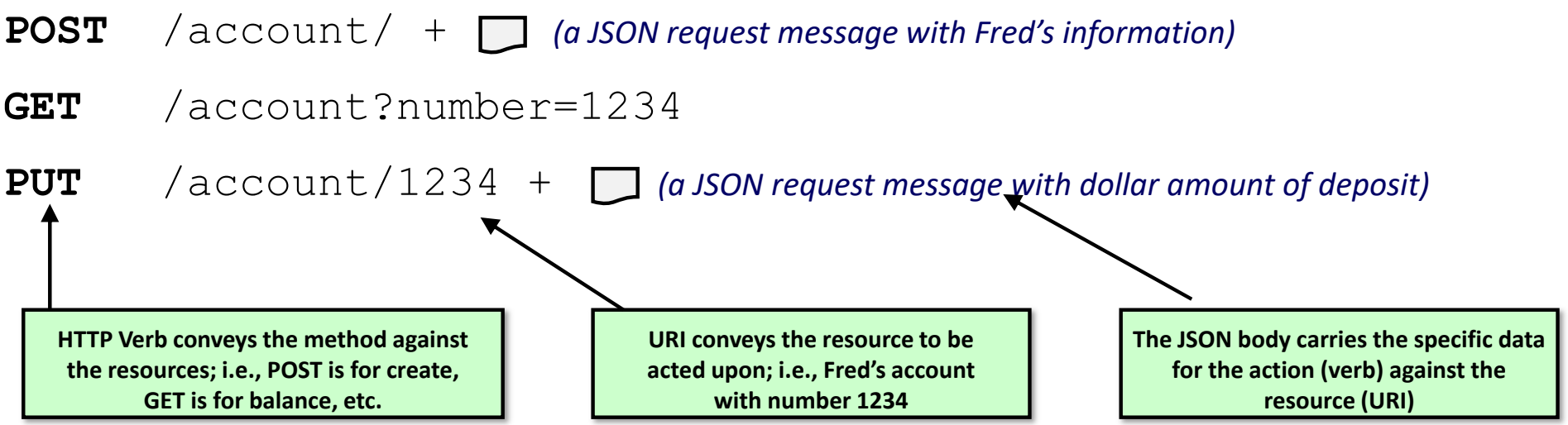
- It should be stateless (transaction management should be managed by the client)
- It should access and/or identify all server resources using only a single URI
- For performing CRUD operations, it should use HTTP verbs such as get, post, put and delete
- It should return the result only in the form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



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

(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

The goal is to strive to design API to use RESTful properties

1. Use the same URIs for the same resource with the appropriate method for operations

```
GET  http://www.acme.com/customers/12345
PUT  http://www.acme.com/customers/12345?address=10%20New%20Street
```

2. Use the same JSON property names between request and response messages

```
POST http://www.acme.com/customers/12345
BODY { "name": "Joe Bloggs",
       "address": "10 Old Street",
       "phoneNo": "01234 0123456" }
```



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

3. Use JSON name/value pairs

```
PUT http://www.acme.com/customers/12345
BODY { "address"    : "10 New Street"}
```



```
RESPONSE HTTP 200 OK
```

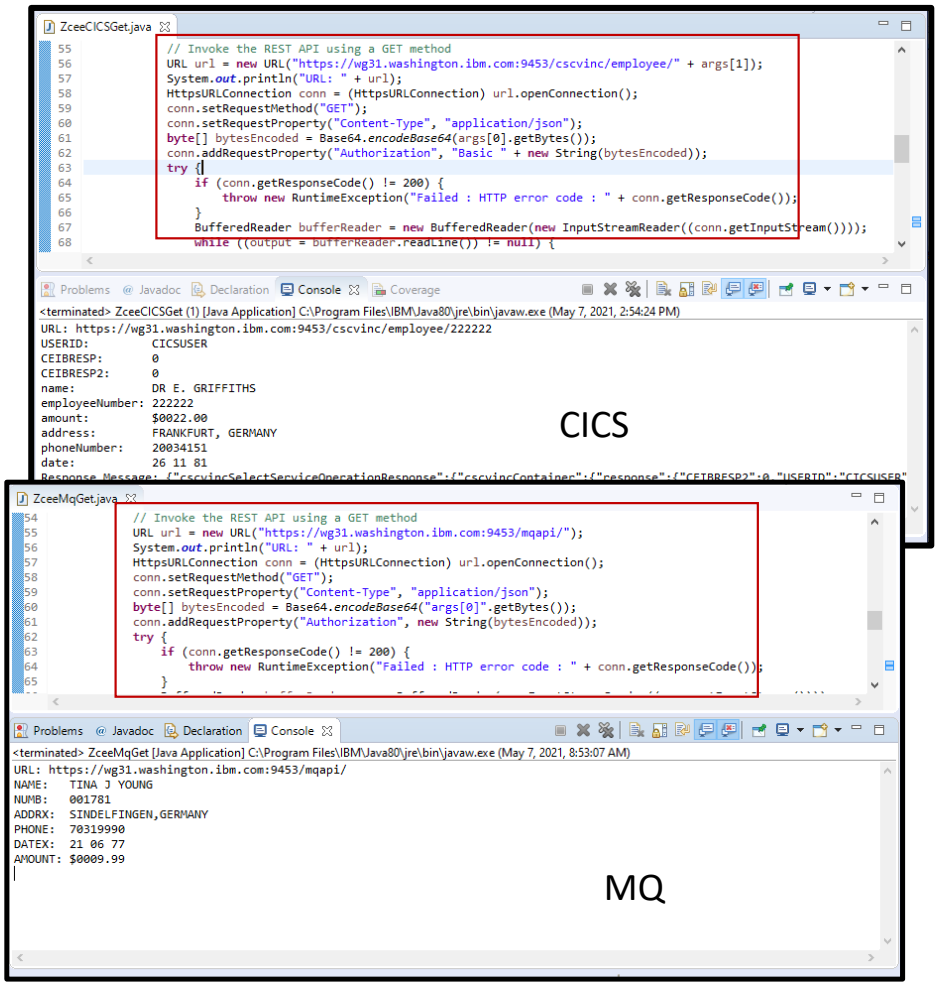
mitchj@us.ibm.com

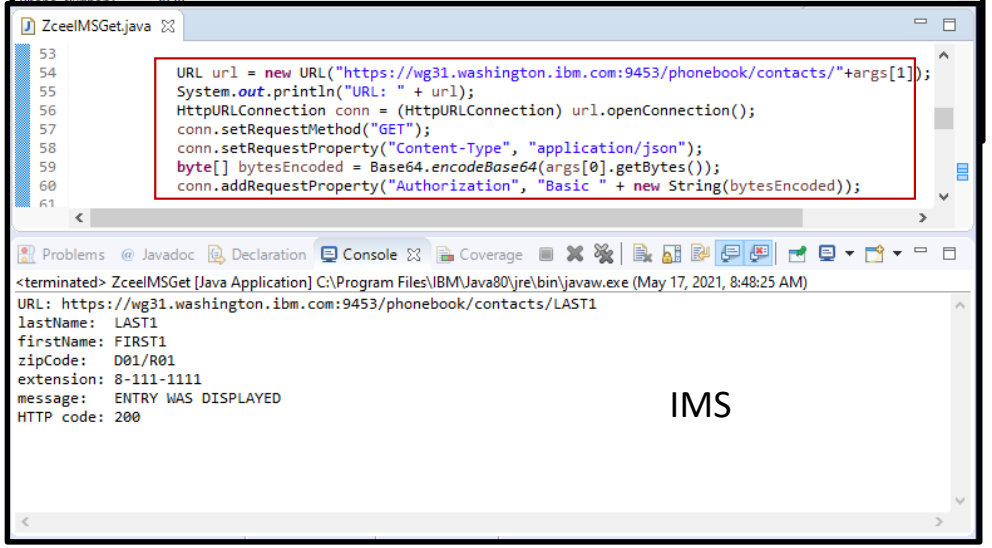
Why is REST popular?

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

Goal: Client code is unaware of the z/OS infrastructure





mitchj@us.ibm.com

How do we describe and/or document an API?

mitchj@us.ibm.com

/oai/open_api_initiative

The industry standard framework for describing REST APIs

The OpenAPI Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how APIs are described. As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral description format. The OpenAPI Specification was originally based on the Swagger Specification, donated by SmartBear Software.

mitchj@us.ibm.com

We use an Open API specification

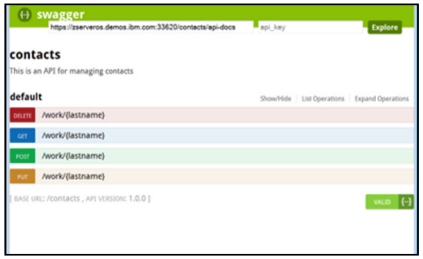
- It is more than just an API framework

OPEN API INITIATIVE

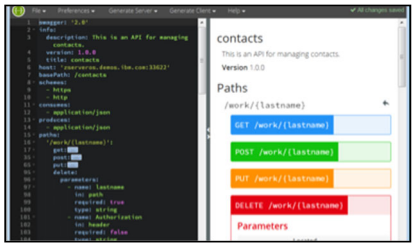
There are a number of tools available to aid consumption:

Code Generation* - create stub code to consume APIs from various languages

Test UIs - allows API consumers to easily browse and try APIs based on an OpenAPI document.



Editors - allows API developers to design their OpenAPI documents.



* z/OS Connect API Requester

+z/OS Connect, MQ REST support, Zowe

Important - You may have used or heard of the term Swagger with the use of APIs. As the use of APIs has grown this term has become in some respects misleading. To be more precise, OpenAPI refers to the API specifications (OpenAPI 2 and OpenAPI3) where Swagger refers to the tooling used to implement the specifcations.

mitchj@us.ibm.com

Let's stop and ask what is the significance of the OpenAPI Specification to z/OS Connect?

The industry standard framework for describing REST APIs

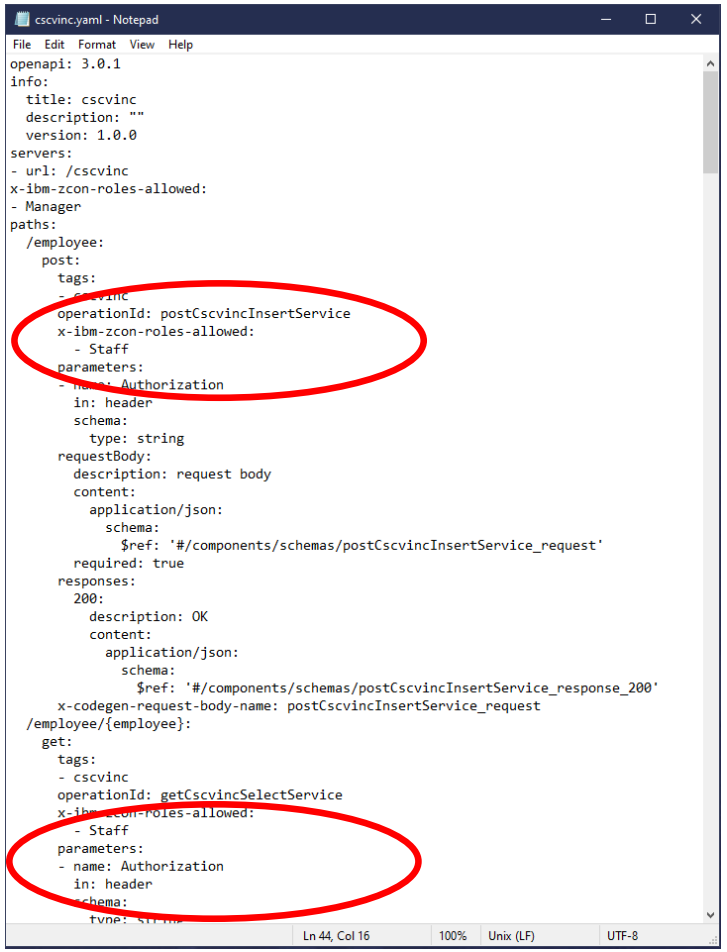
- **z/OS Connect and Swagger 2.0 (Open API Specification 2), supported initially by z/OS Connect**
  Initially, accessing z/OS resources was the only desire for developing APIs .The interactions with the z/OS resources was driven by the layout of the CICS COMMAREA or CONTAINER, the IMS or MQ messages or the Db2 REST service.
  - The details of the interactions with the z/OS resource determined the contents of the API request and response messages and the subsequent specification document.
  - **z/OS Connect produces the specification document that describes the methods and request and response messages.**

- **z/OS Connect and Open API Specification 3, supported by z/OS Connect starting in March 2022 service, V3.0.55**
  As companies mature their API strategy, they begin to introduce API governance boards to drive consistency in their API design. As more public APIs are created, government and industry standards bodies begin to regulate and drive for standardization. This drives the need for "API first" functional mapping capabilities within the integration platform. The external API design determined the layouts of the API request and response messages provided by the specification documents which was consumes by z/OS Connect to describe the z/OS resource interactions.
  - The API details of the methods and layouts of request and response messages are provided in advance and access to the z/OS resource is driven by the API design
  - **z/OS Connect consumes the specification document that describes the methods and request and response messages**

mitchj@us.ibm.com

Contrast the z/OS Connect OpenAPI 2 /OpenAPI 3 differences

z/OS Connect OpenAPI2 tooling **produces** an OpenAPI 2 specification document, where the details of the methods and request/response messages in the API specification are driven by the nature of the z/OS asset (JSON format).



z/OS Connect OpenAPI3 tooling **consumes** an OpenAPI3 specification document and the details of the methods and request/response messages are driven by the API specification (YAML format*) and not the nature of the z/OS asset. Also, JSONata can be used to augment the API response

mitchj@us.ibm.com

https://swagger.io/blog/api-strategy/difference-between-swagger-and-openapi/

*Yet Another Markup Language

Why /zos_connect?

Truly RESTful APIs to and from your mainframe.

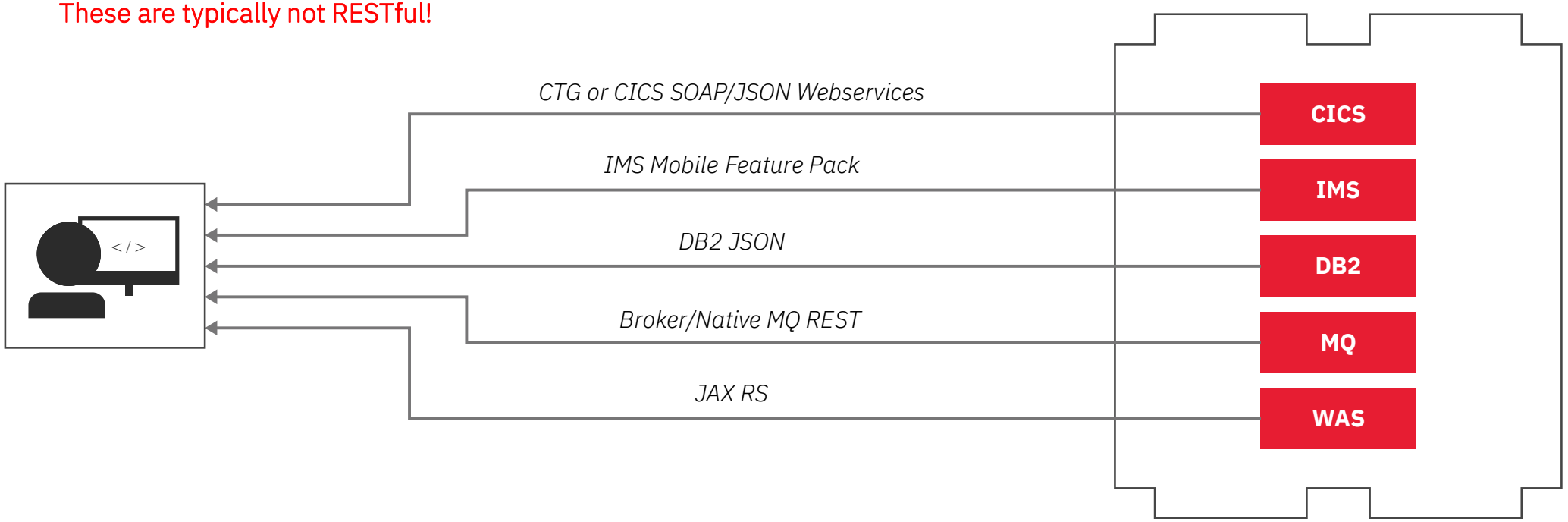
mitchj@us.ibm.com

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!



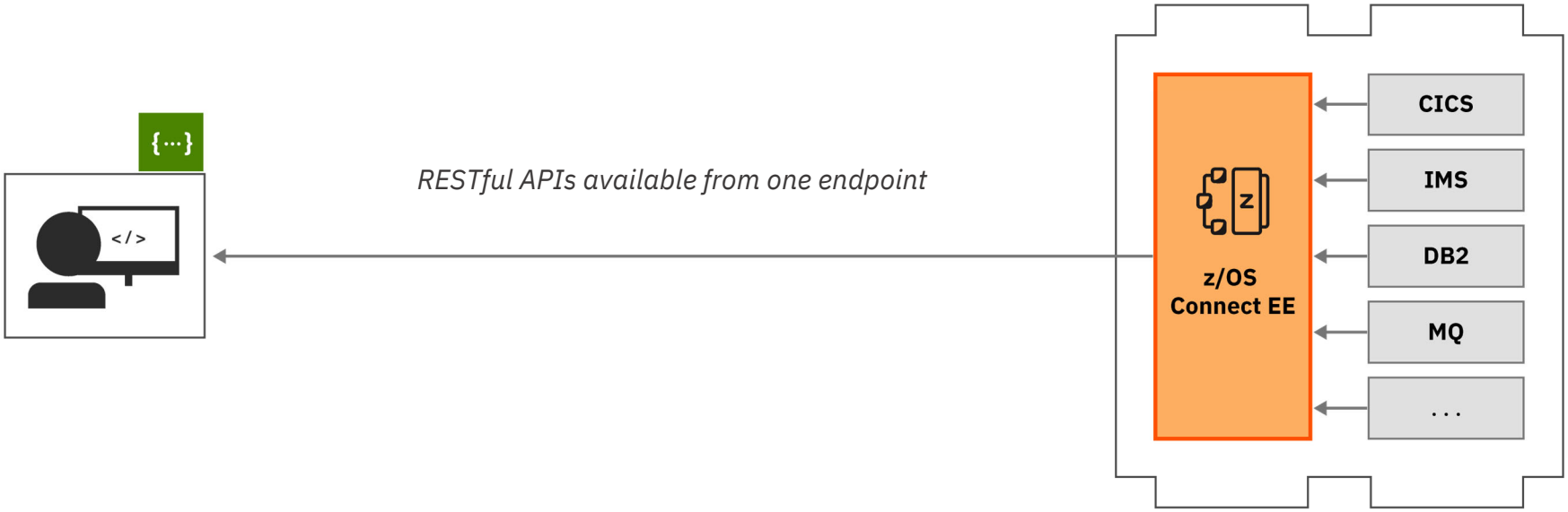
mitchj@us.ibm.com

z/OS Connect provides a single-entry point

- And exposes 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.



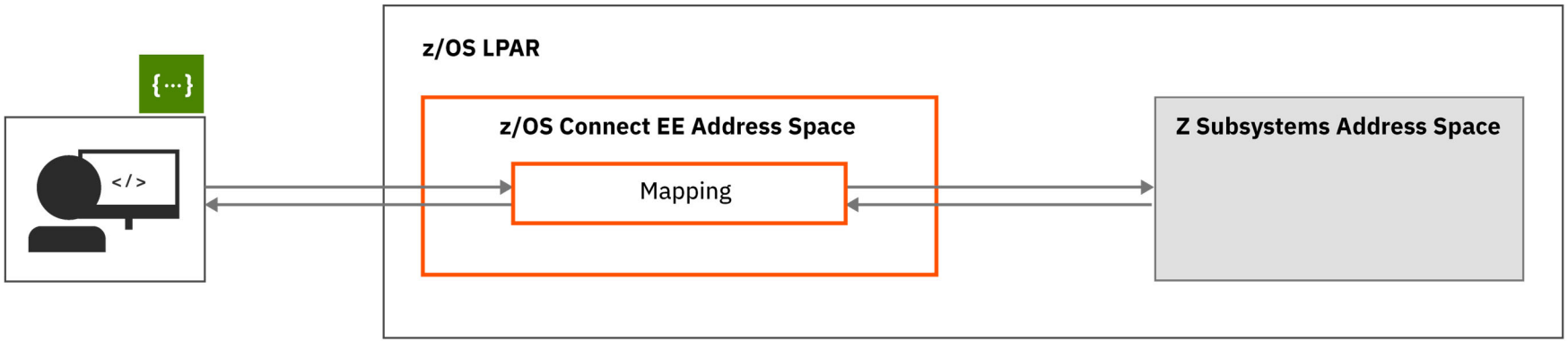
mitchj@us.ibm.com

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

mitchj@us.ibm.com

Data mapping/transformation

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

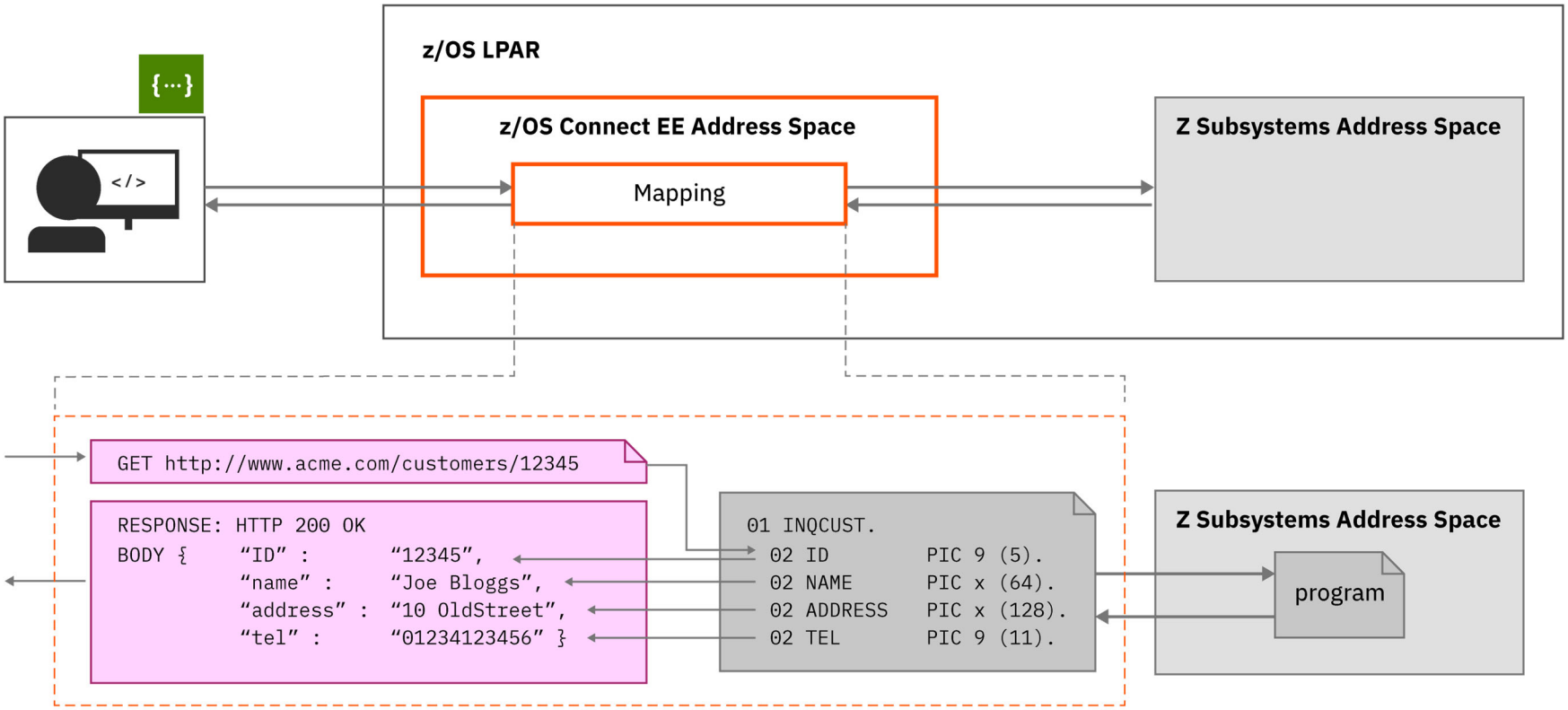


* Most z/OS subsystems depend information in a serial data format and do not normally work with JSON request/response messages. Examples of serialized messages are CICS COMMAREAs and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation refers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

mitchj@us.ibm.com

Data mapping and transformation example

- A closer look



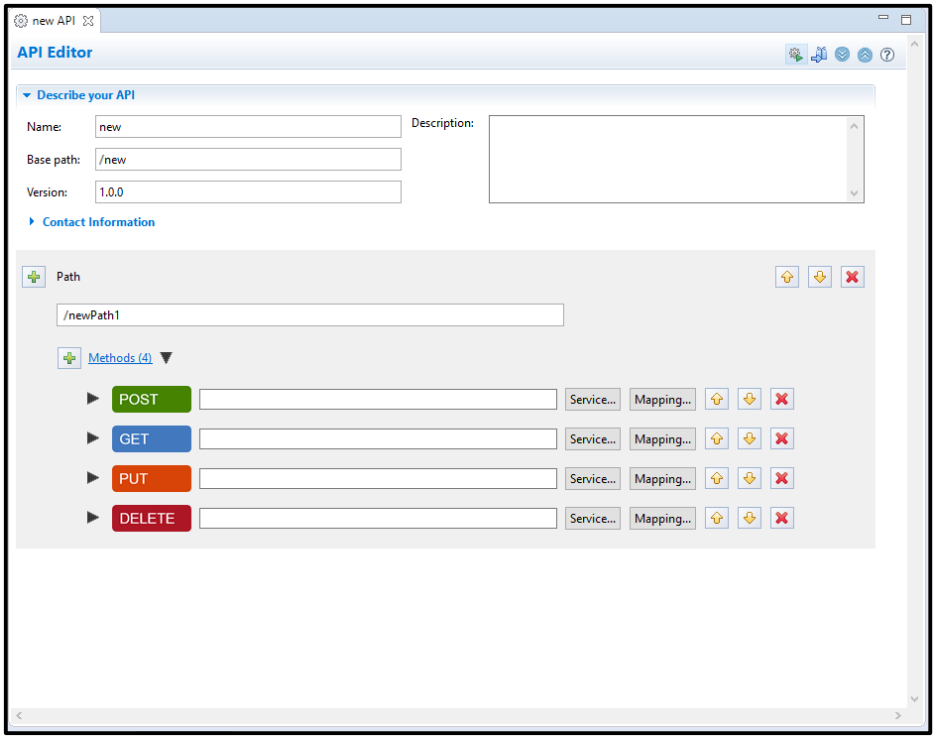
mitchj@us.ibm.com

z/OS Connect OpenAPI Tooling

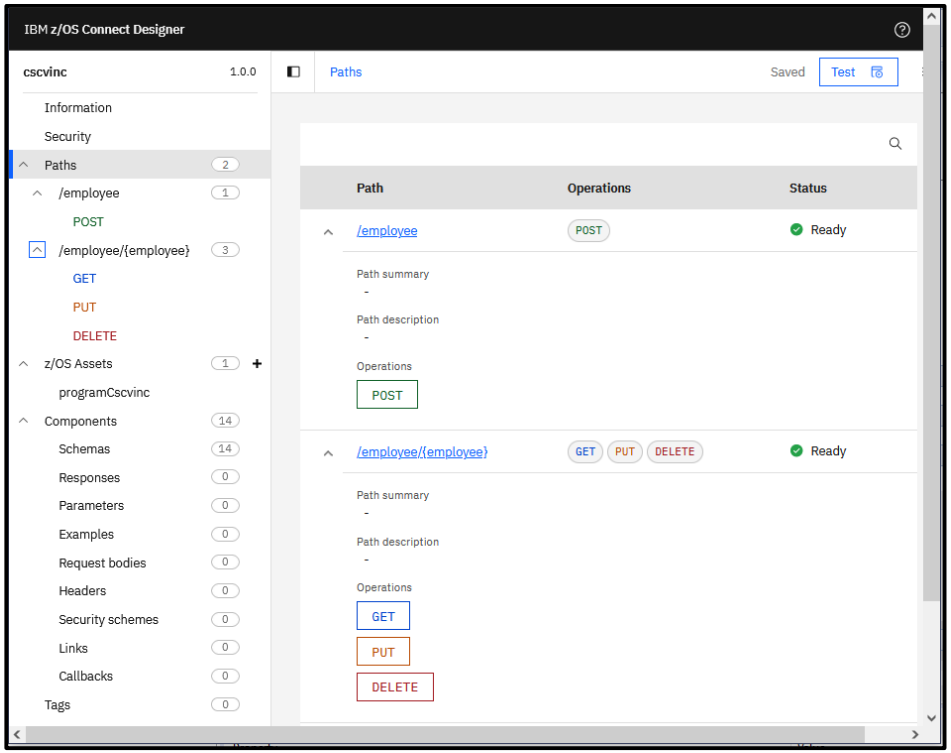
mitchj@us.ibm.com

zCEE - OpenAPI 2 Palette versus the OpenAPI 3 API Designer

z/OS Connect API Toolkit (Eclipse)



z/OS Connect Designer (Designer Container)

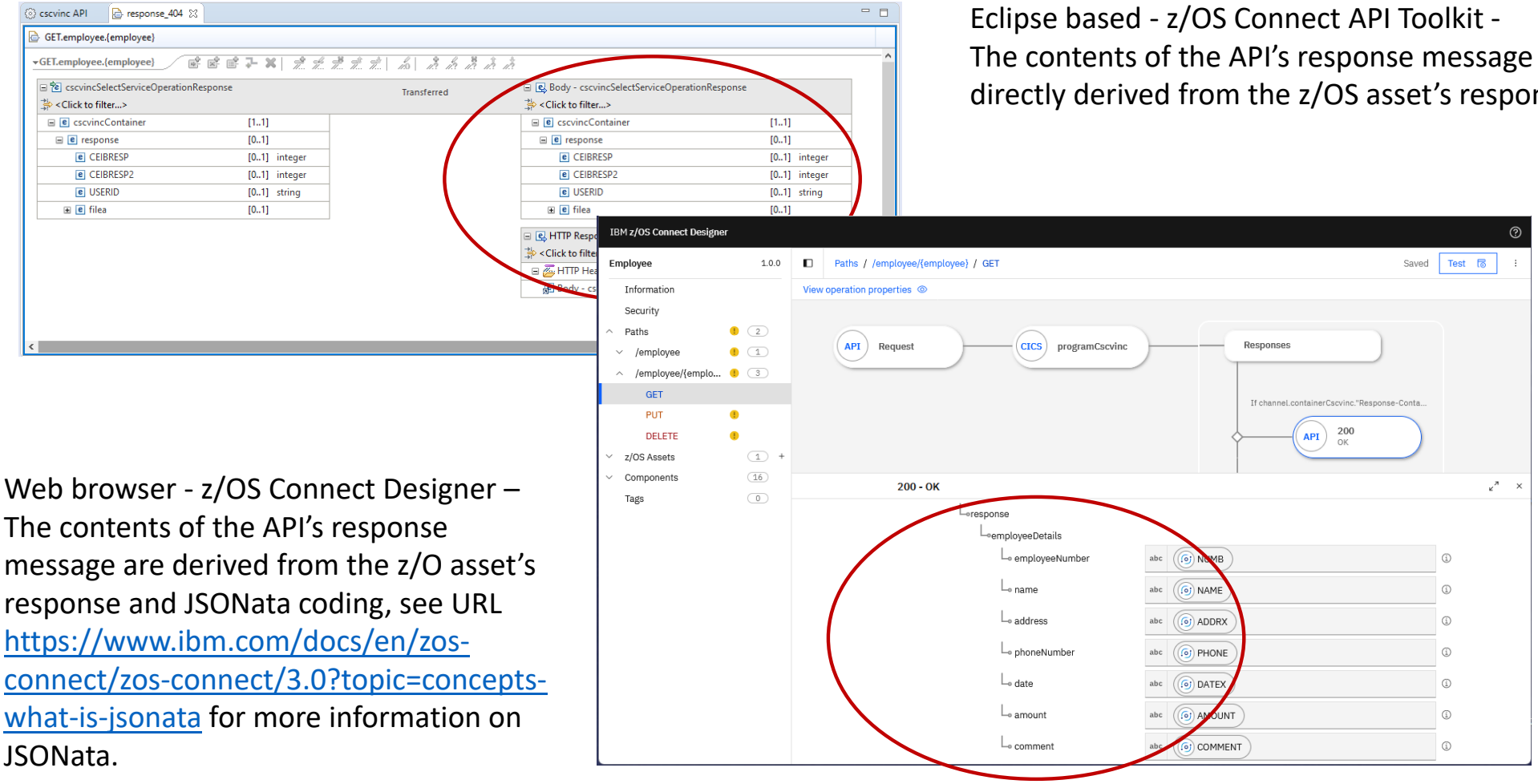


The API toolkit is used to define the URI paths and methods.

The API specification provides predefined URI Paths and methods.

mitchj@us.ibm.com

An OPENAPI 2 versus an OPENAPI 3 Response Message



Eclipse based - z/OS Connect API Toolkit - The contents of the API's response message are directly derived from the z/OS asset's response

Web browser - z/OS Connect Designer – The contents of the API's response message are derived from the z/O asset's response and JSONata coding, see URL https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=concepts-what-is-jsonata for more information on JSONata.

mitchj@us.ibm.com

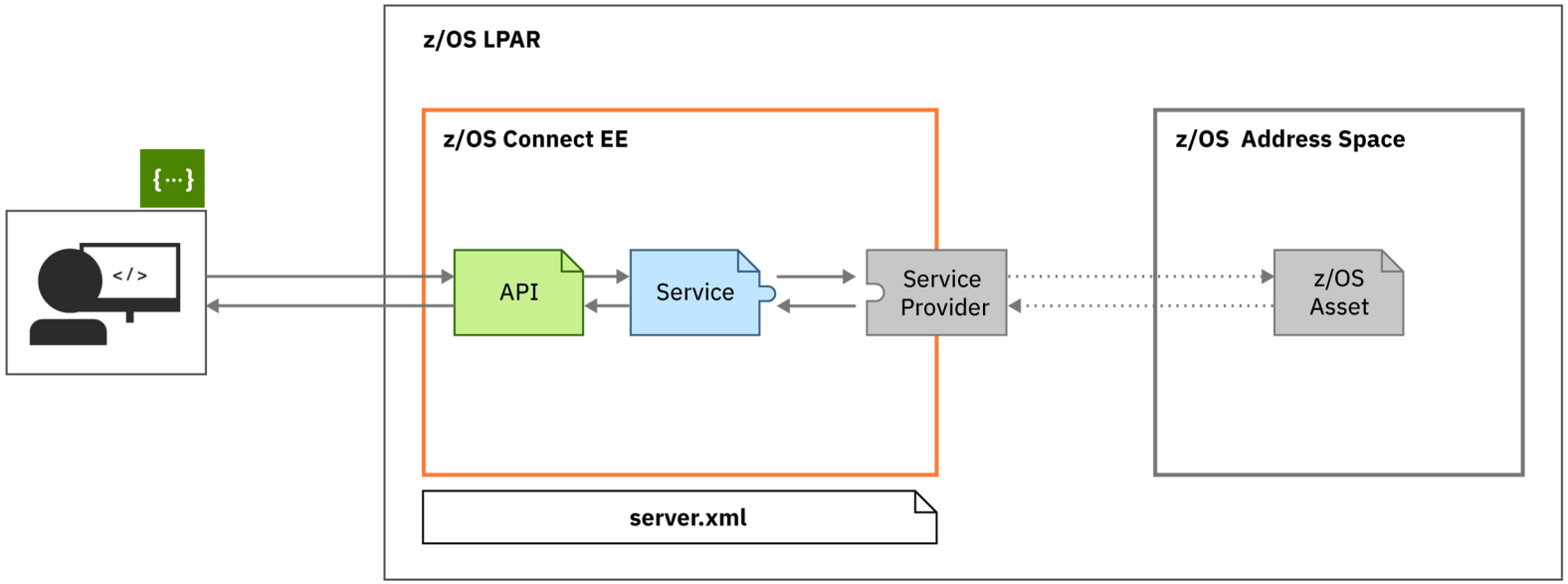
OpenAPI 2

Simple **service creation** not using the Eclipse Toolkit

mitchj@us.ibm.com

Accessing a z/OS asset (Open API 2)

z/OS Connect OpenAPI 2 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)

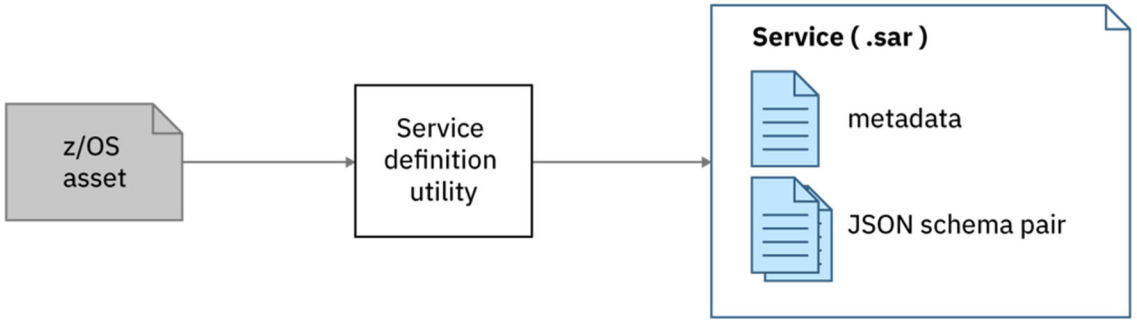
mitchj@us.ibm.com

Describing the interaction (service) with the z/OS resource (OpenAPI 2)

Start by creating a service to represent an interaction with the z/OS 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.
```

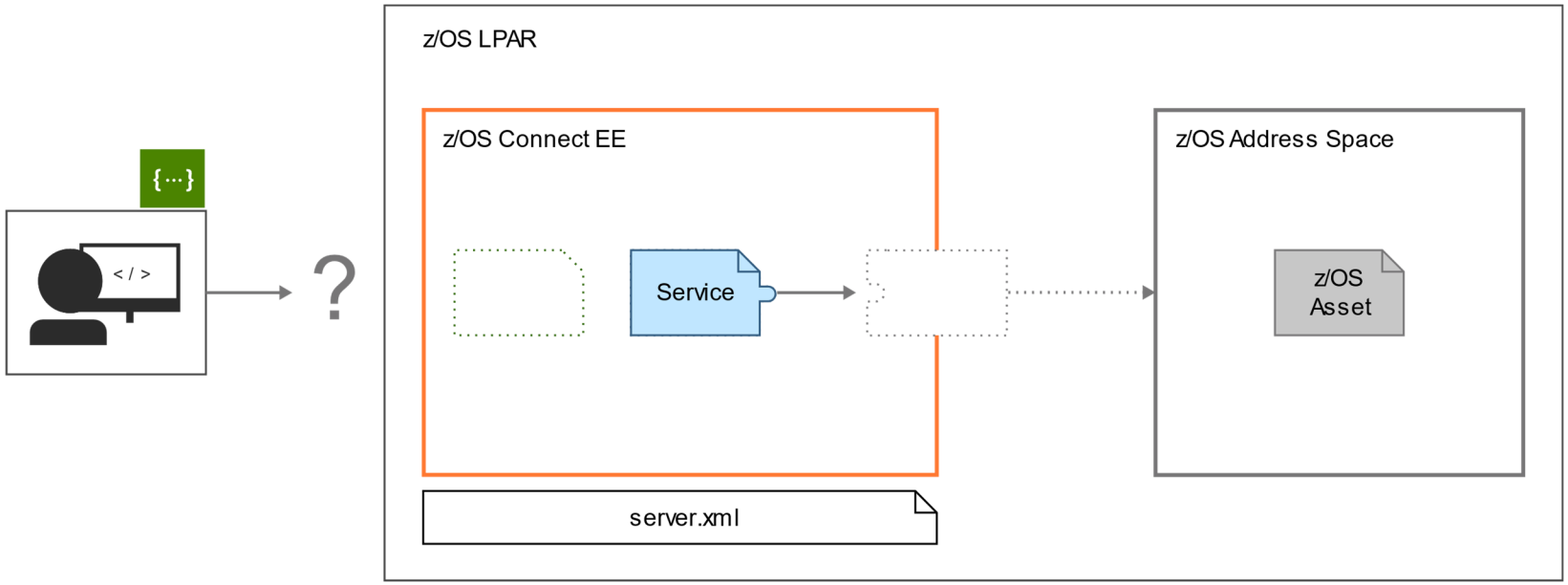
Use a system-appropriate utility to generate a service archive file for the z/OS application

- 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

Deploy and export the service archive (OpenAPI 2)

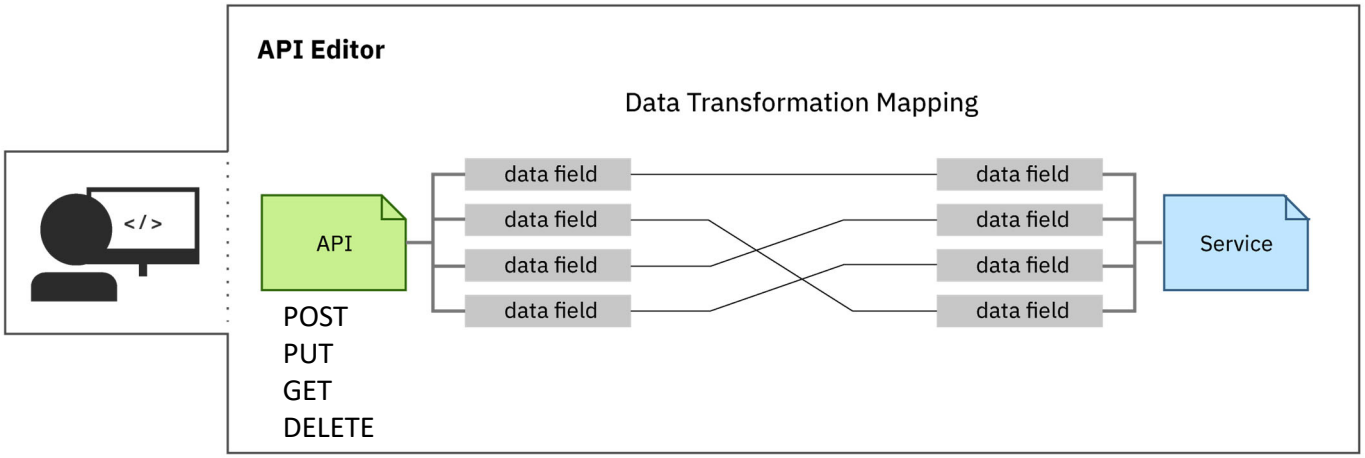
Deploy and export the service archive file



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

Develop an API using the service interactions (Open API 2)

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

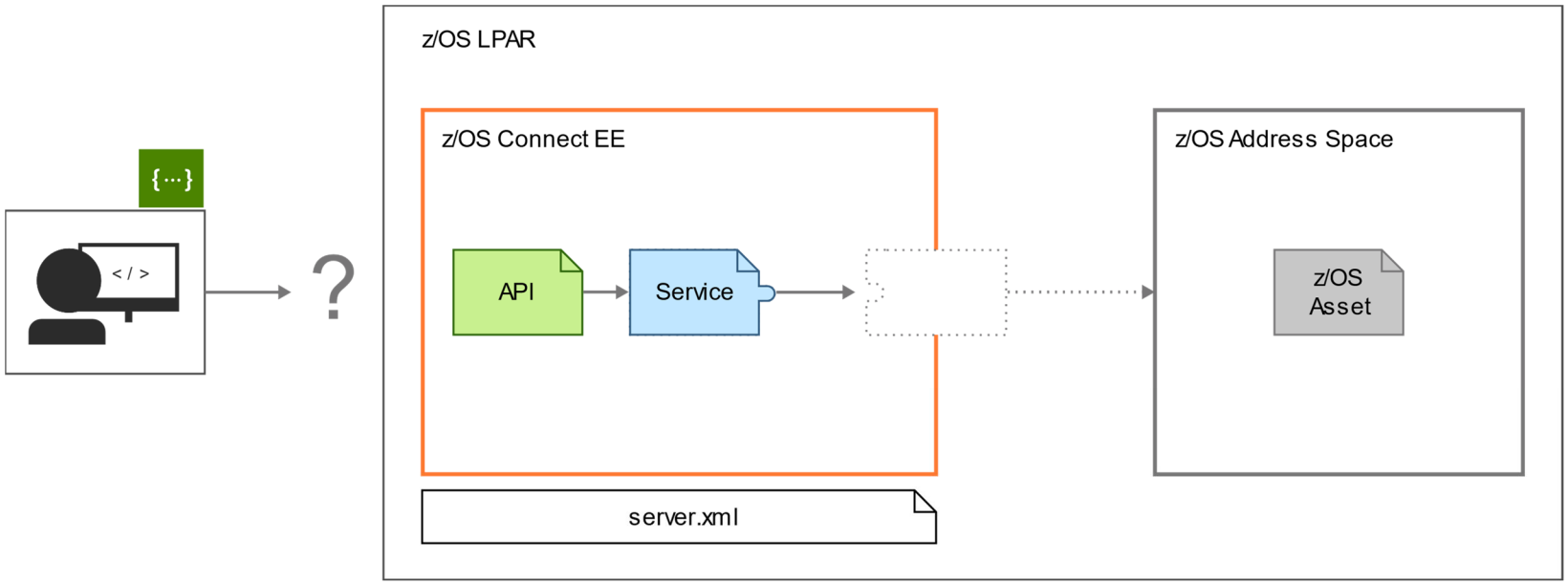


- Import the service archive file into the **API toolkit** and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.

mitchj@us.ibm.com

Deploy the API (Open API 2)

Deploy the API archive file

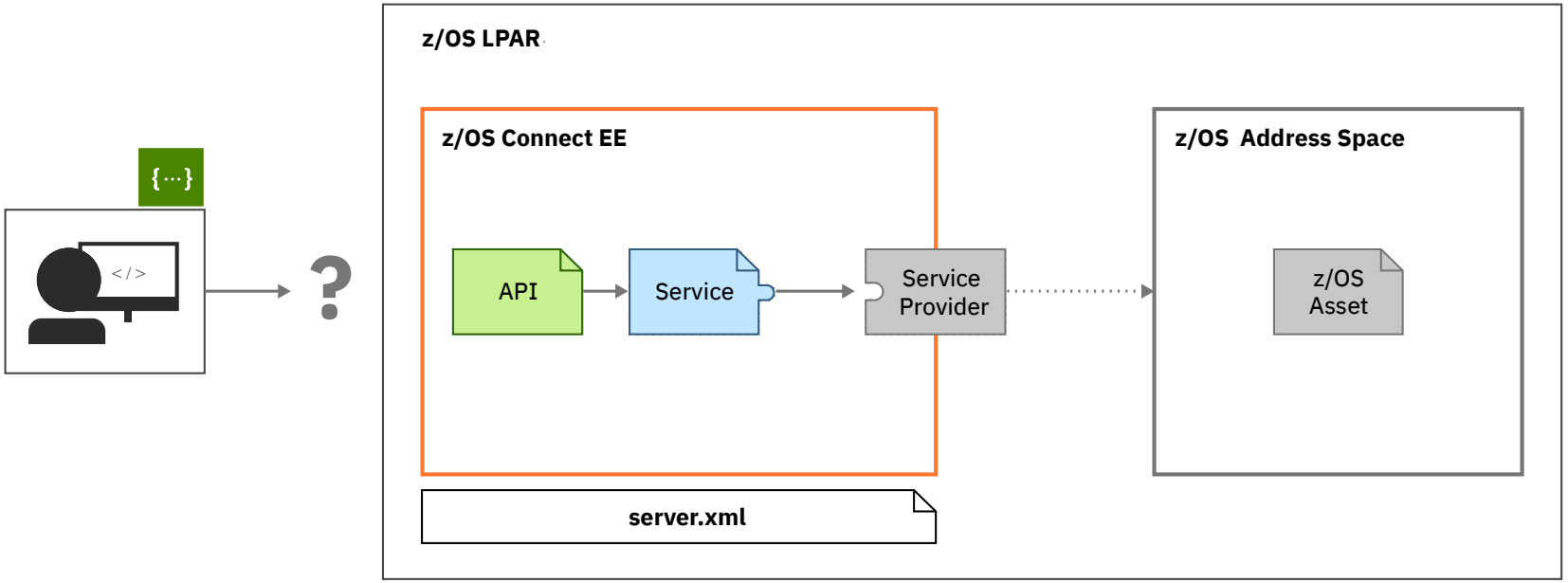


Deploy your API using the right-click deploy in **the API toolkit**

mitchj@us.ibm.com

Configure access the z/OS sub system (Open API 2)

Configure the service provider

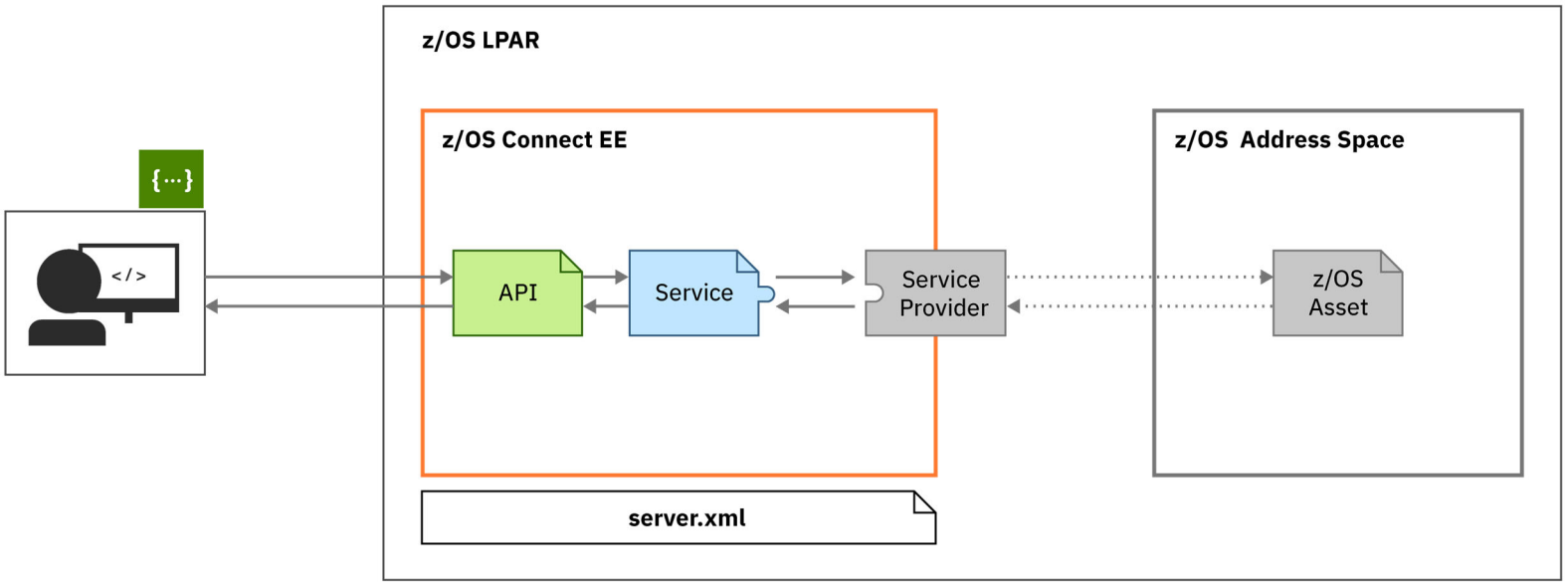


Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.

mitchj@us.ibm.com

Complete access to a z/OS Asset (Open API 2)

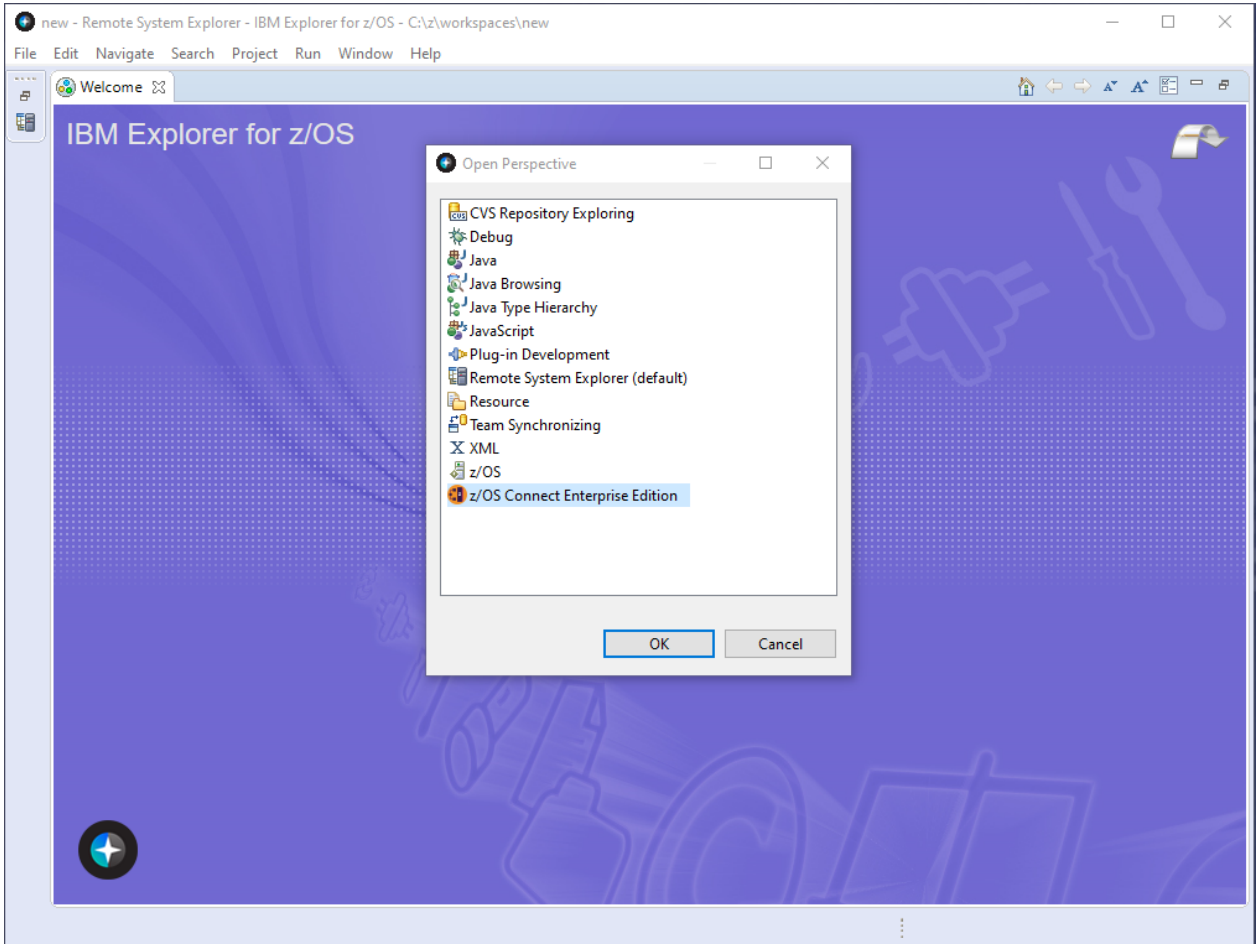
Done



- 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

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

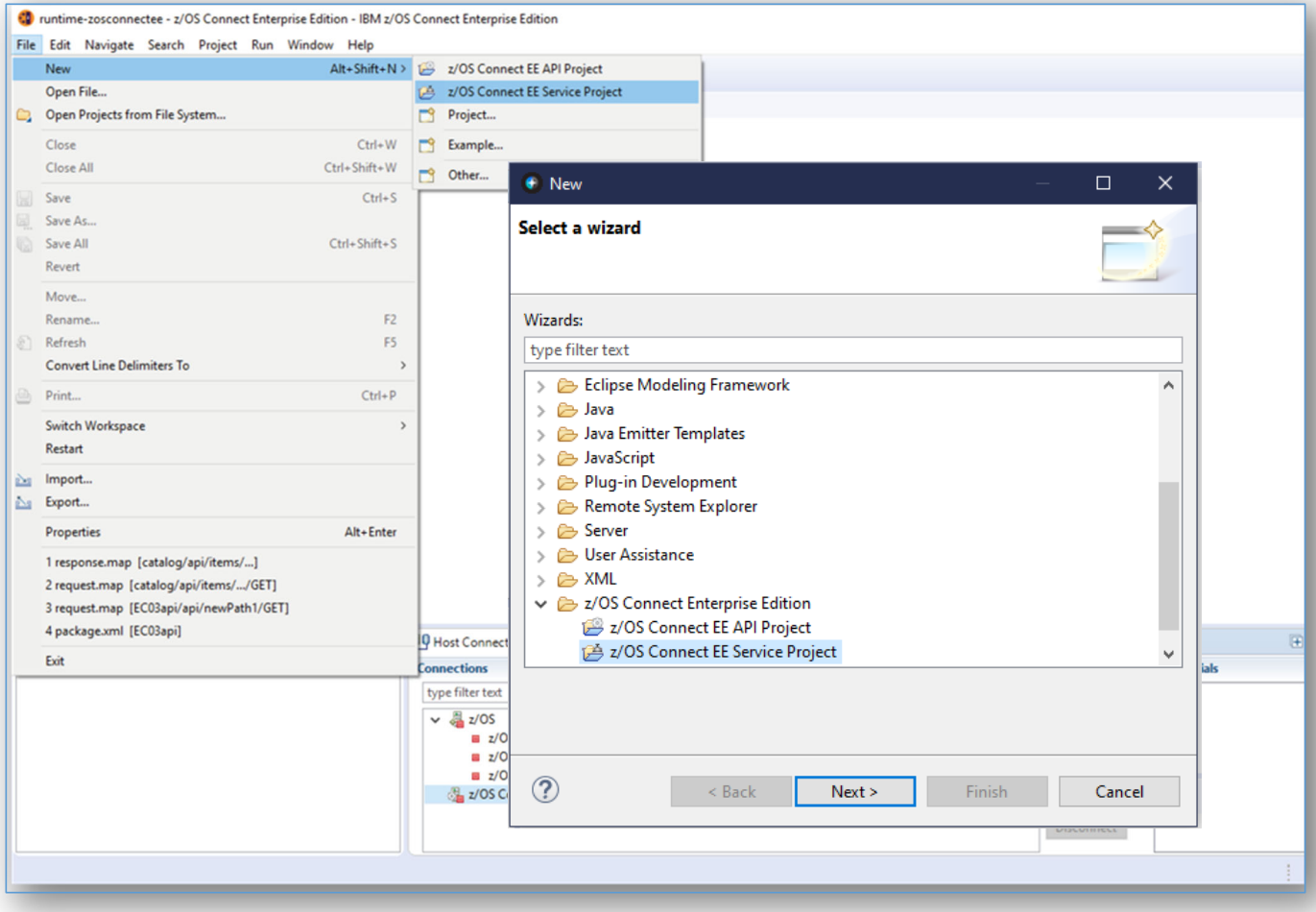


Use the **API toolkit** to create services
through Eclipse-based tooling.

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



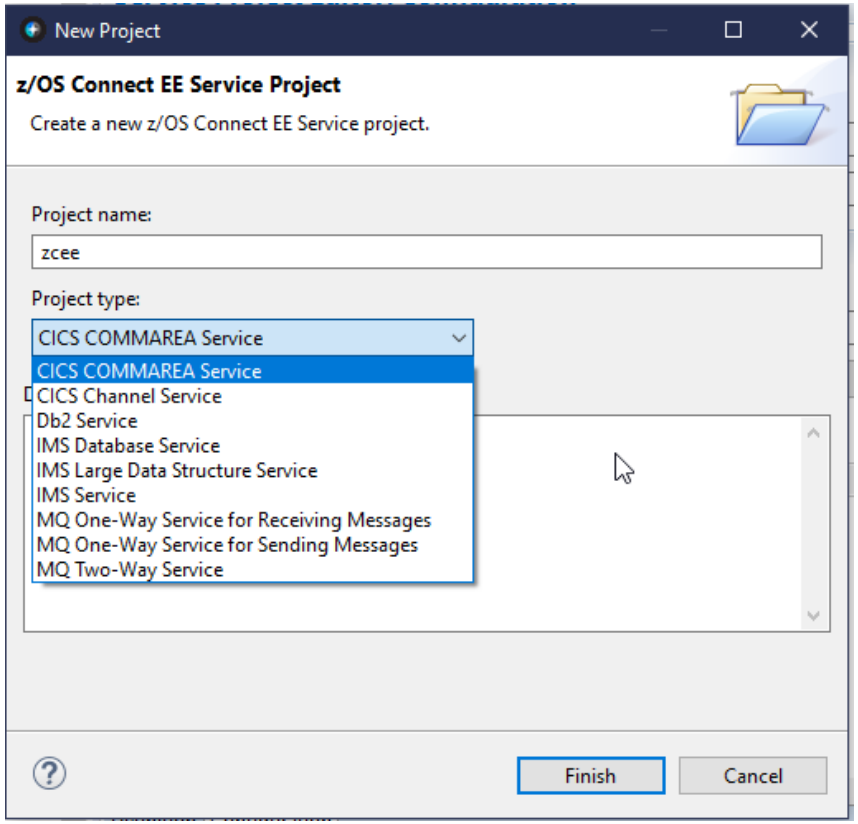
Use the **API toolkit** to create services through Eclipse-based tooling.

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

Service creation – a common interface



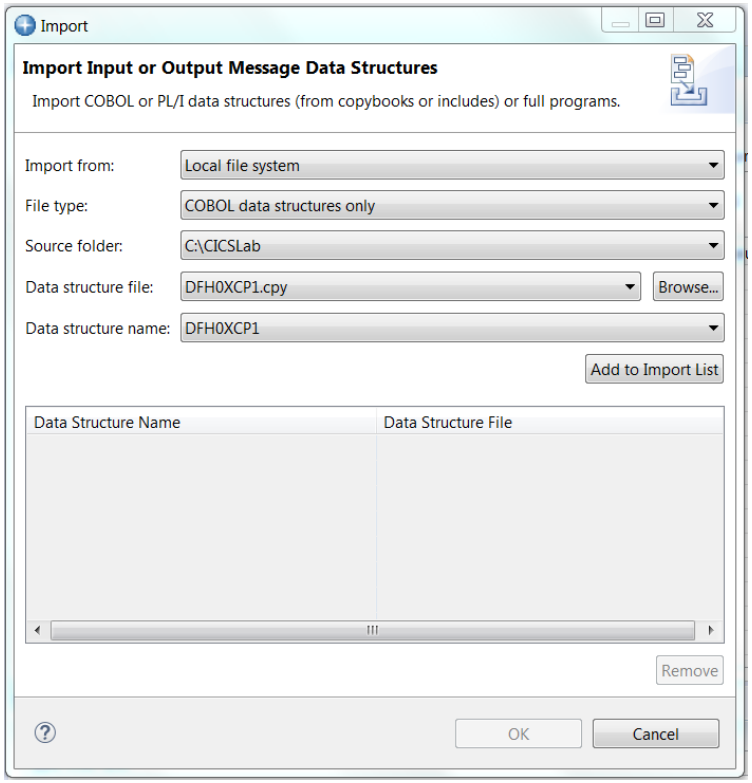
- CICS services that can invoke almost any CICS programs accessed by EXEC CICS LINK request (COMMAREA or CHANNEL). See URL http://www.ibm.com/docs/en/cics-ts/5.6?topic=link-exception-conditions-command for a list of EXEC CICS APIs not allowed in a program when invoked using a CICS Dynamic Program Link request.
- Db2 services that invoke a Db2 REST service.
- IMS DB services that access an IMS database.
- IMS TM services that sends a messages on an IMS message processing region.
- MQ services that use MQ request/reply queues for two-way services or access a single queue for MQ PUTs and MQ GETs on a either a local or remote queue manager

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

mitchj@us.ibm.com

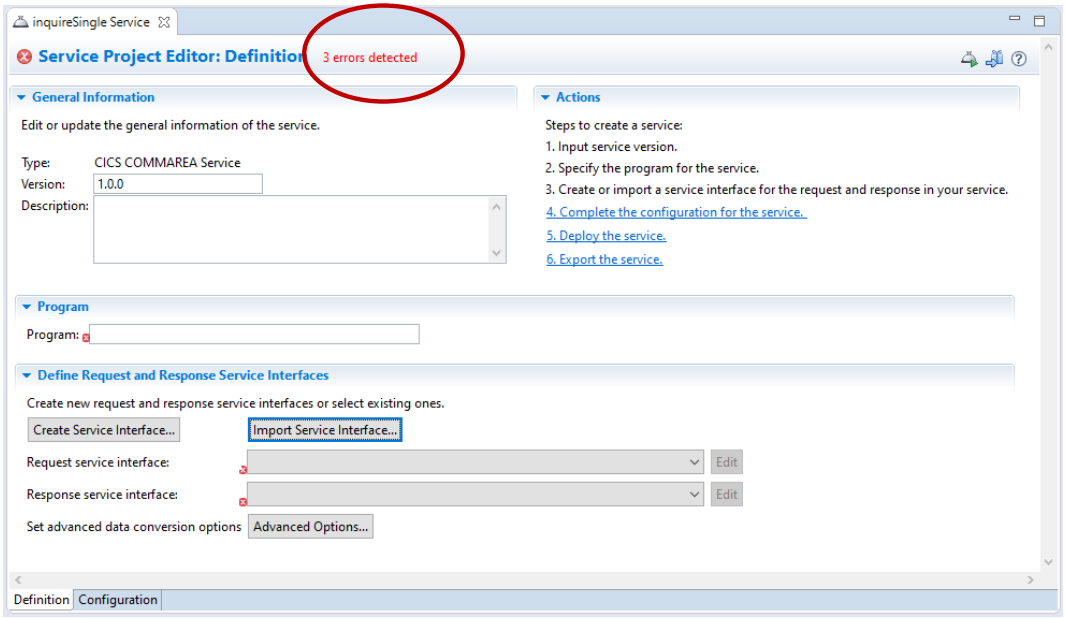
API toolkit – Creating Services for CICS, IMS TM and MQ

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.



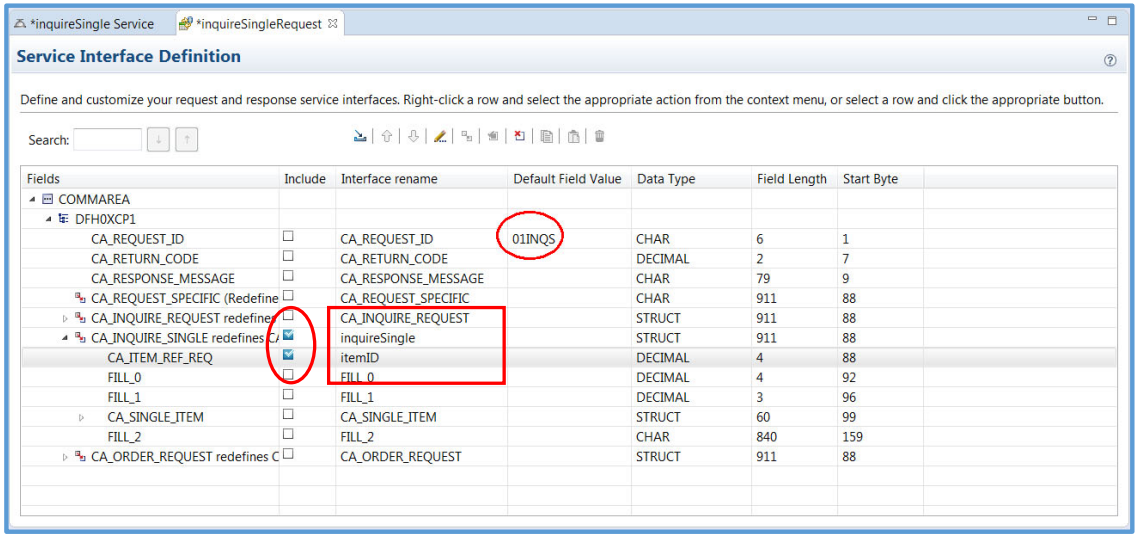
mitchj@us.ibm.com

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

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
```

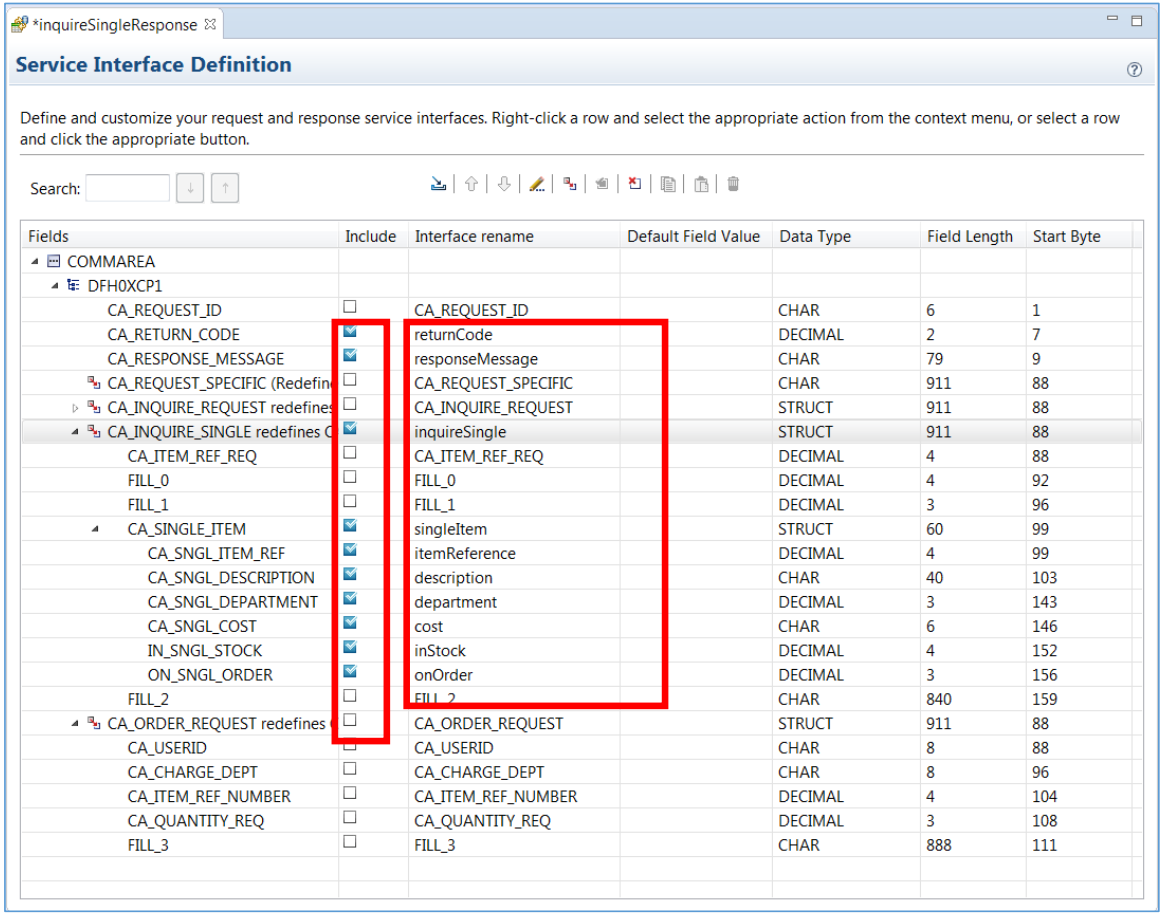
See the imported data structure and then can **redact fields**, **rename fields**, and **add default values to fields** to make the service more consumable for an API developer.



mitchj@us.ibm.com

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

And editing a response message service interface definition



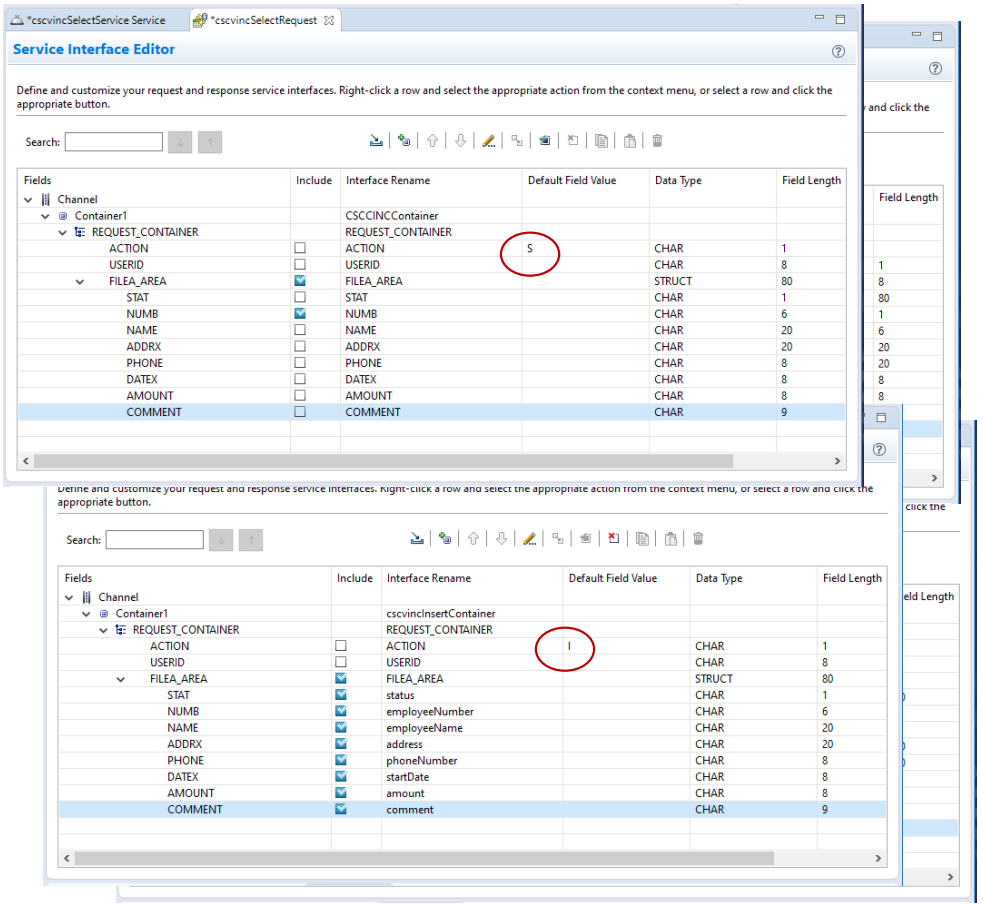
Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID		CHAR	6	1
CA_RETURN_CODE	☑	returnCode		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☑	responseMessage		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefin	☐	CA_REQUEST_SPECIFIC		CHAR	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

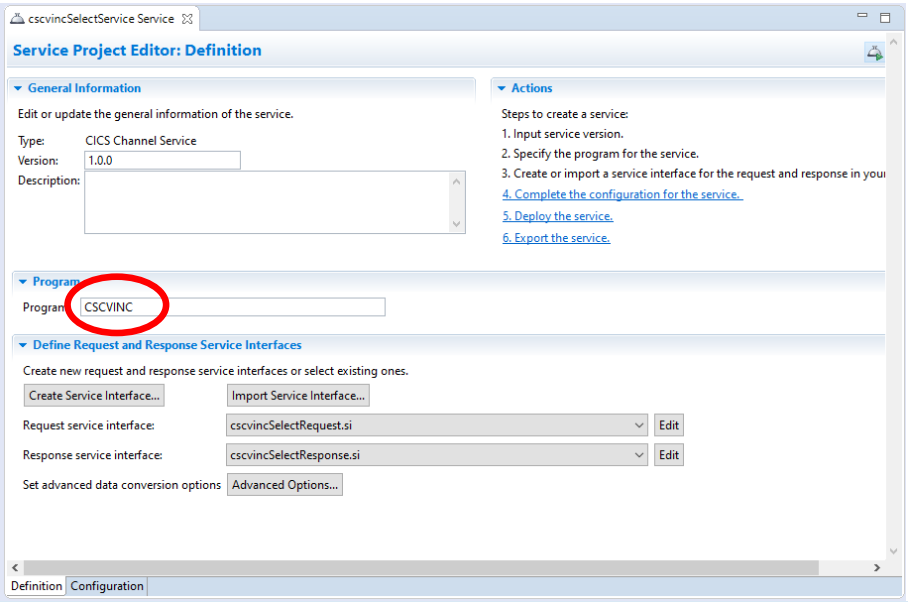
See the imported data structure and can **redact fields** and **rename fields**

mitchj@us.ibm.com

API toolkit – Creating Services for CICS

Creating multiple services definitions to the same resource



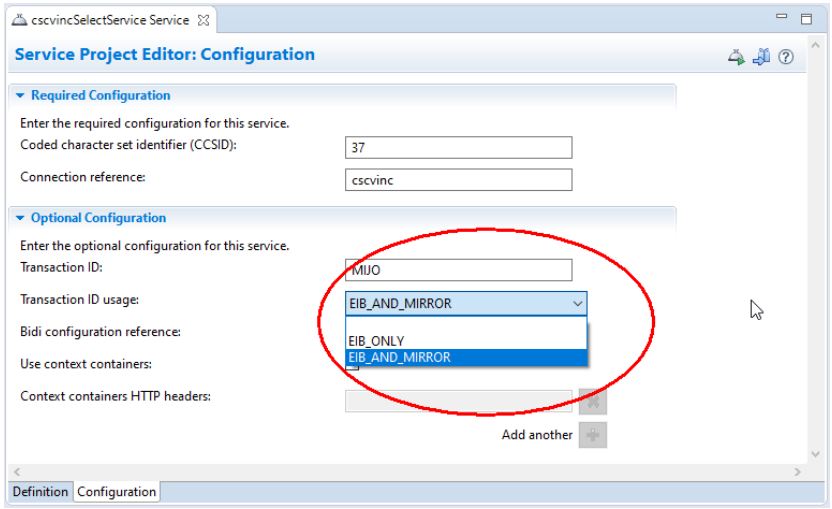



```
EVALUATE ACTION of Request-Container
    WHEN 'D'
      PERFORM Delete-Record
    WHEN 'I'
      PERFORM Insert-Record
    WHEN 'U'
      PERFORM Update-Record
    WHEN 'S'
      PERFORM Select-Record
 END-EVALUATE.
```

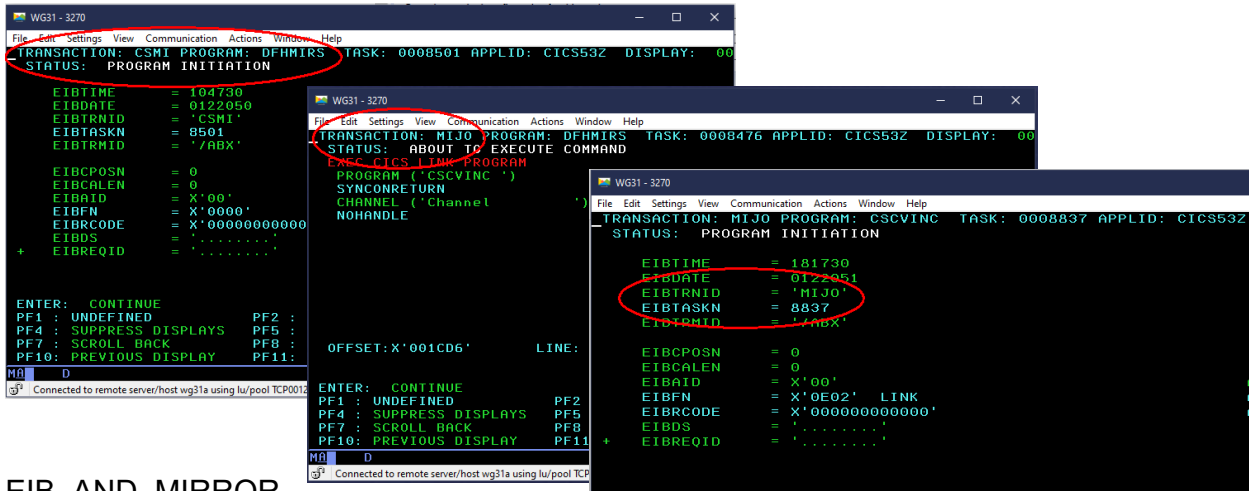
The service developer creates distinct services for each function by setting the ACTION field to S for select, I for insert, U for update or D for delete

mitchj@us.ibm.com

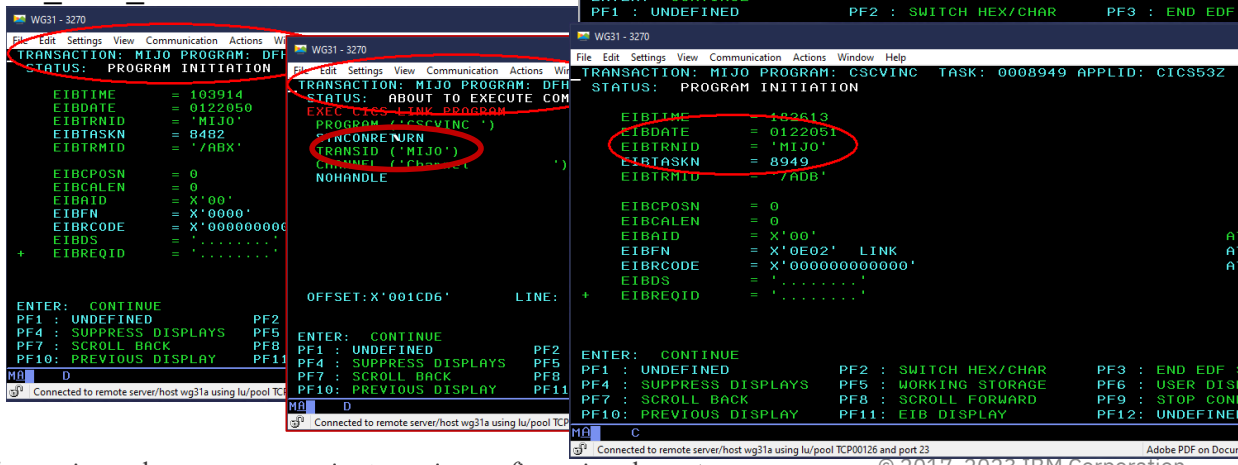
Accessing a CICS program – Transaction ID Usage



EIB_ONLY



EIB_AND_MIRROR



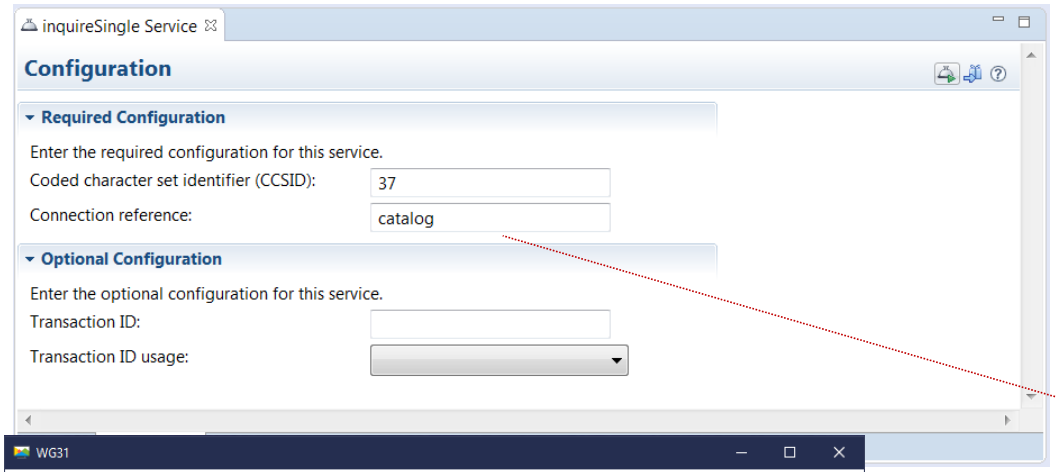
- **Transaction ID** attaches a CICS transaction (CSMI is the default) that starts the CICS DFHMIRS program.

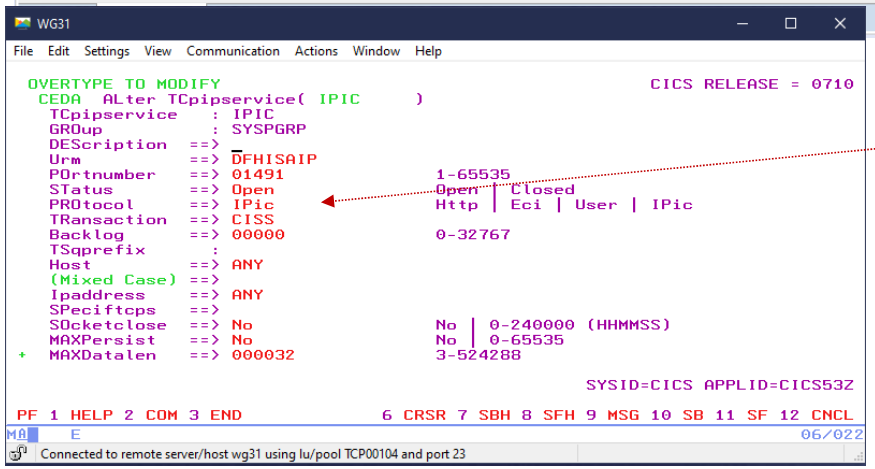
- **Transaction ID Usage** attribute useful for:
  - Transaction security requirements
  - Db2 plan selection
  - Transaction classification and reporting

These attributes also be used in *zosconnect_cicsIpicConnection* and *zosconnect_services>service configuration* elements.

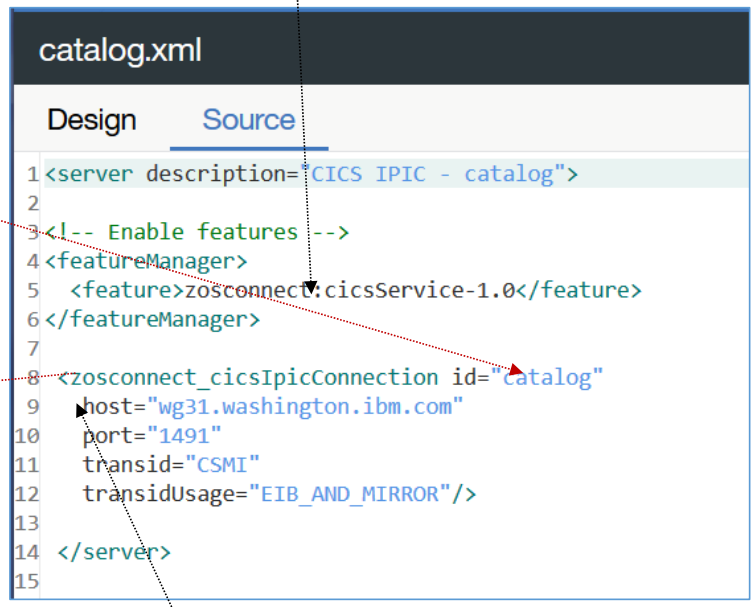
Accessing a CICS program uses CICS IP interconnectivy (IPIC)

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



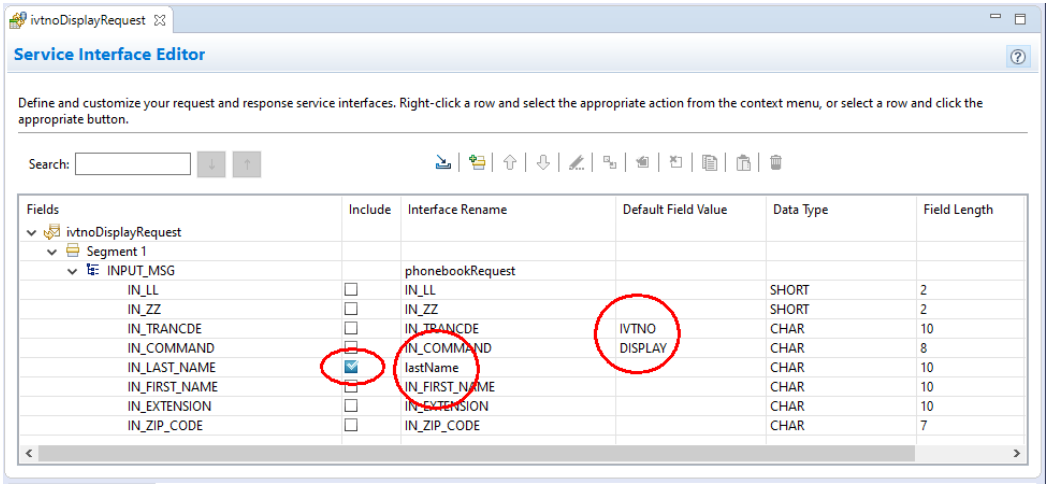
Define IPIC connection to CICS

mitchj@us.ibm.com

API toolkit – Creating Services for IMS

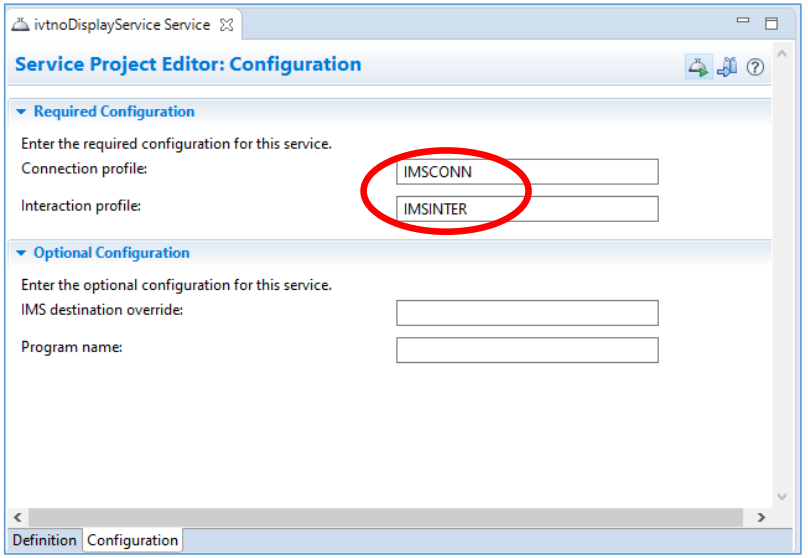
Creating a “GET” service interface request definition

```
*-------------------------------------------------------------------*
*       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
```

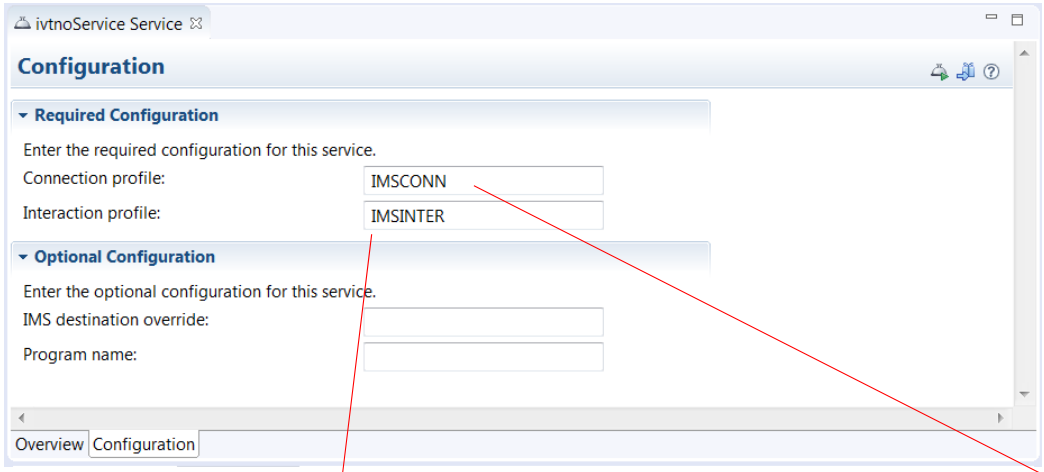


The service developer creates distinct services for each function.

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



IMS Connections and Interactions



Connection Profile

```
<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 Profile

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

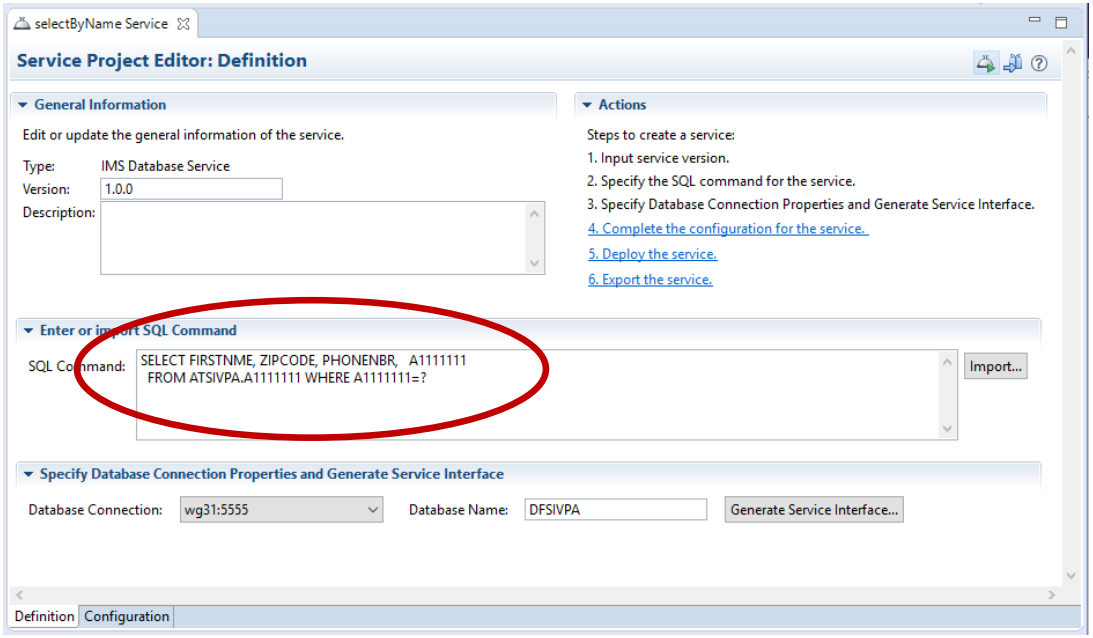
IMS Connect HWSCFG

```
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)
```

mitchj@us.ibm.com

API toolkit – Creating Services for IMS DB

Creating a service project from the IMS Catalog



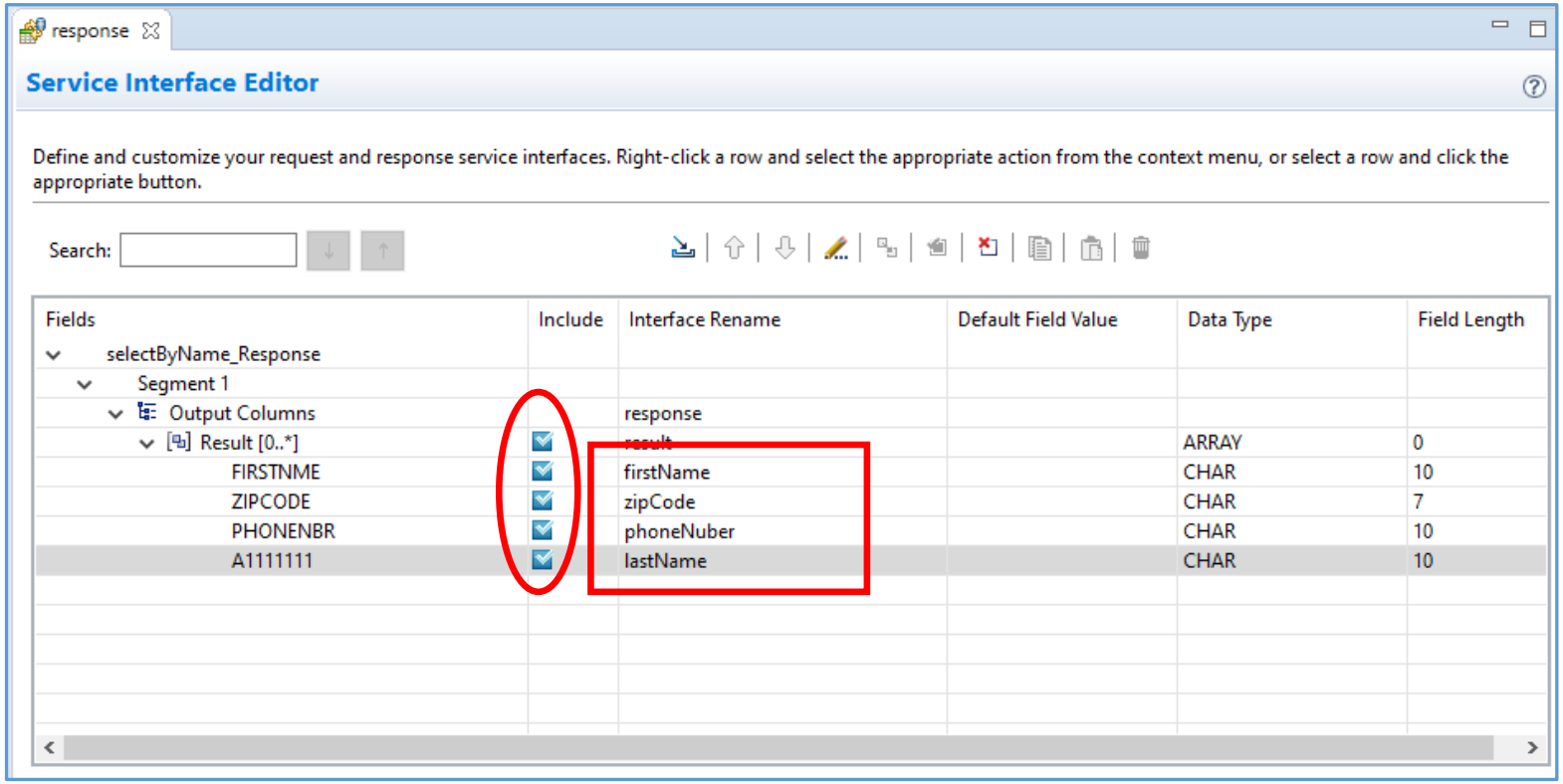
Use the IMS Catalog to assist with developing and testing SQL SELECT commands used for accessing IMS databases.

```
*-------------------------------------------------------------------*
*  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

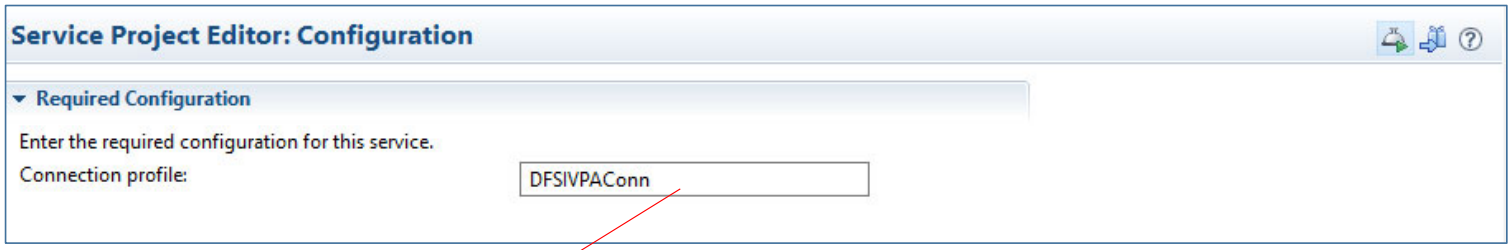
The Toolkit allows editing a service interface definitions*



*Using a slightly different process

mitchj@us.ibm.com

IMS Connection Factory in the server XML



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

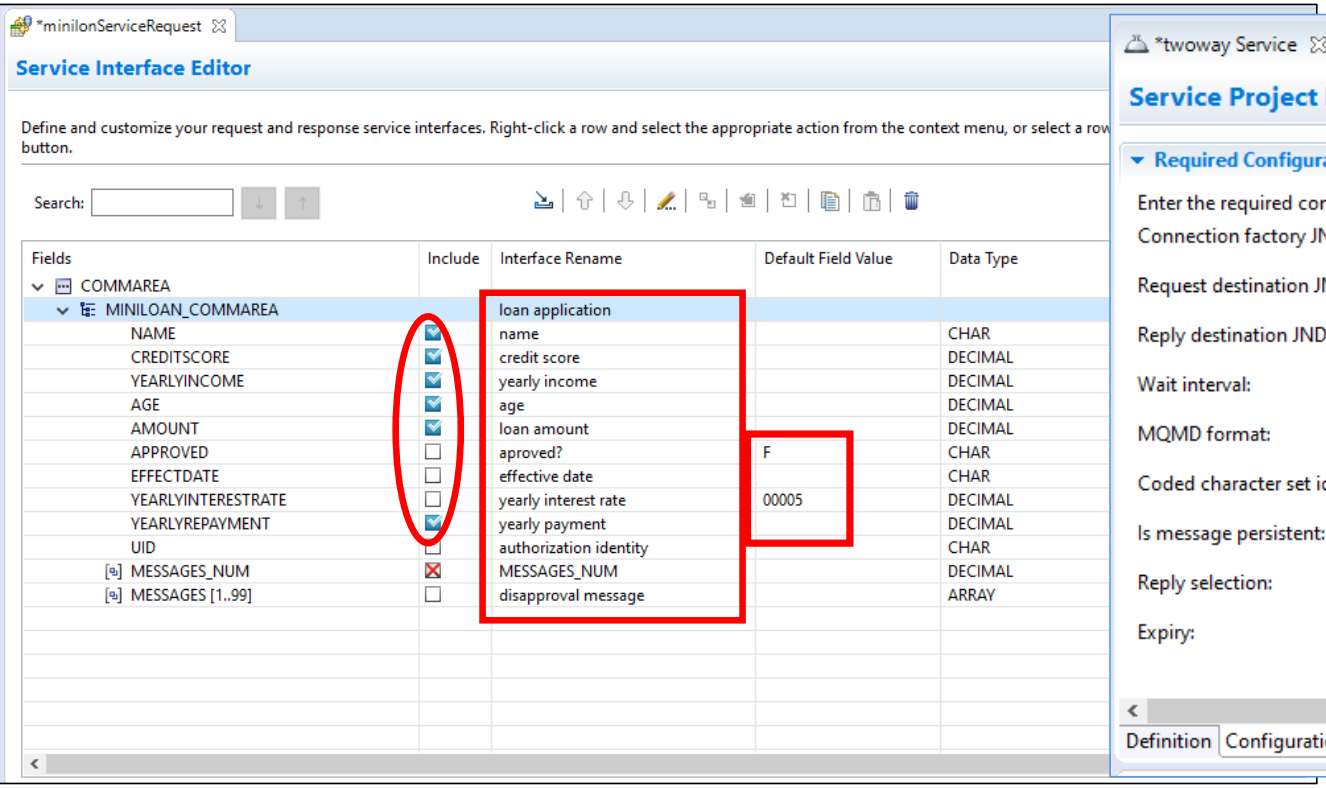
```
HWS=(ID=IMS14HWS,XIBAREA=100,RACF=N,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)
DATASTORE=(GROUP=OTMAGRP,ID=IVP1,MEMBER=HWSMEM,TMEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS,TMEMBER=PLEX1)
ODACCESS=(ODBMAUTOCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000),ODBMTMOT=6000)
```

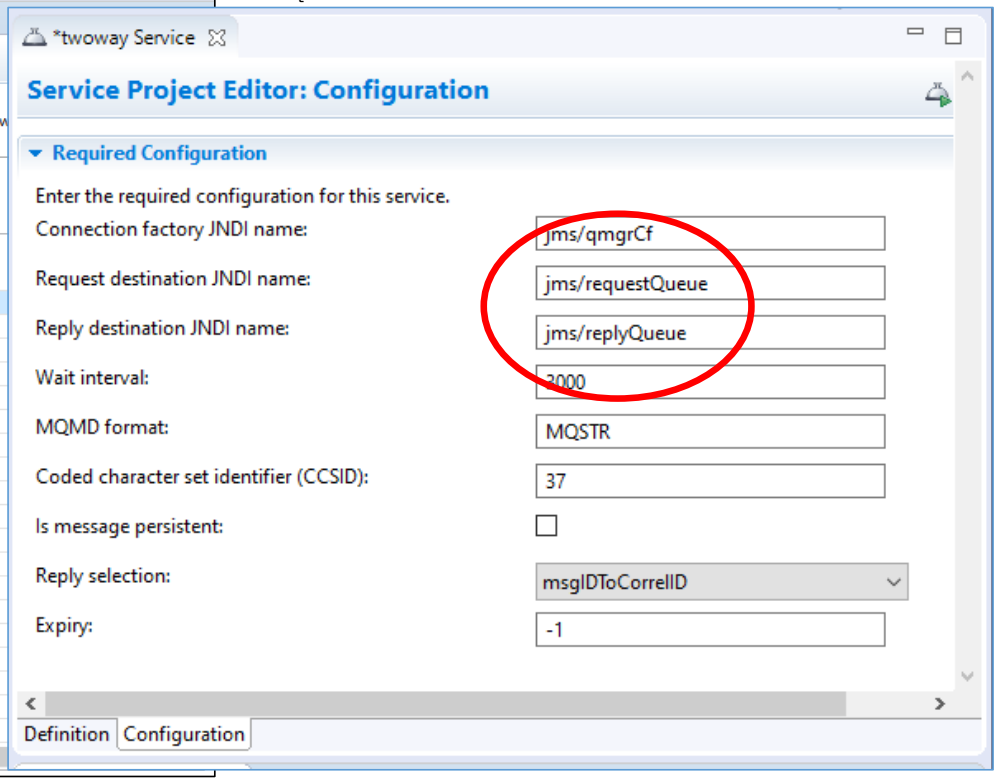
mitchj@us.ibm.com

API toolkit – Creating Services for MQ

Creating a MQ PUT (“POST”) service interface definition

MQ Meta Data

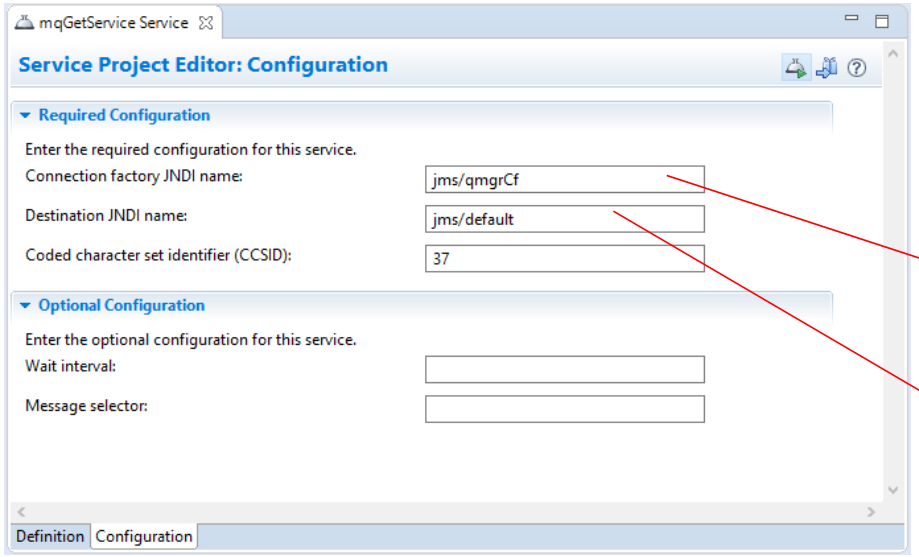




Again the service developer can then see the imported data structure and can **redact fields**, **rename fields**, and **add default values to fields** to make the service more consumable for an API developer.

mitchj@us.ibm.com

Using JMS to access MQ (One-Way)



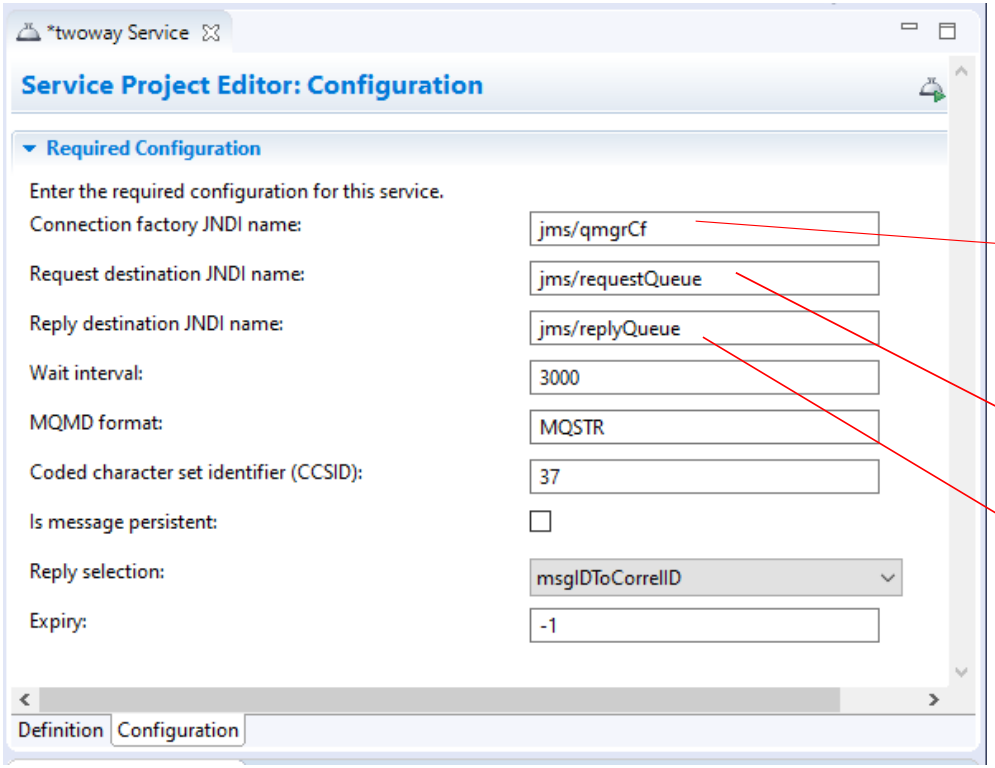
mqClient.xml — Read only — Close

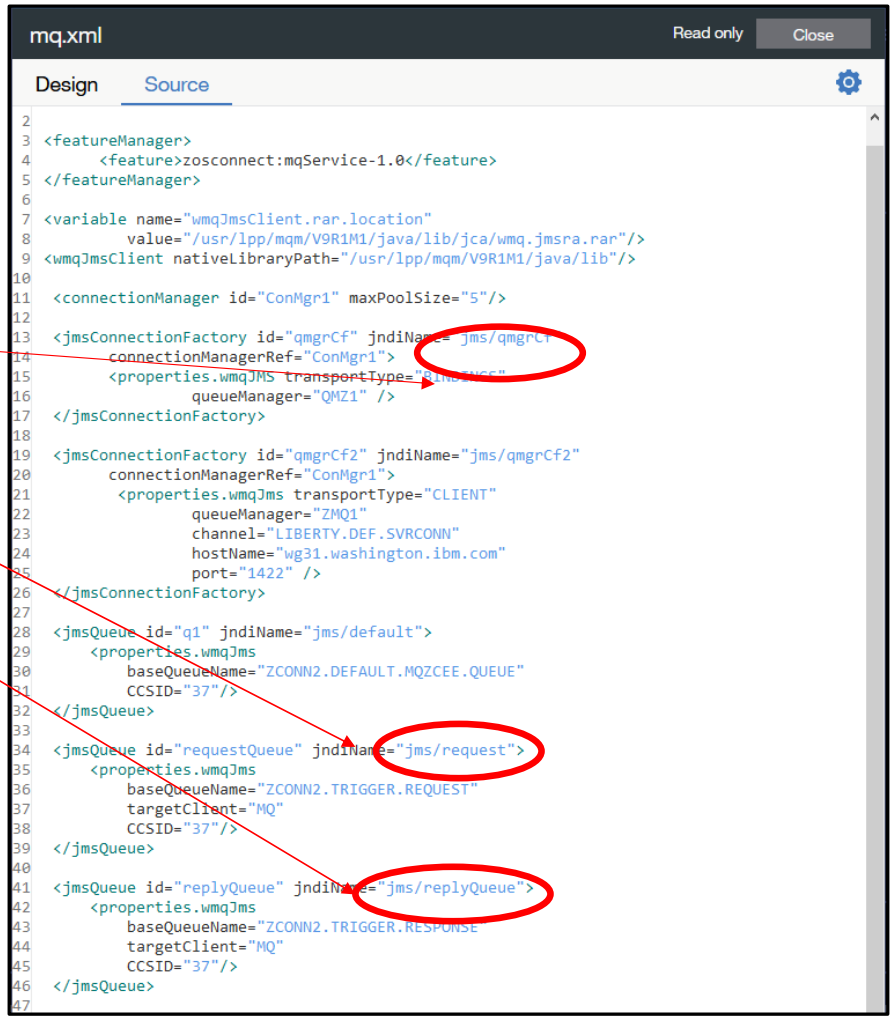
Design | Source

```
1  <server description="MQ Service Provider">
2
3   <featureManager>
4         <feature>zosconnect:mqService-1.0</feature>
5   </featureManager>
6
7   <variable name="wmqJmsClient.rar.location"
8             value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
9   <wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
10
11  <zosconnect_services>
12        <service name="mqPutService">
13              <property name="useCallerPrincipal" value="false"/>
14        </service>
15  </zosconnect_services>
16
17   <connectionManager id="ConMgr1" maxPoolSize="5"/>
18
19   <jmsConnectionFactory id="qmgrCf" jndiName="jms/qmgrCf"
20          connectionManagerRef="ConMgr1">
21         <properties.wmqJMS transportType="CLIENT"
22                queueManager="ZMQ1"
23                channel="LIBERTY.DEF.SVRCONN"
24                hostName="wg31.washington.ibm.com"
25                port="1422" />
26   </jmsConnectionFactory>
27
28   <jmsQueue id="q1" jndiName="jms/default">
29       <properties.wmqJms
30           baseQueueName="ZCEE.DEFAULT.MQZCEE.QUEUE"
31           CCSID="37"/>
32   </jmsQueue>
33
34 </server>
35
```

mitchj@us.ibm.com

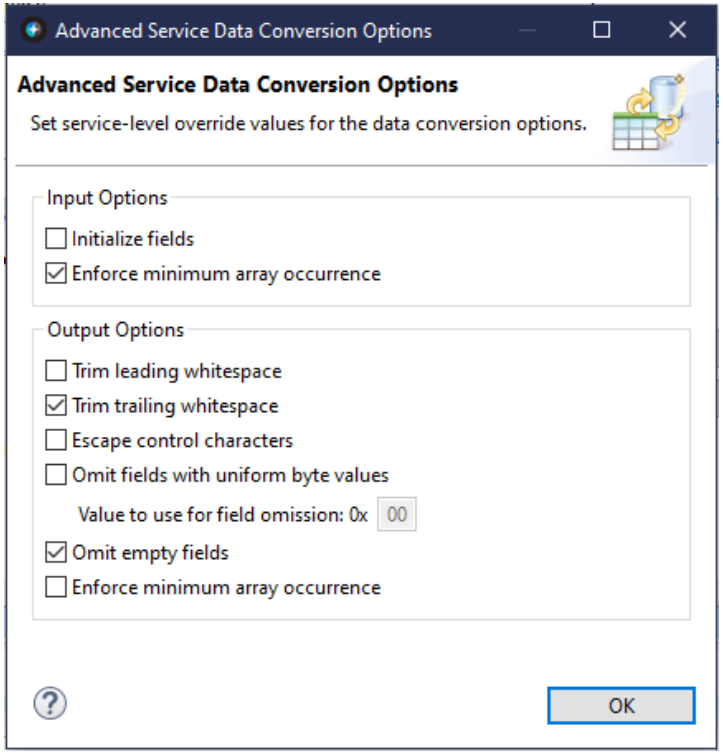
Using JMS to access MQ (Two-Way)





mitchj@us.ibm.com

API toolkit – Advanced Data Conversion Options



Request Messages:

- Initialize fields
- Enforce minimum array occurrence

Response Messages:

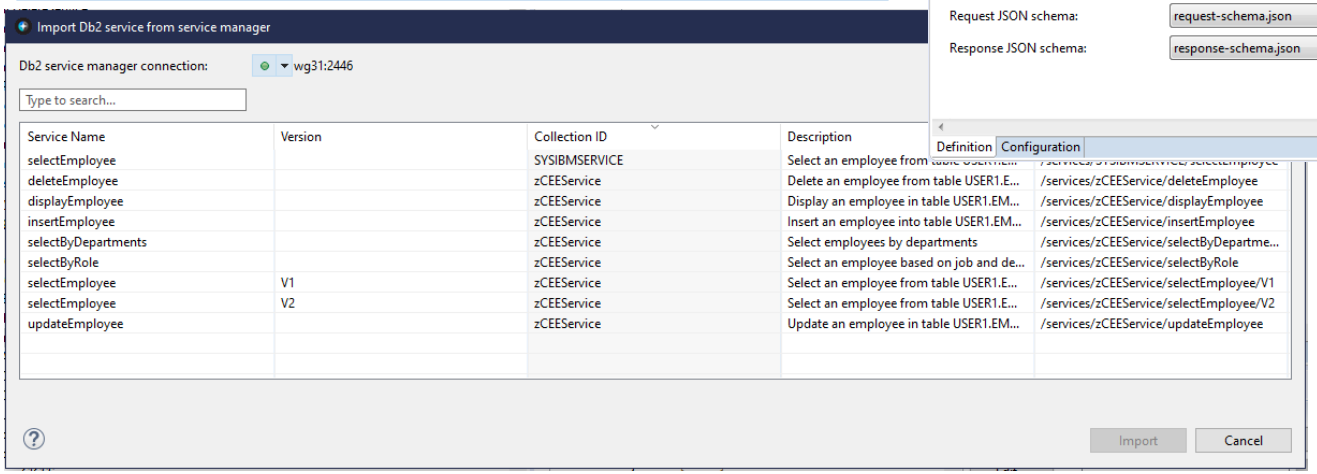
- Trim leading whitespace
- Trim trailing whitespace
- Escape control characters
- Omit fields with uniform byte values
- Omit empty fields
- Enforce minimum array occurrence

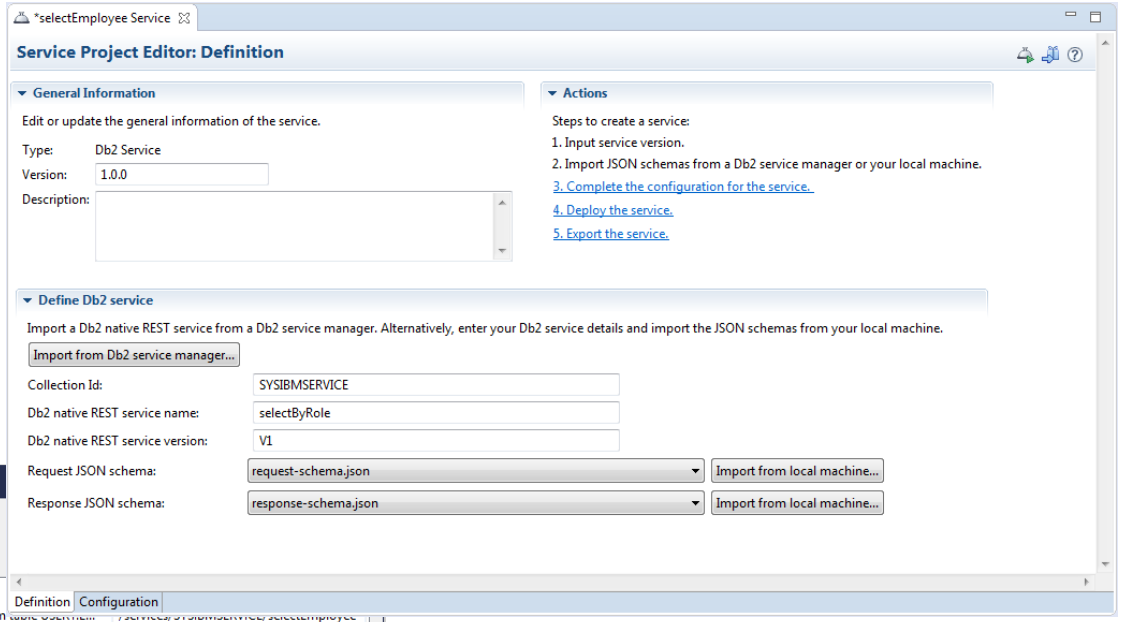
mitchj@us.ibm.com

API toolkit – Creating Services for Db2

Creating a service project from Db2 REST service

```
//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')
```



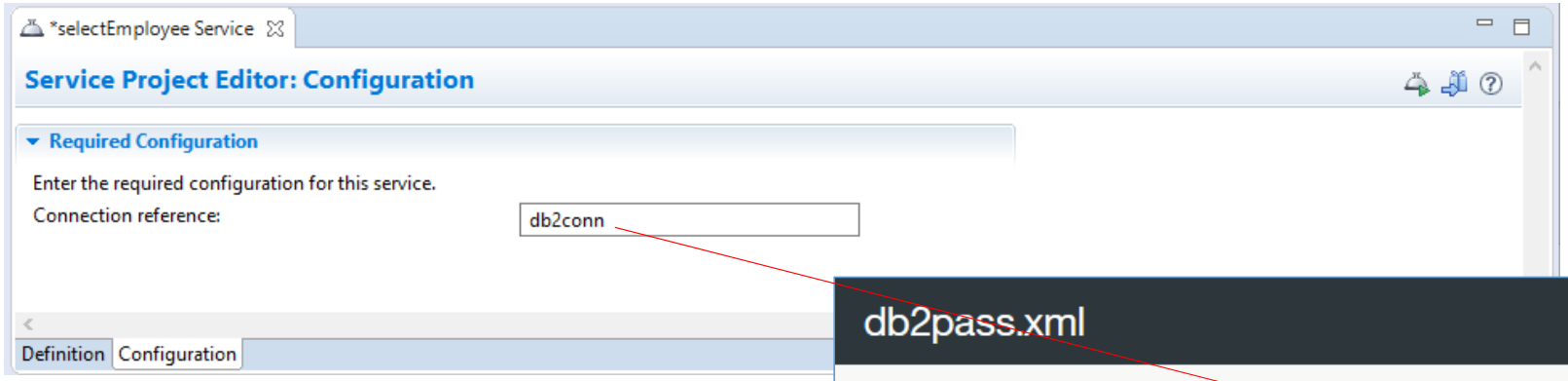


The service developer retrieves details about the Db2 REST services

Note there is no service interface editor available

mitchj@us.ibm.com

Accessing a Db2 REST service resource




```
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

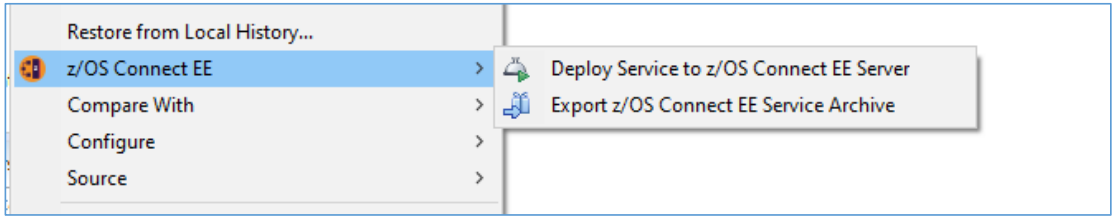
```
1  <server description="DB2 REST">
2
3   <zosconnect_zosConnectServiceRestClientConnection id="db2conn"
4         host="wg31.washington.ibm.com"
5         port="2446"
6         basicAuthRef="dsn2Auth" />
7
8   <zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
9         applName="DSN2APPL"/>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

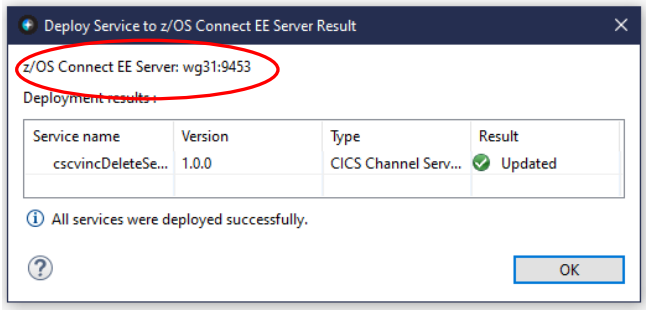
mitchj@us.ibm.com

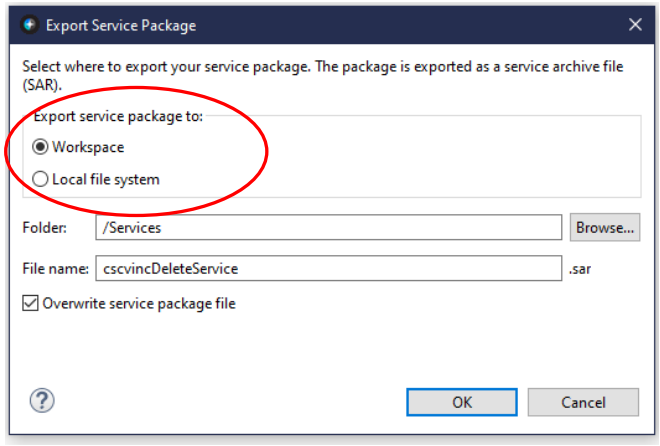
API toolkit – Services Editor

Server connection and Services deployment

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







mitchj@us.ibm.com

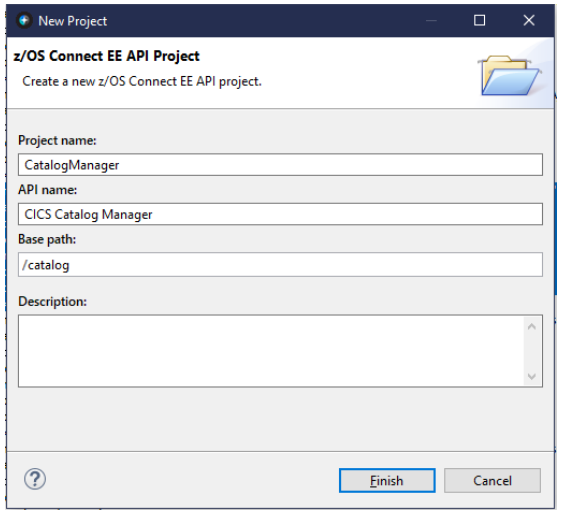
Once we have a Service Archive (SAR) What's next?

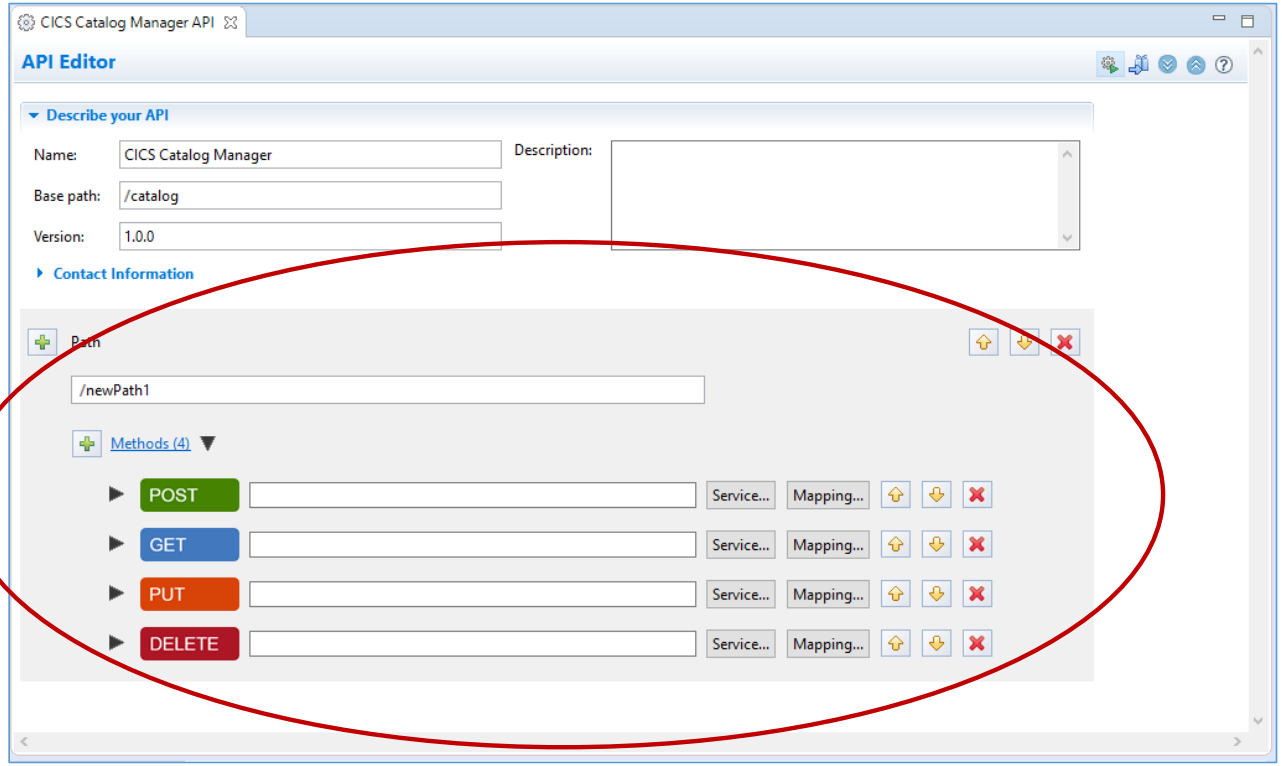
Quick and easy **API mapping.**

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

mitchj@us.ibm.com

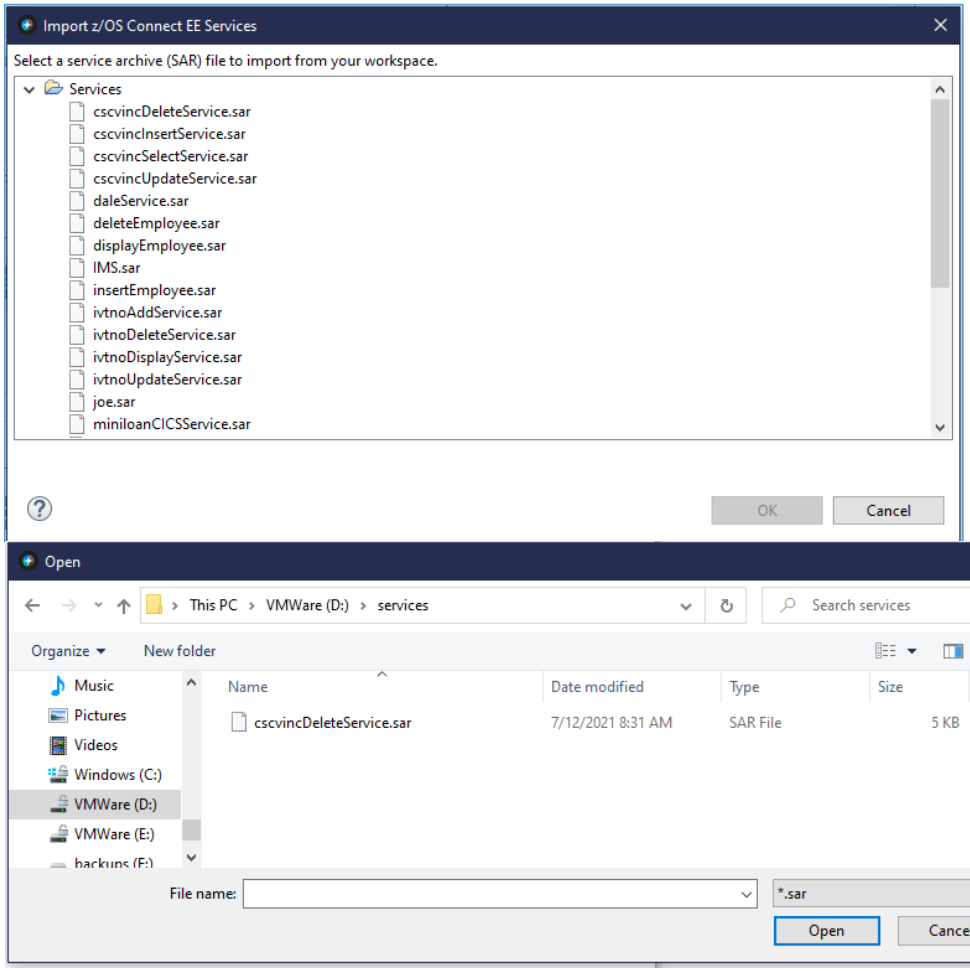
API toolkit – API Editor

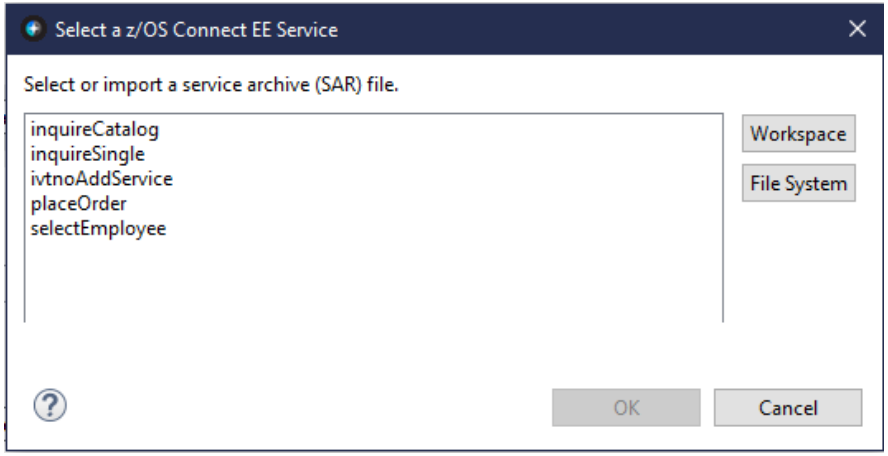




mitchj@us.ibm.com

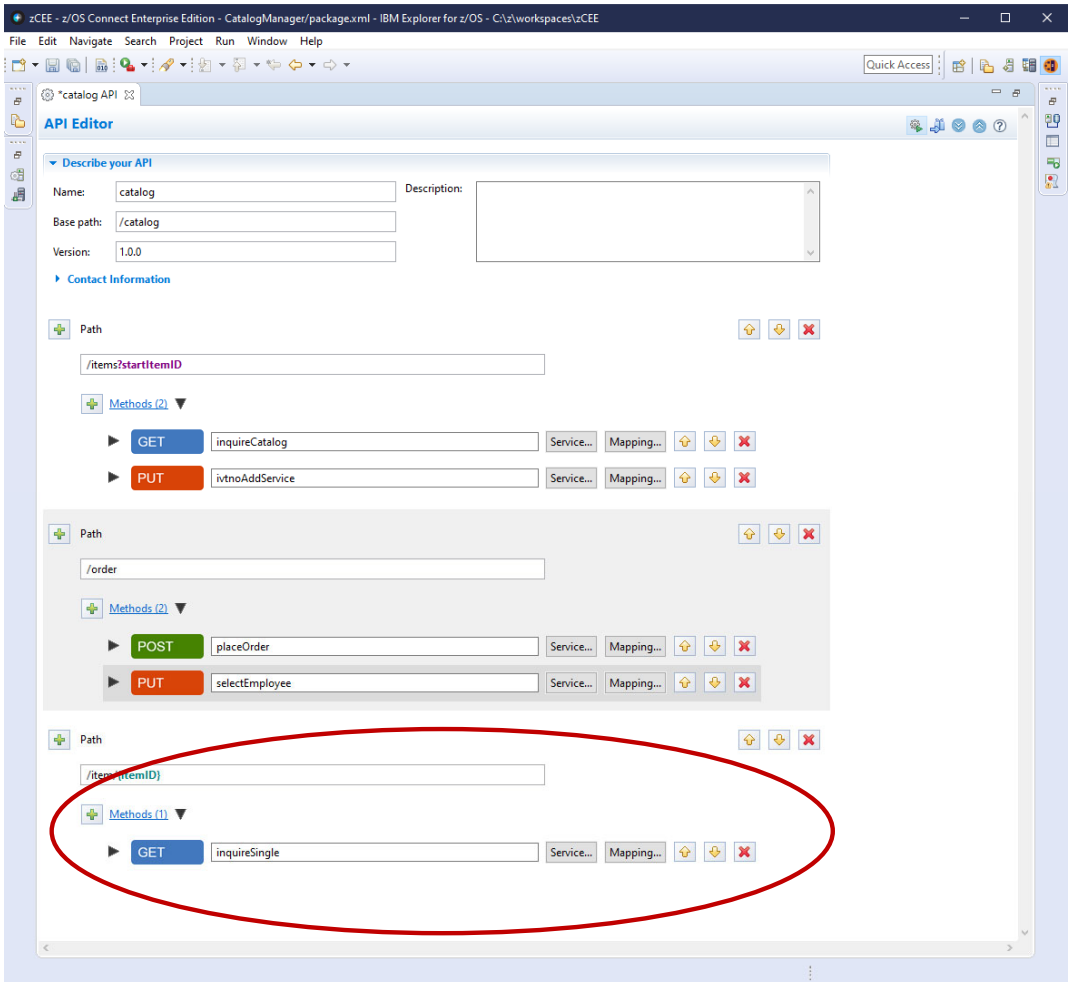
Importing the service archives files





mitchj@us.ibm.com

API toolkit – API Editor



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

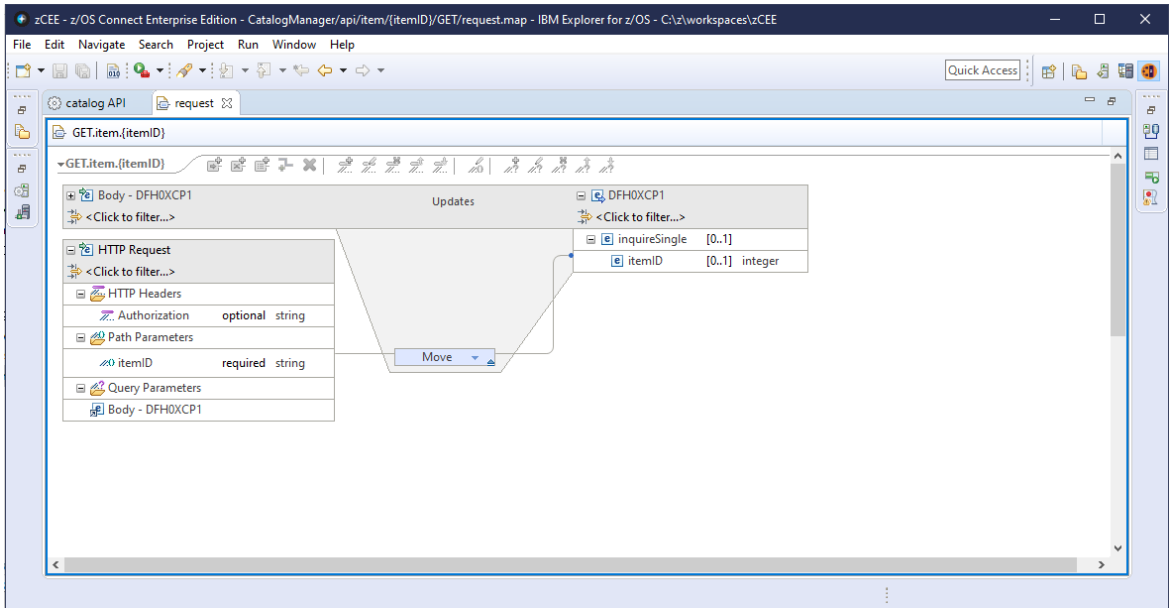
Once you define your API, you can map backend services to each request.

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

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.

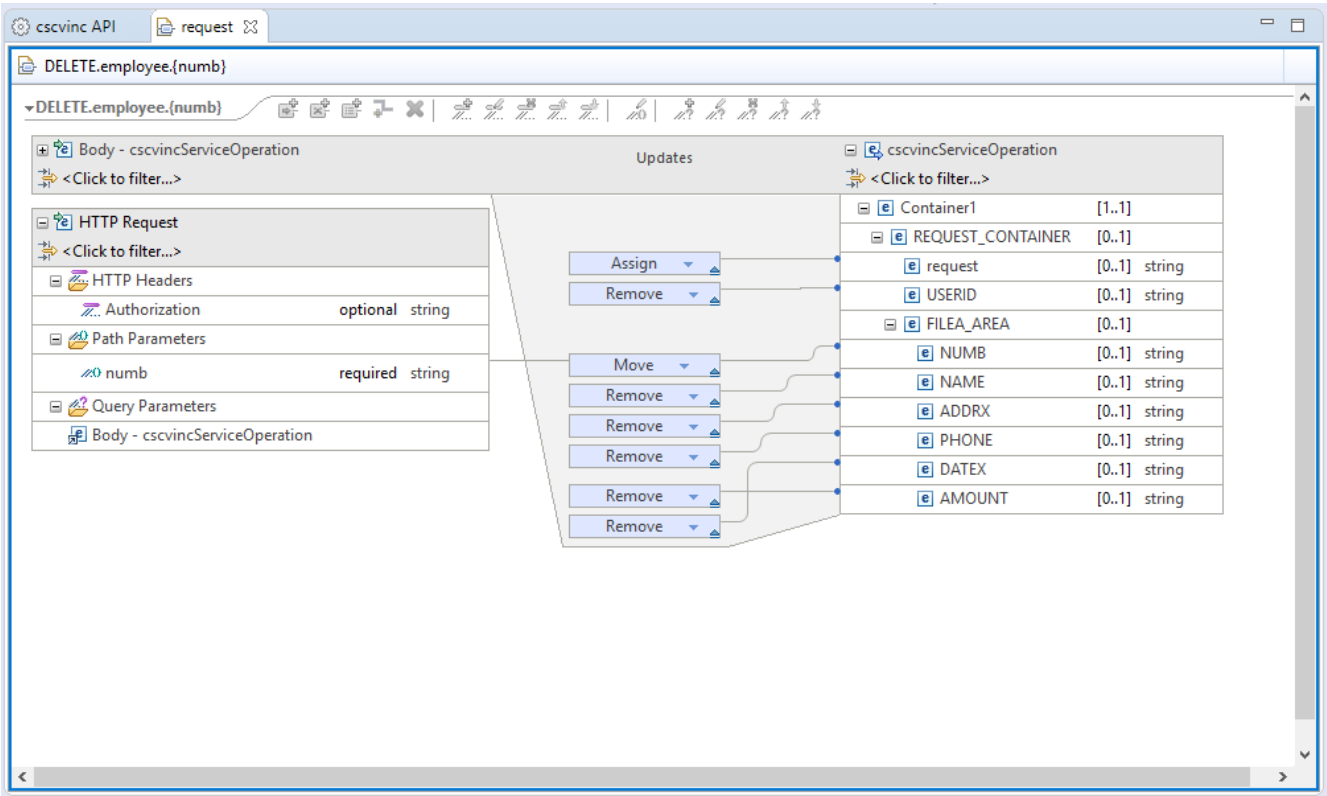
Assign static values to fields, useful for Op codes.

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

mitchj@us.ibm.com

API toolkit – API Editor

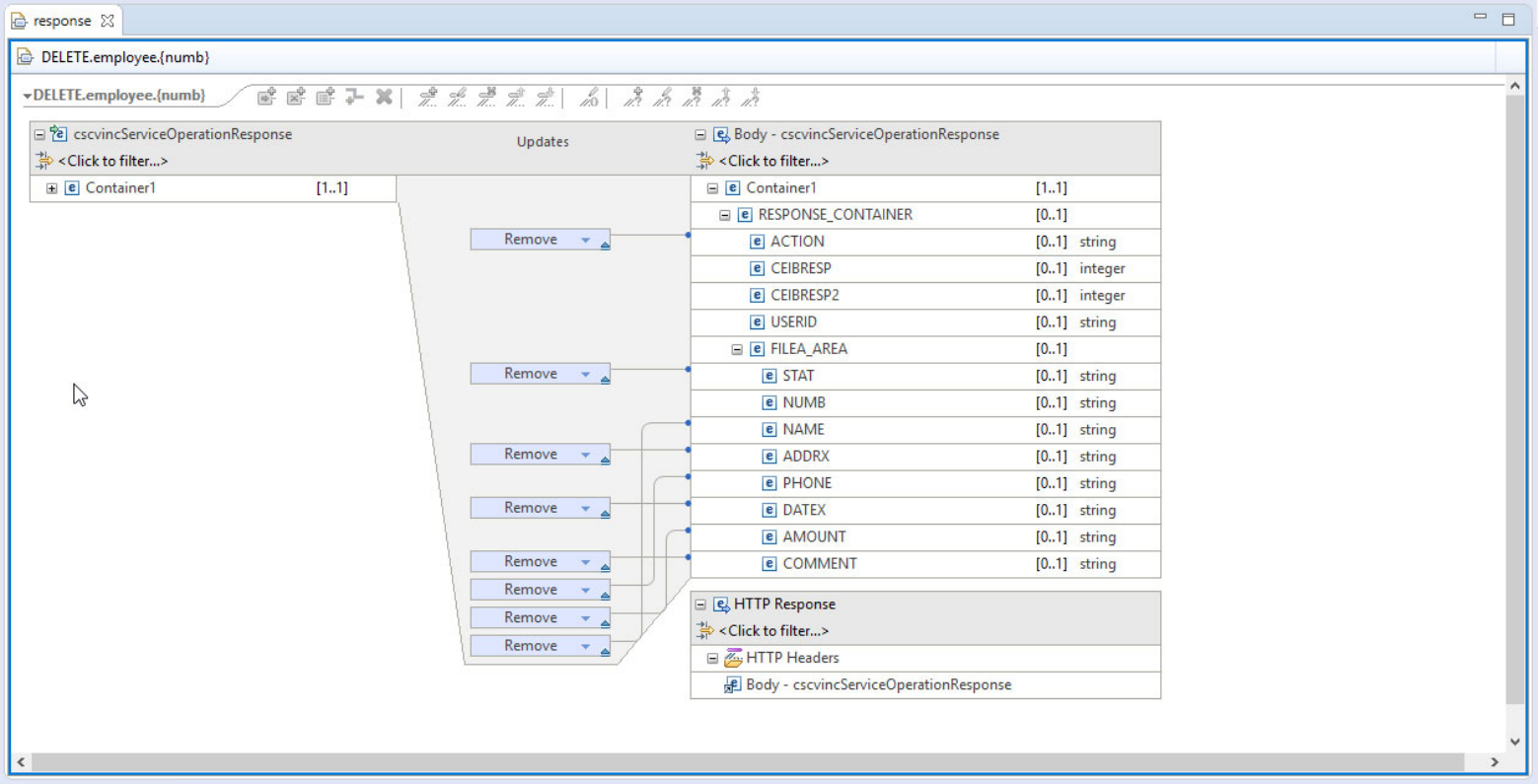
API mapping: Remove or assign values to the fields exposed by service developer



mitchj@us.ibm.com

API toolkit – API Editor

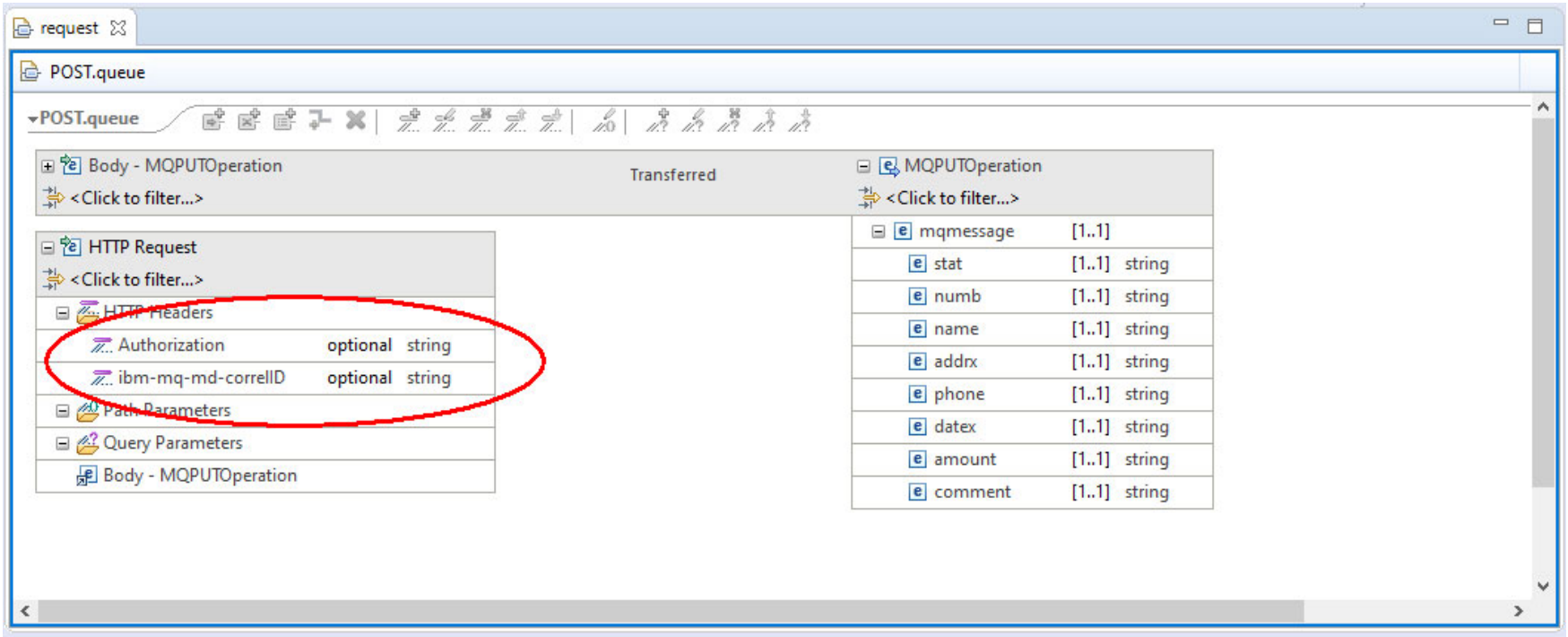
API mapping: Allows the API Developer to remove fields from the response to tailor the API



mitchj@us.ibm.com

API toolkit – Header properties

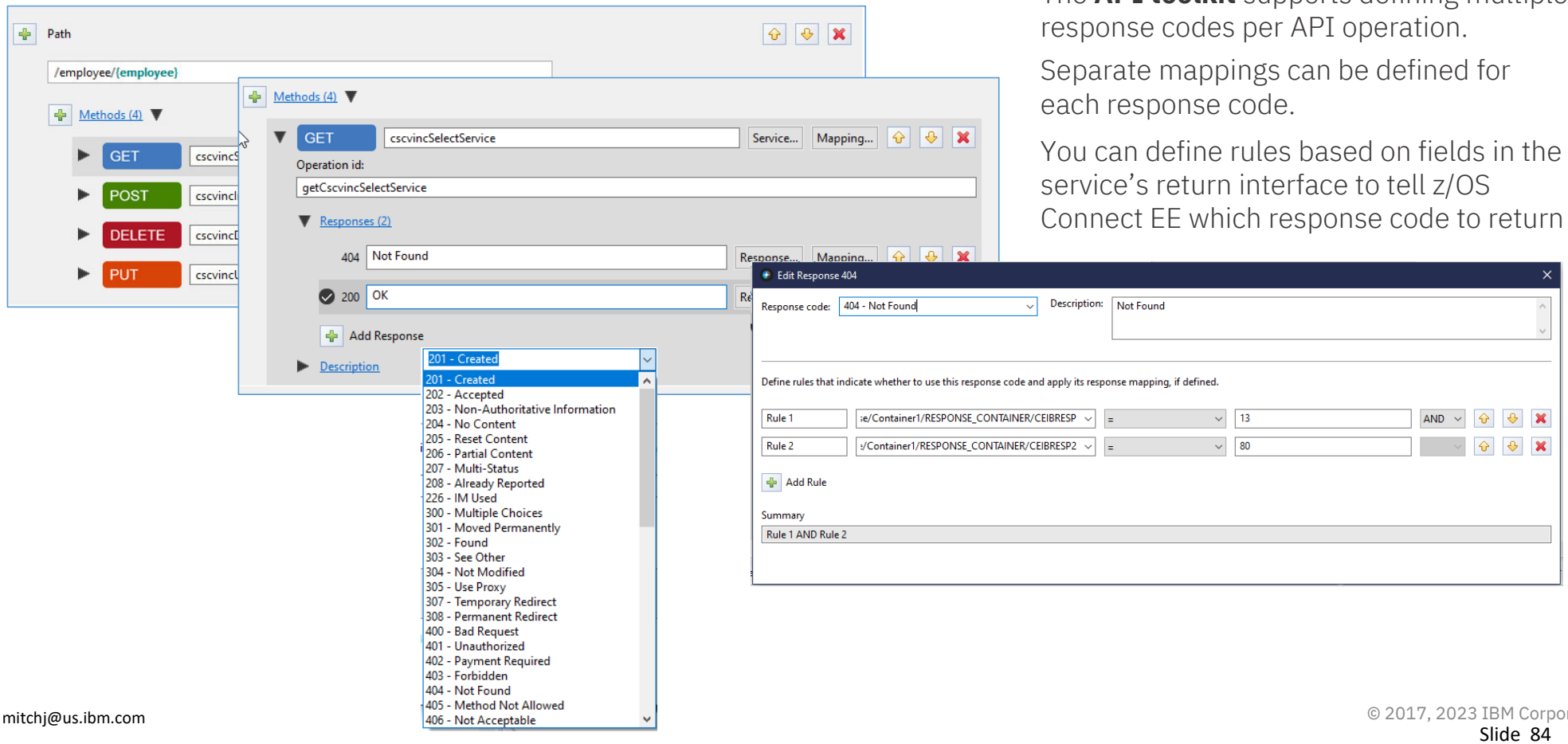
API mapping: Allows adding HTTP header properties



mitchj@us.ibm.com

API toolkit

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.

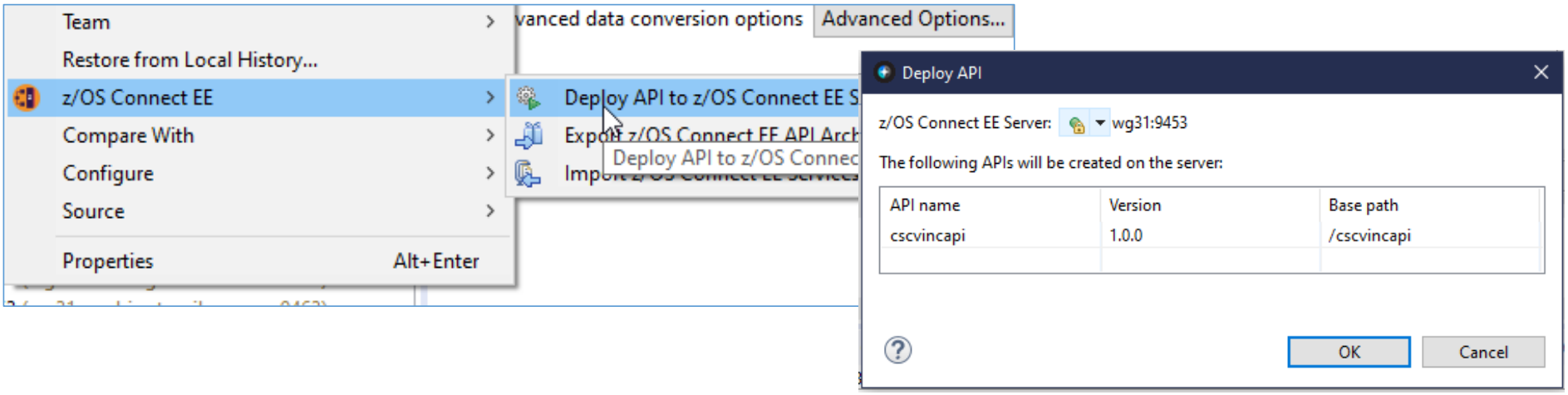
You can define rules based on fields in the service's return interface to tell z/OS Connect EE which response code to return

mitchj@us.ibm.com

API toolkit – API Editor

Server connection and API deployment

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.

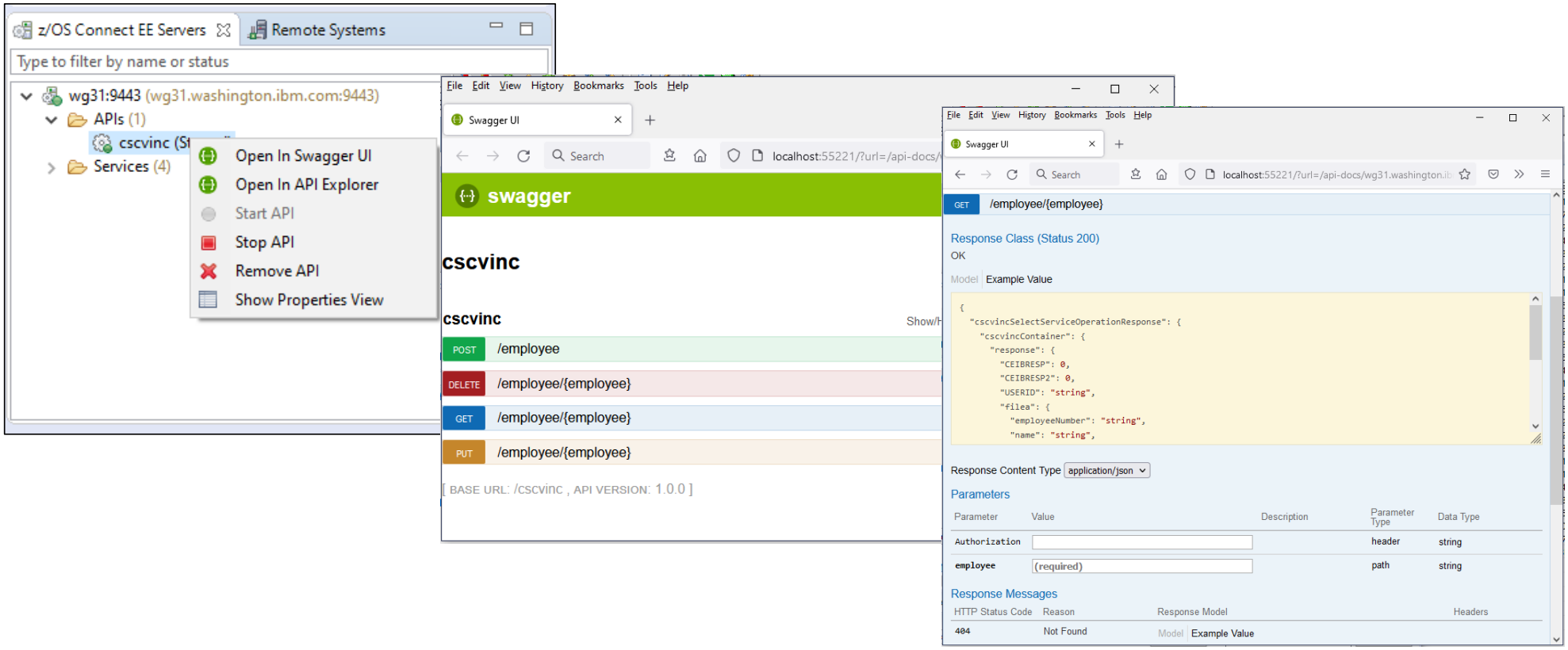
z/OS Connect EE servers view allows you to start, stop, and remove APIs from a running server.

mitchj@us.ibm.com

API toolkit – API Editor

Testing with Swagger UI

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

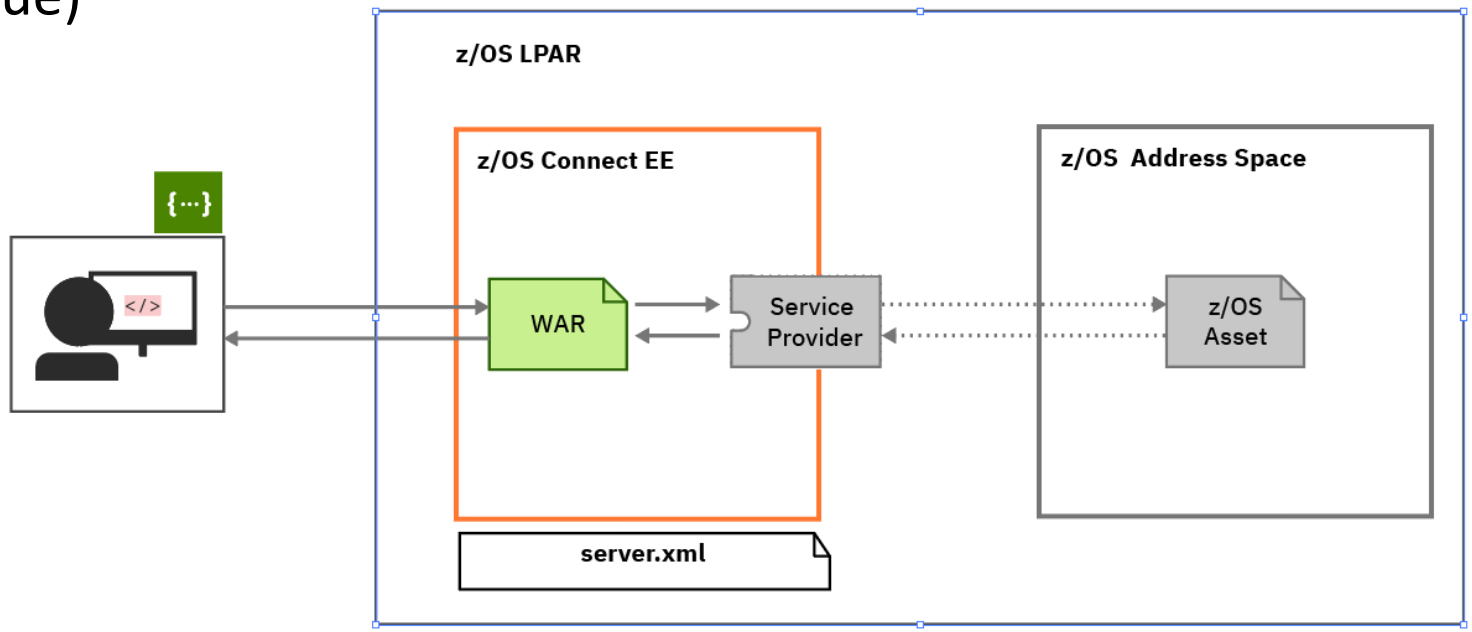
/zosConnect/designer

Simple API Development using the z/OS Connect Designer

mitchj@us.ibm.com

Accessing the CICS or Db2 asset (Open API 3)

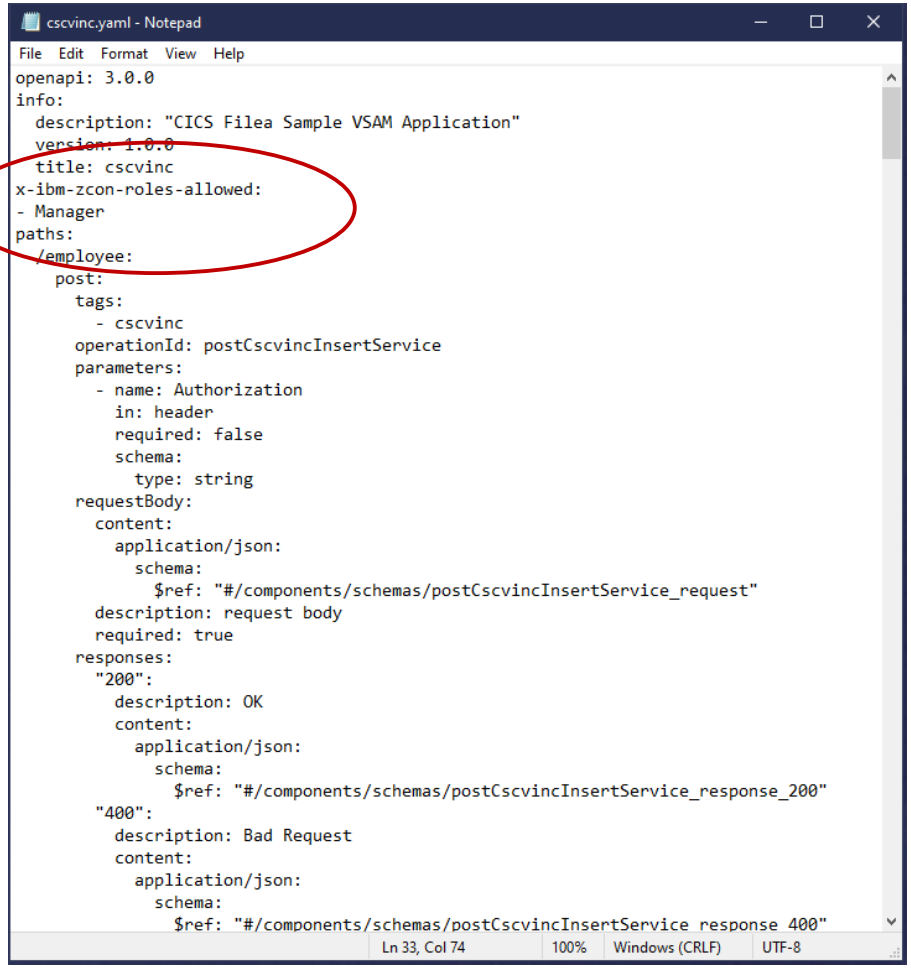
- z/OS Connect OpenAPI 3 APIs are developed using a z/OS Connect Designer web browser tool to developer Web ARchive (WAR) files (a traditional Java packaging technique)



- The WAR provides the RESTful interface is ready to be consumed by a client and it requires the client to have no knowledge that a z/OS resource is being accessed as well as the mapping and transformation for accessingg the resource.
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port)

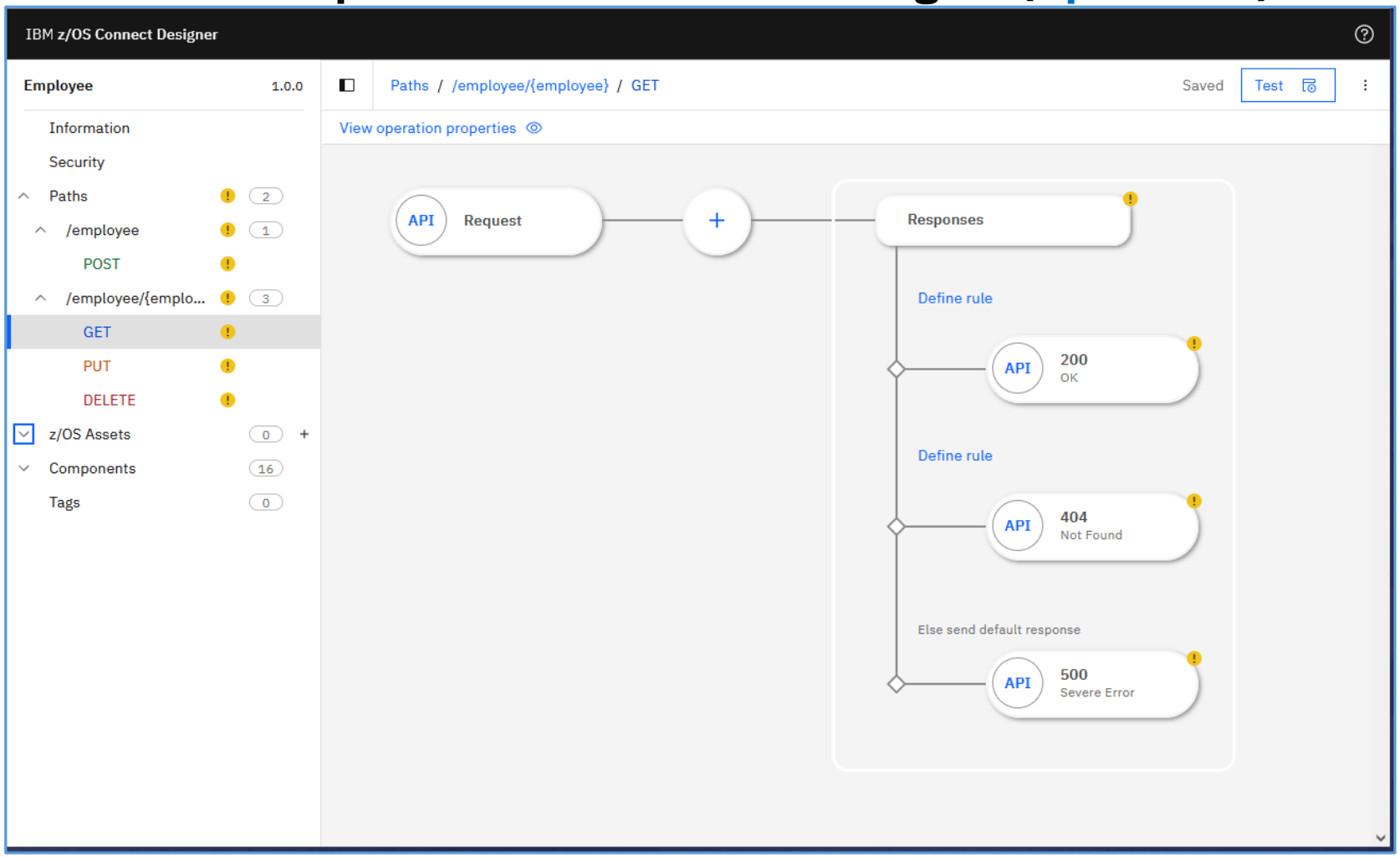
mitchj@us.ibm.com

Accessing a z/OS asset starting with a YAML description of an API (OpenAPI 3)



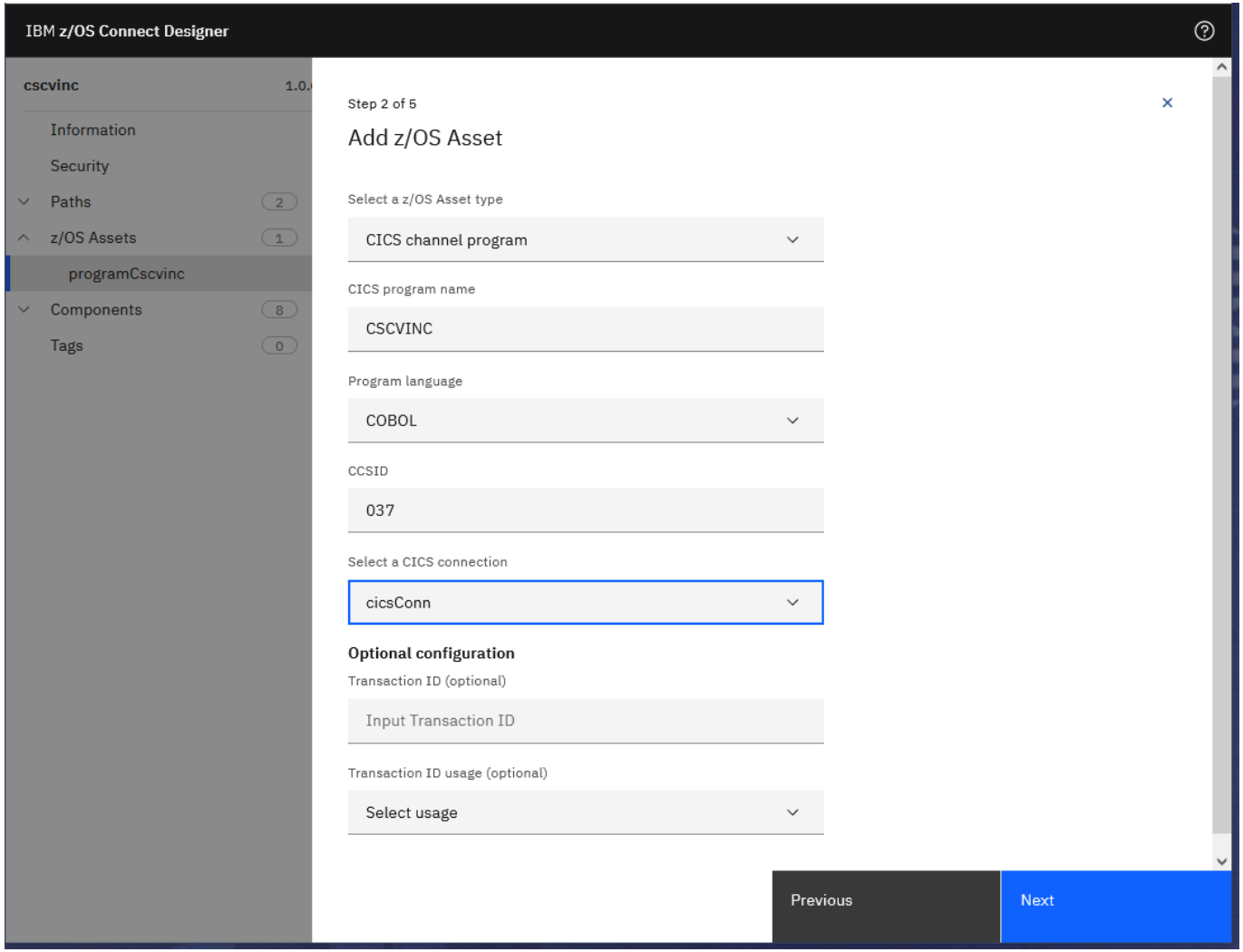
mitchj@us.ibm.com

Import the YAML description of an API into the Designer (OpenAPI 3)



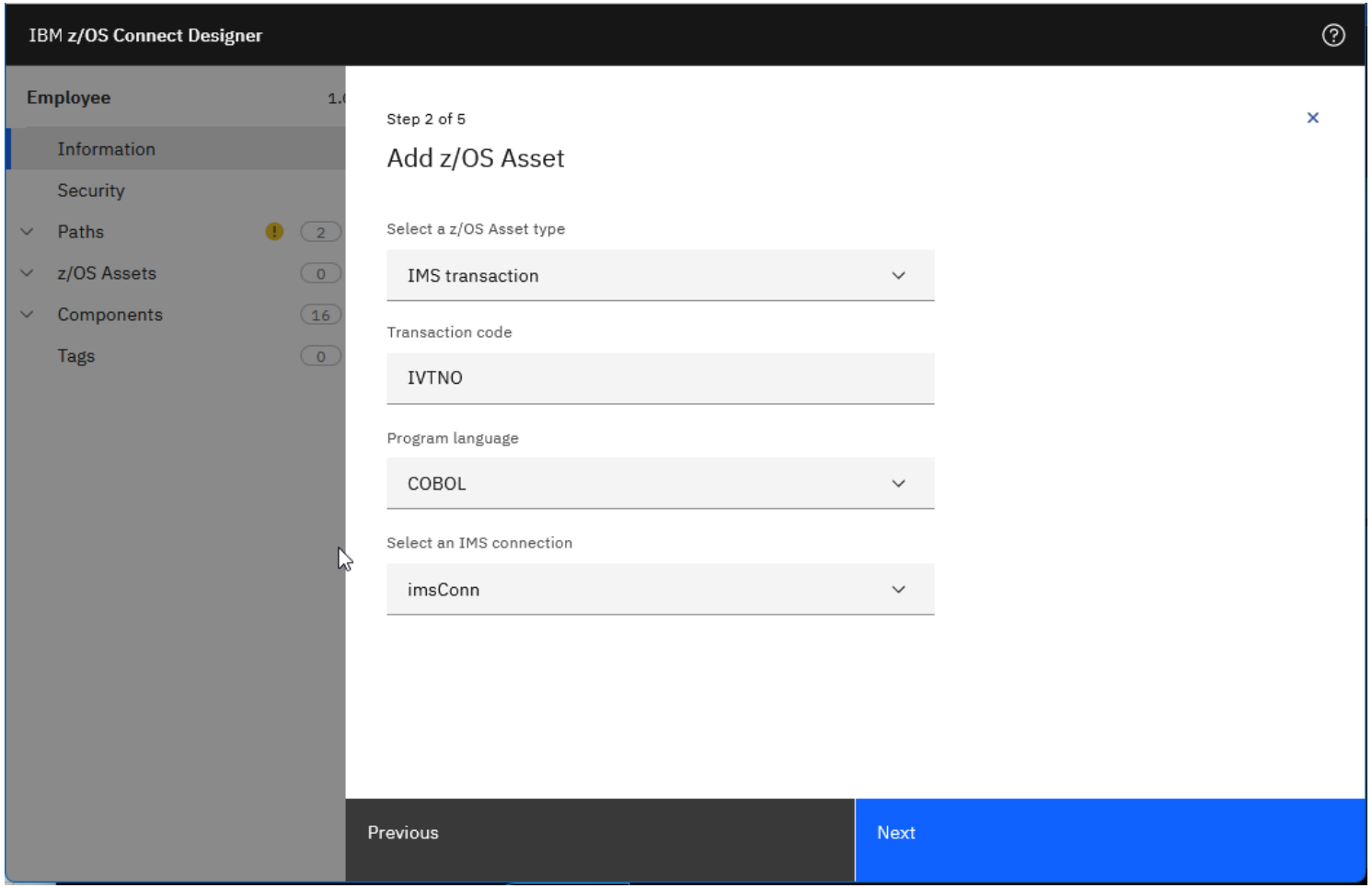
mitchj@us.ibm.com

Describe the z/OS asset, accessing a CICS program (OpenAPI 3)



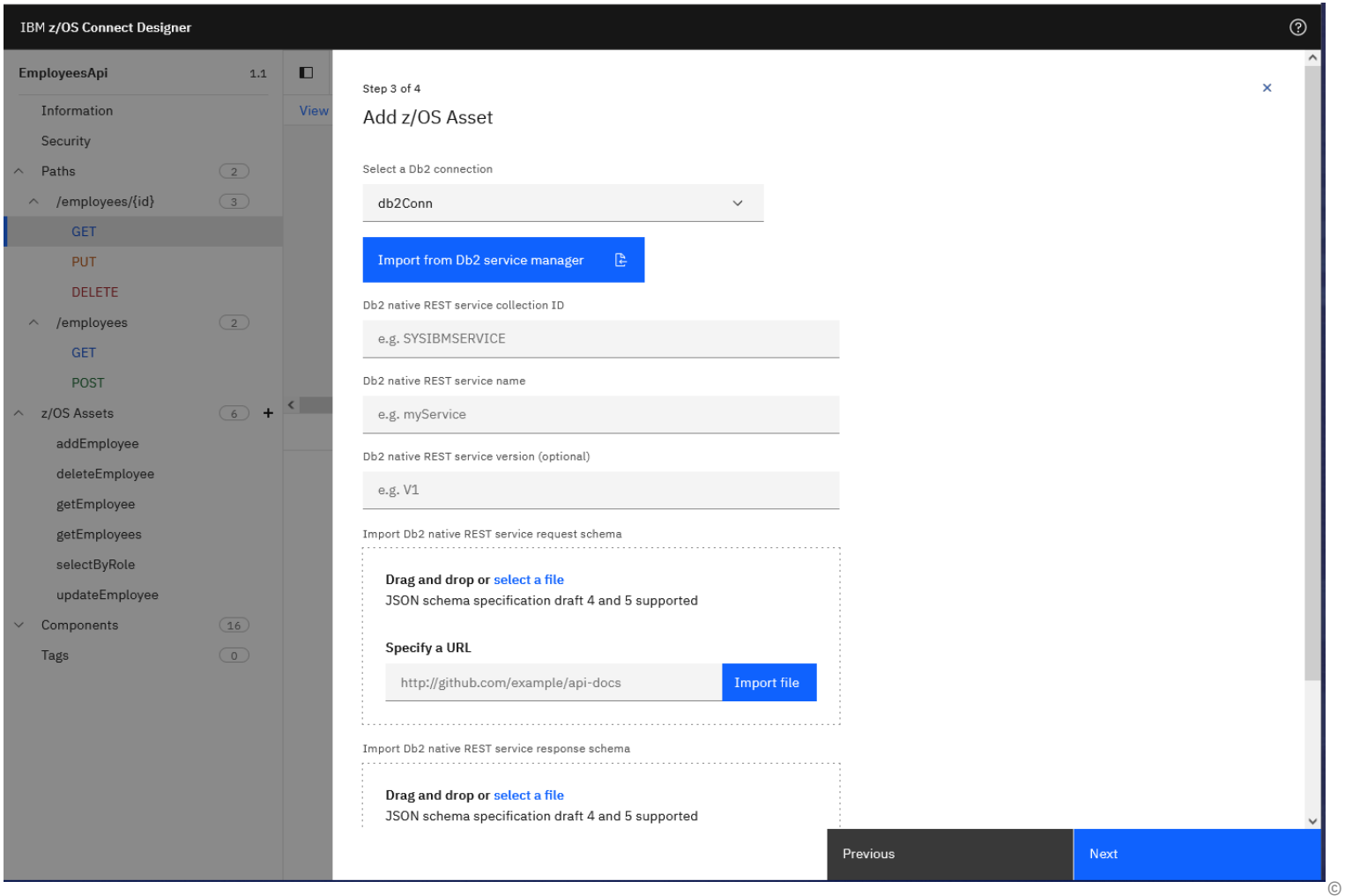
mitchj@us.ibm.com

Or accessing a z/OS IMS transaction (OpenAPI 3)



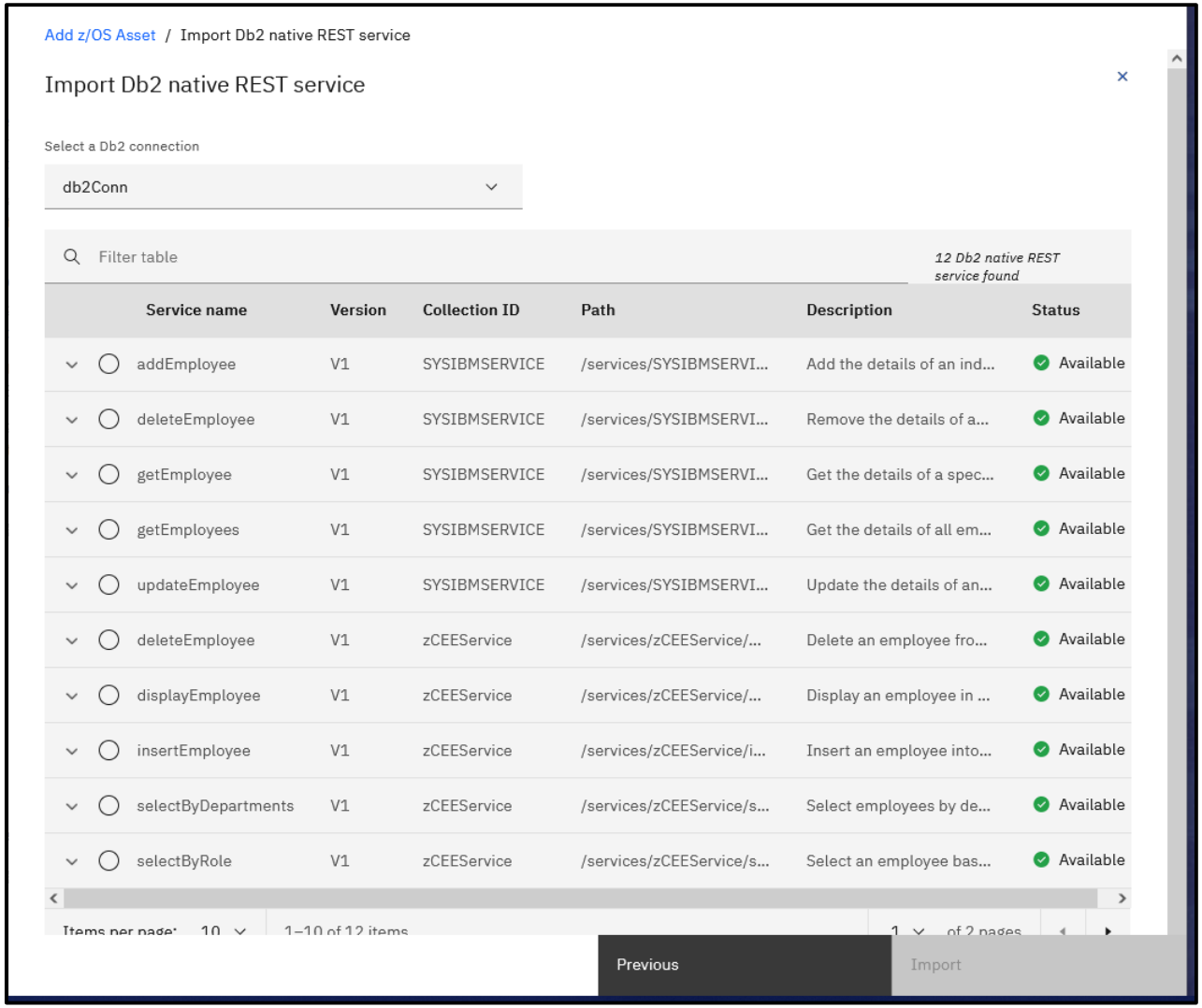
mitchj@us.ibm.com

Or accessing a Db2 REST service (OpenAPI 3)



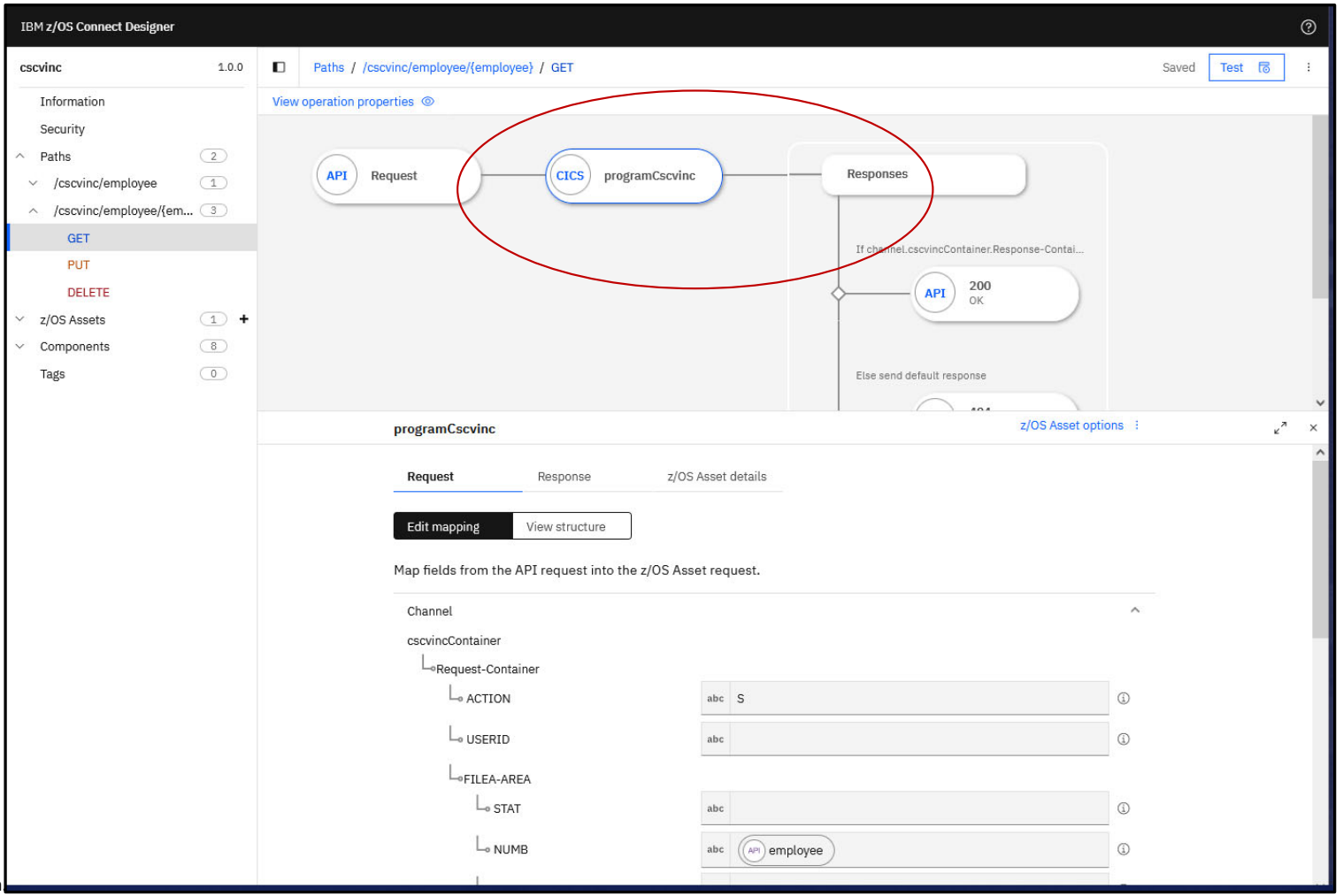
mitchj@us.ibm.com

Select the z/OS Db2 REST Service (OpenAPI 3)

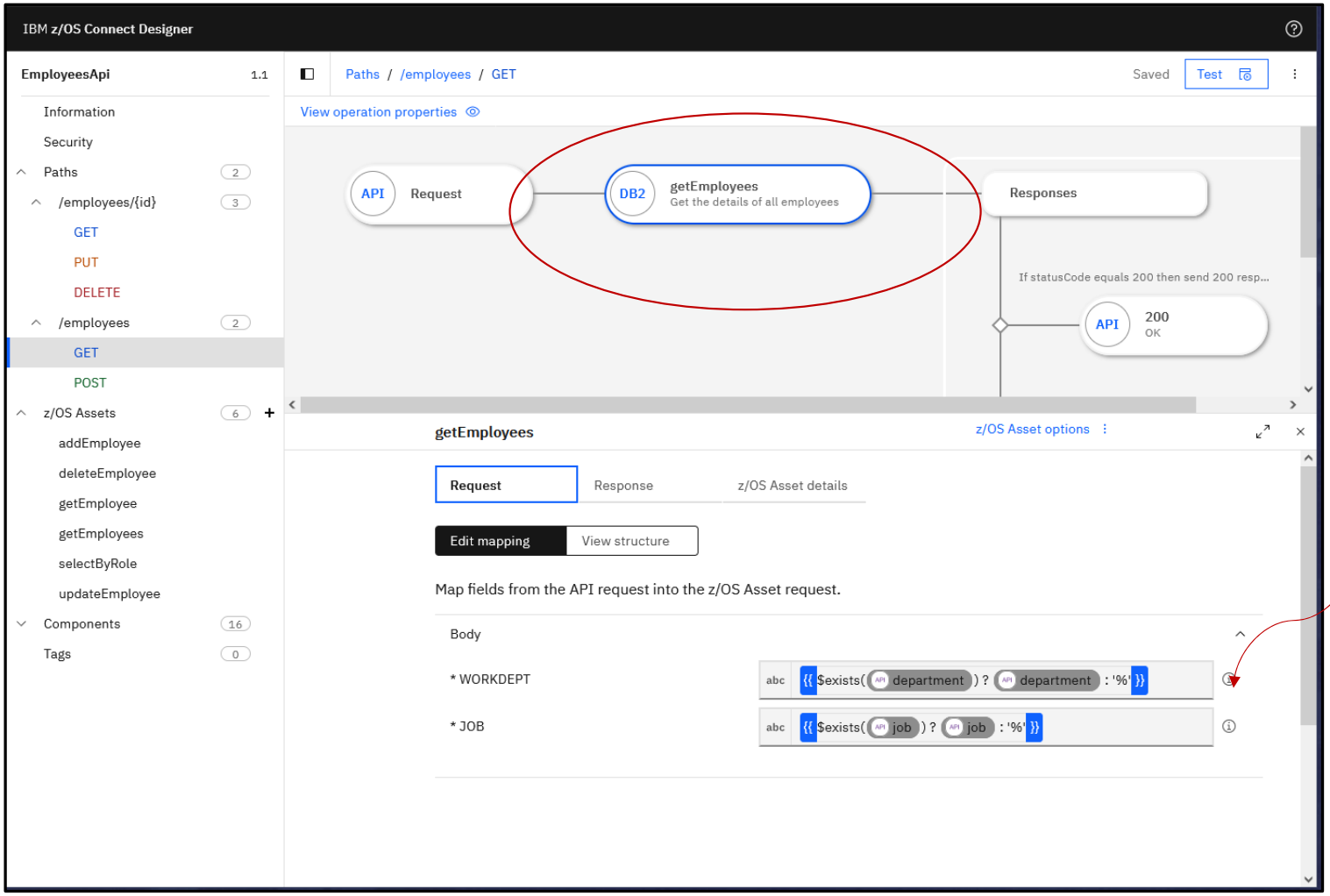


mitchj@us.ibm.com

Map the method and request message with a serialized message (OpenAPI 3)



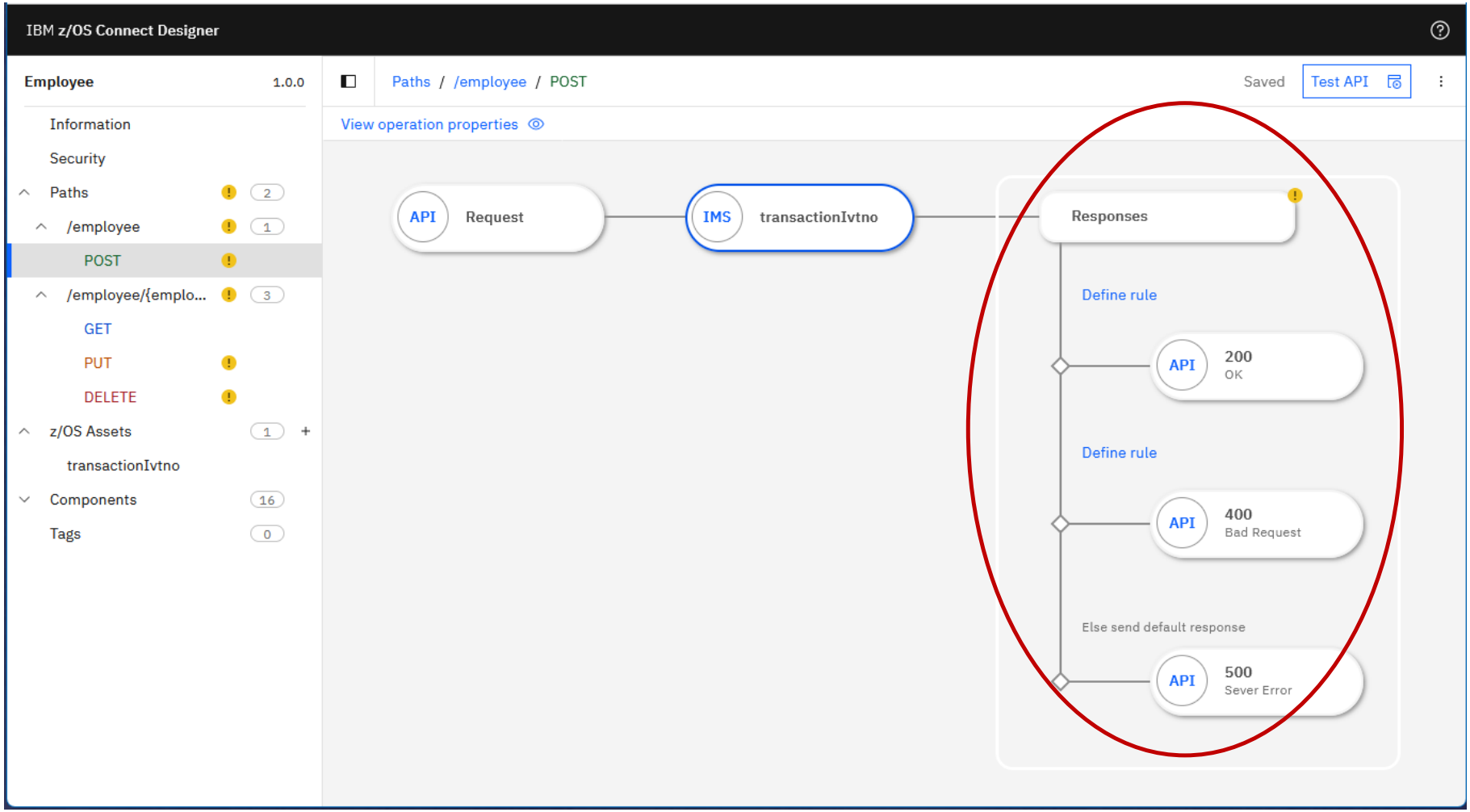
Map the method and the response message with a Db2 REST service (OpenAPI 3)



JSONata example

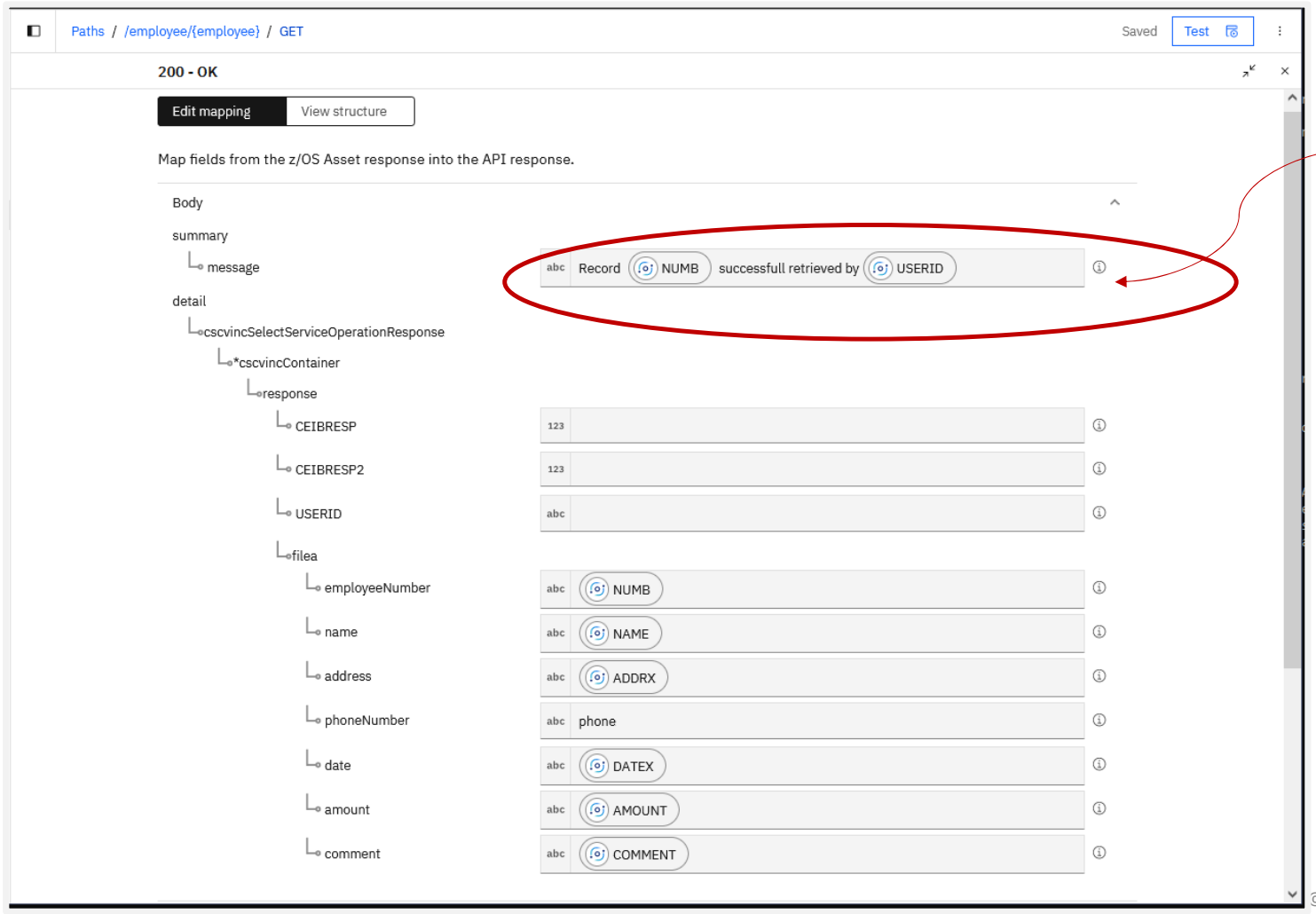
mitchj@us.ibm.com

Map results to the specification's responses (OpenAPI 3)



mitchj@us.ibm.com

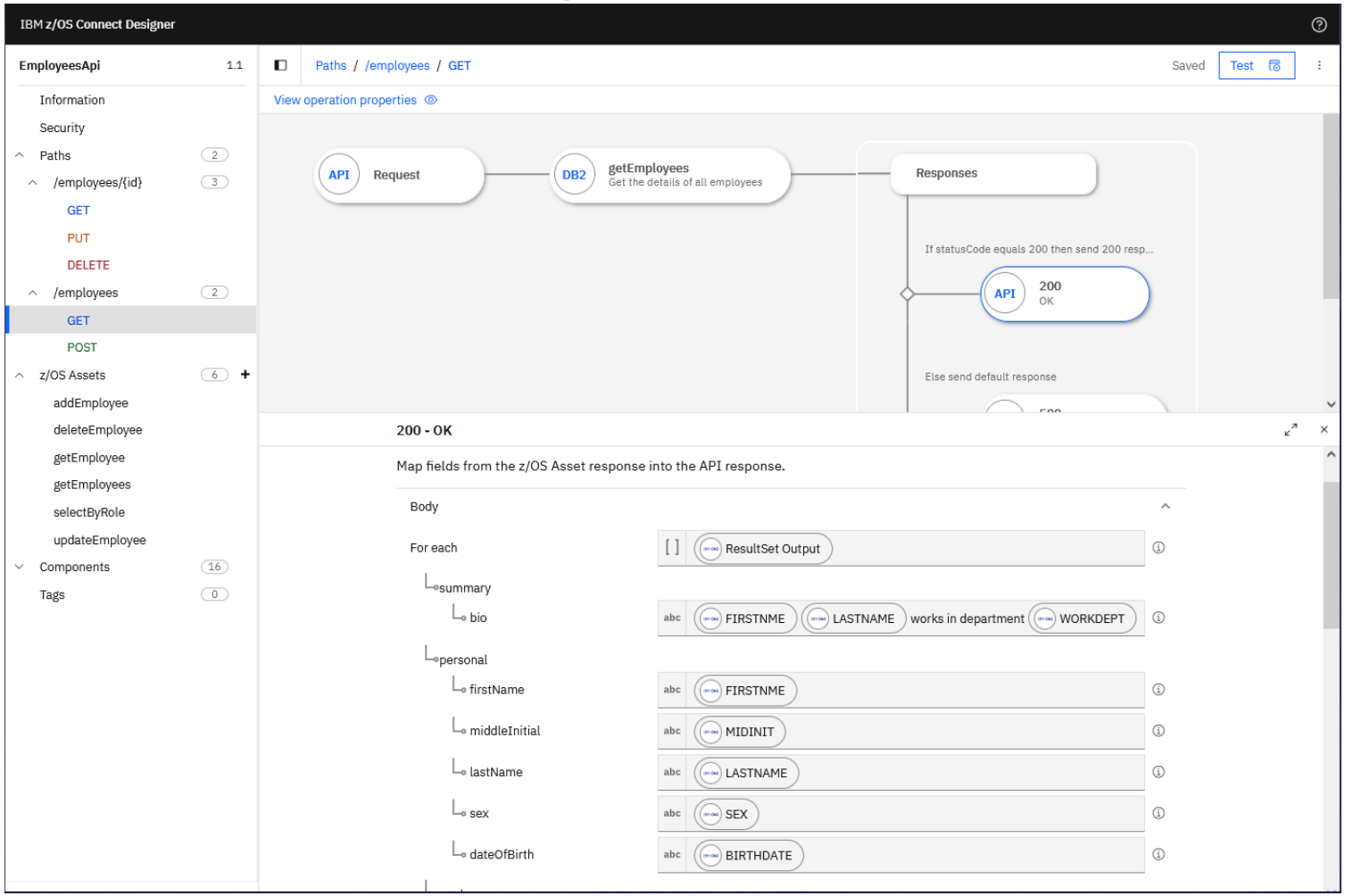
Map the serialzied response message (OpenAPI 3)



JSONata example

mitchj@us.ibm.com

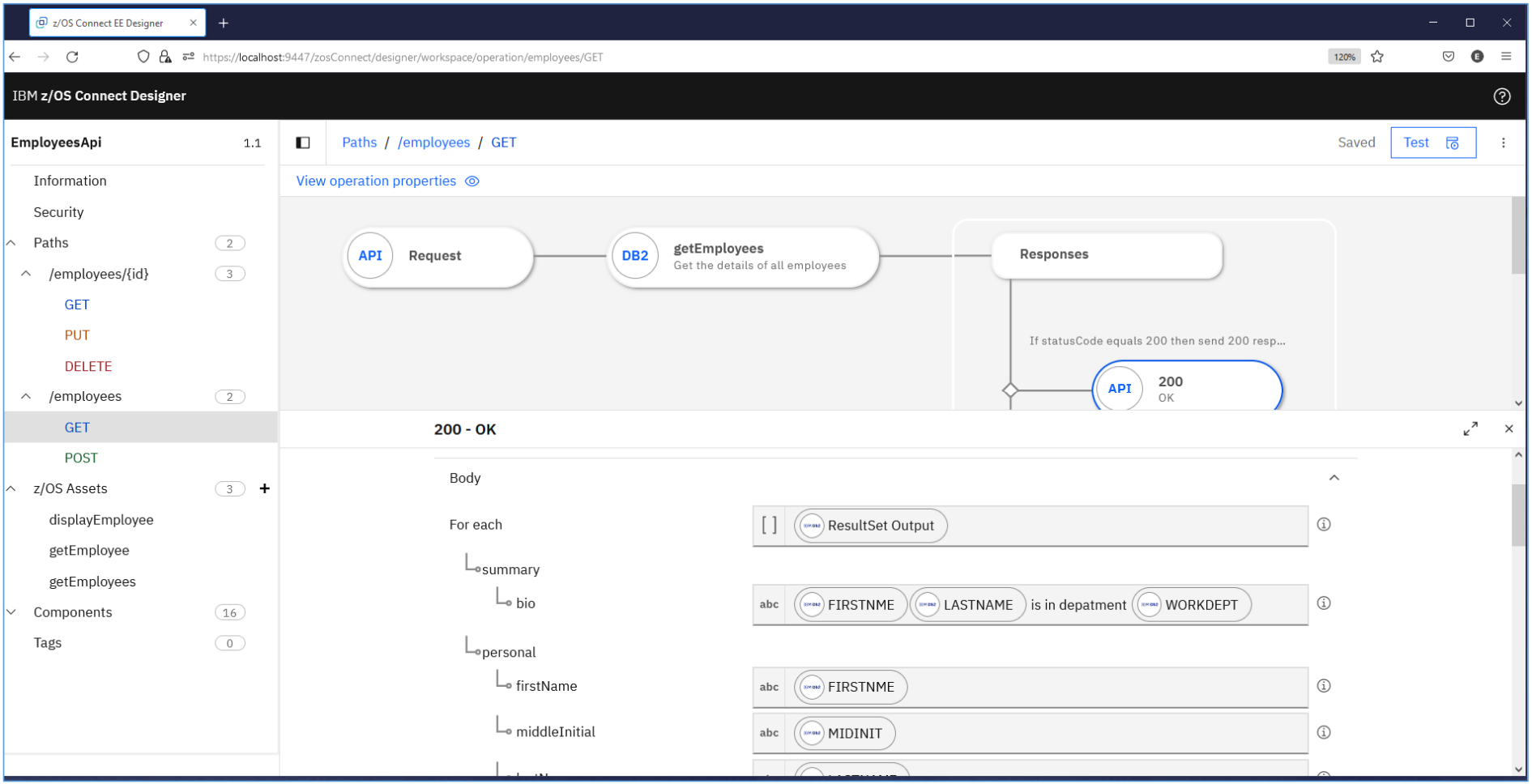
Map the response message (OpenAPI 3)



ResutSet Output is a list or array of results and the "For each" processes each element in the list.

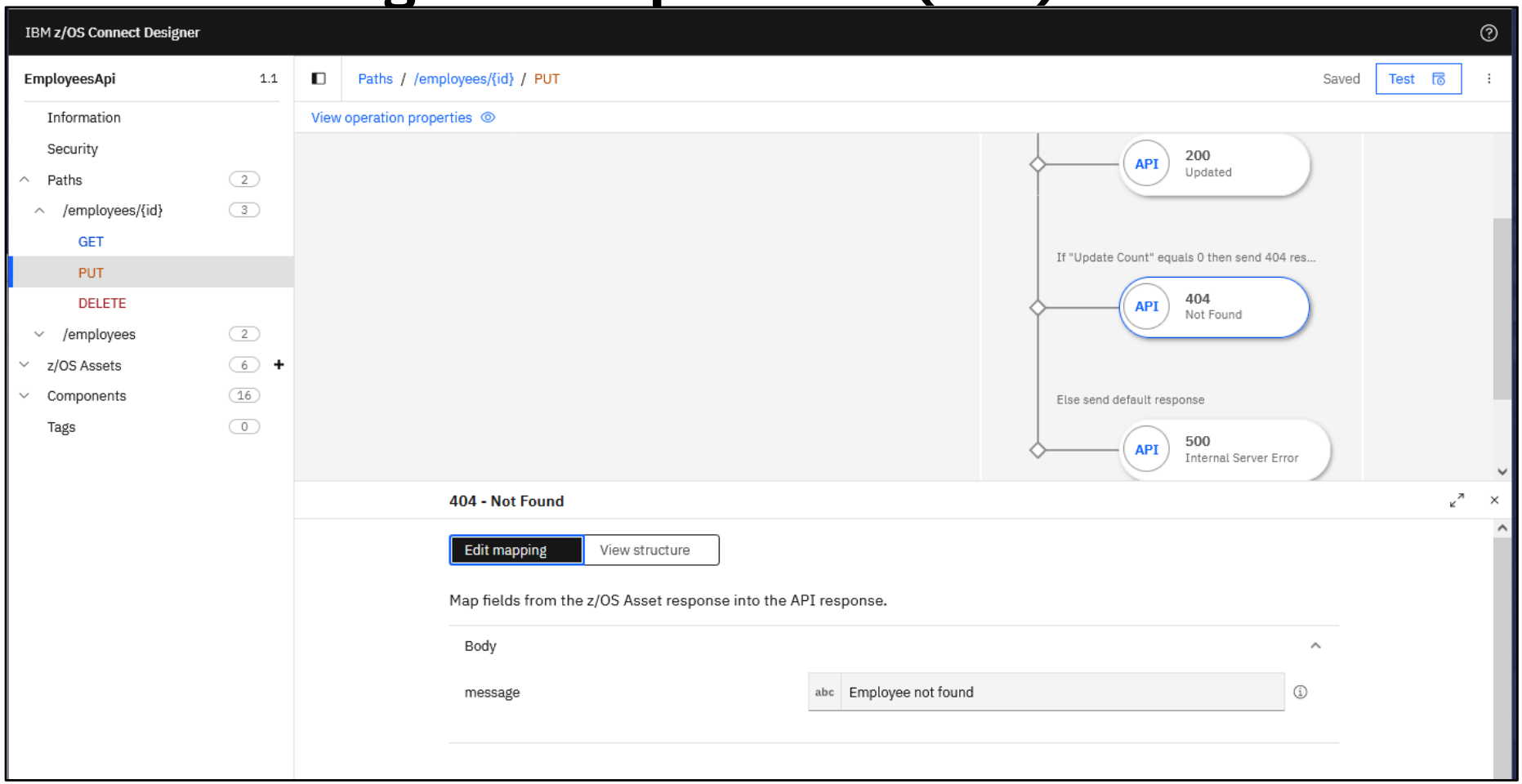
mitchj@us.ibm.com

z/OS Connect Designer for OpenAPI 3 (200)



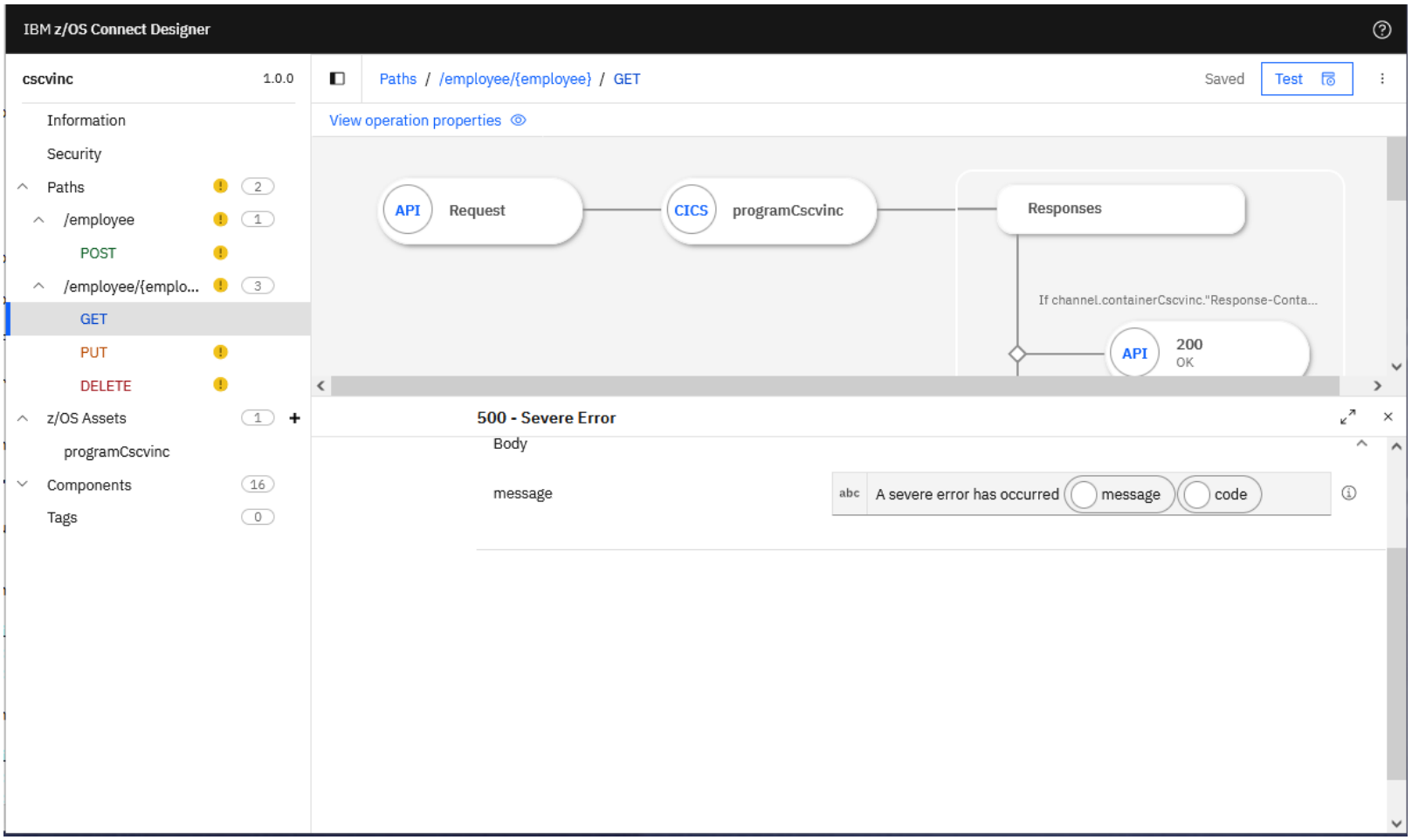
mitchj@us.ibm.com

z/OS Connect Designer for OpenAPI 3 (404)



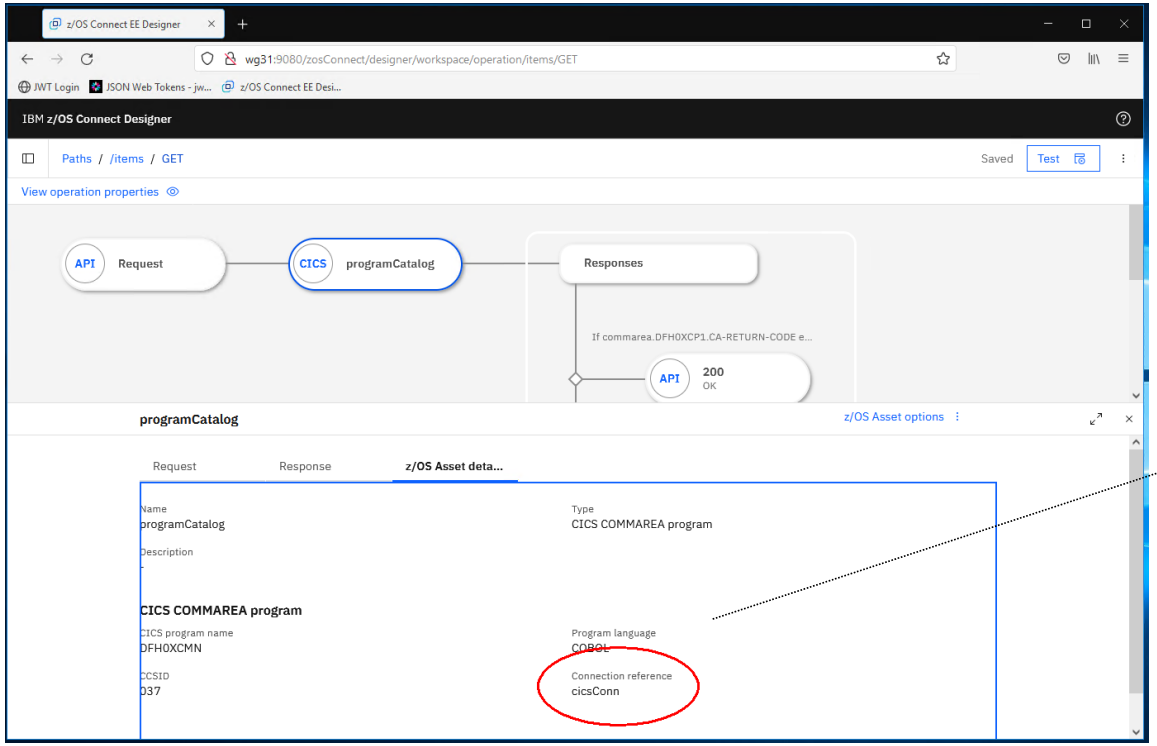
mitchj@us.ibm.com

z/OS Connect Designer for OpenAPI 3 (500)



mitchj@us.ibm.com

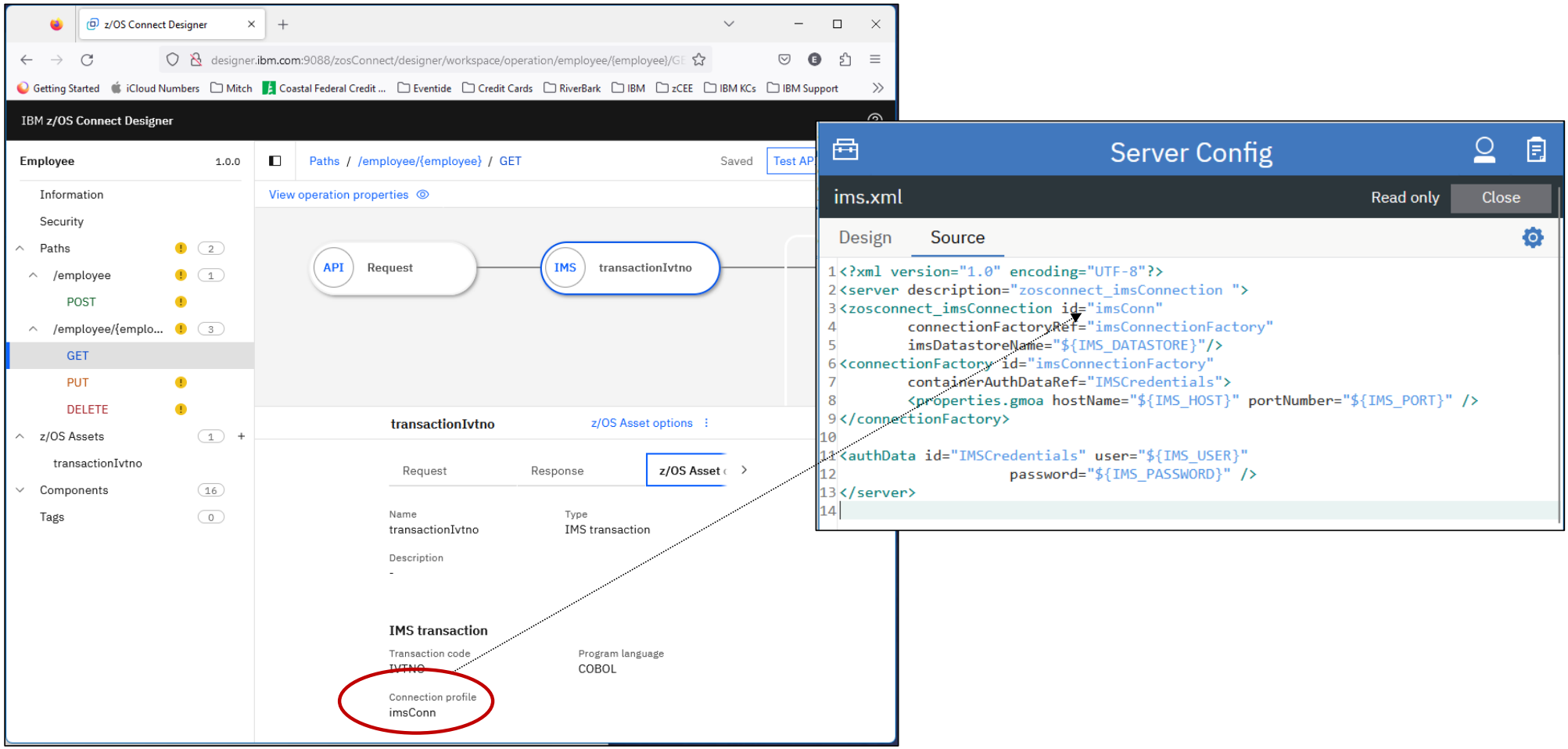
Server XML - Accessing a CICS program using IPIC (OpenAPI 3)



The connection references identifies a zosconnect_cicsIpicConnection configuration element.
Which provides the connection details to a CICS region.

mitchj@us.ibm.com

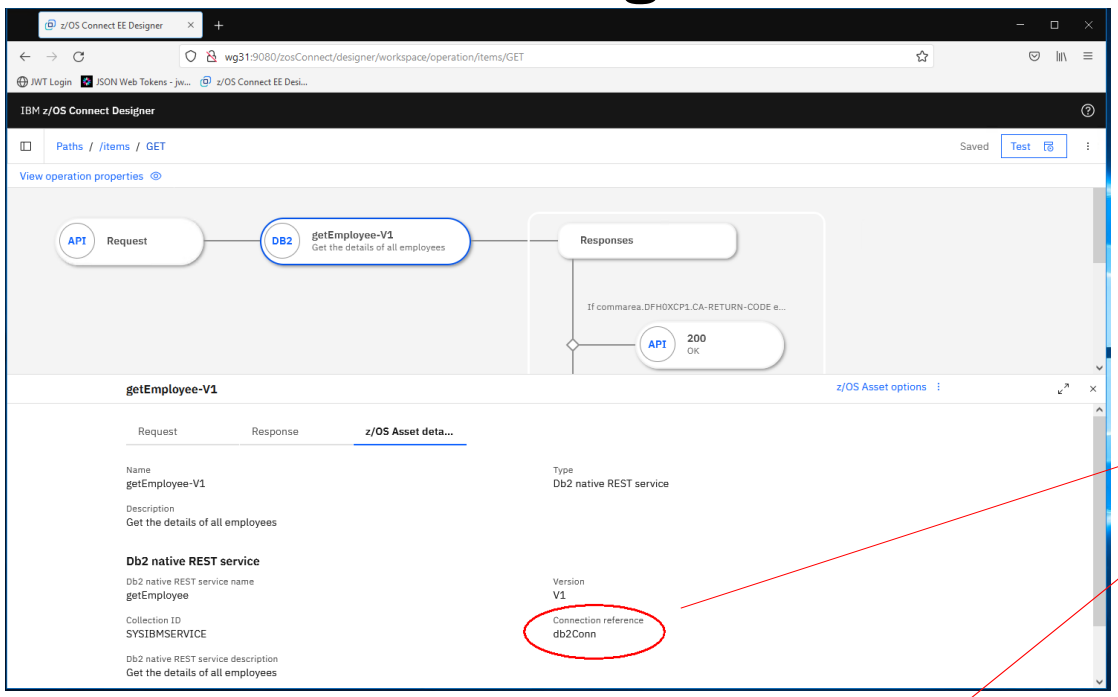
Server XML - Accessing an IMS transaction(OpenAPI 3)

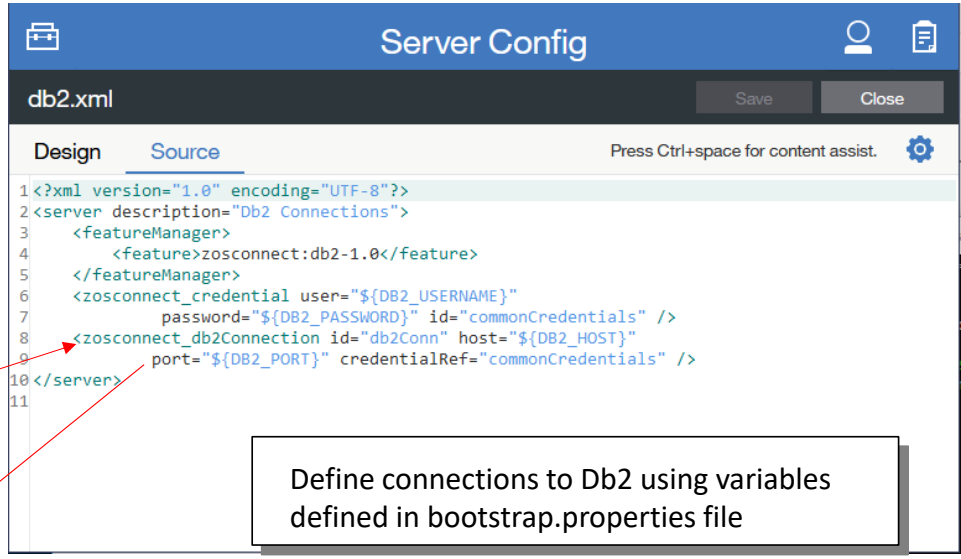


The connection references identifies a zosconnect_imsConnection element. Which provides the connection details to IMS.

mitchj@us.ibm.com

Server XML - Accessing a Db2 REST service (OpenAPI 3)





Define connections to Db2 using variables defined in bootstrap.properties file

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

The connection references identifies a zosconnect_db2Connection configuration element. Which provides the connection details to a DB2 DDF task.

mitchj@us.ibm.com

EJB roles for z/OS Connect (OpenAPI 3)

```
<safCredentials unauthenticatedUser="WSGUEST" profilePrefix="BBGZDFLT" />

<webApplication id="CatalogManager" location="${server.config.dir}/apps/api.war" contextRoot="catalog"
name="CatalogManager"/>

<safRoleMapper profilePattern=%profilePrefix%.%resourceName%.%role%
```

```
openapi: 3.0.0
. . .
servers:
- url: /
x-ibm-zcon-roles-allowed:
- Manager
. . .
paths:
  /items:
    get:
     operationId: itemsGet
     . . .
  /items/{id}:
    get:
     . . .
     operationId: itemsIdGet
     x-ibm-zcon-roles-allowed:
        - Staff
 /orders:
    post:
     . . .
     operationId: ordersPost
     x-ibm-zcon-roles-allowed:
       - Staff
```

*From the OpenApi document, the value for %role% would be either **Manager** or **Staff**.*

So, the required SAF EJB roles to be defined would be:

- *BBGZDFLT.CatalogManager.**Manager***
- *BBGZDFLT.CatalogManager.**Staff***

```
REDFINE EJBROLE BBGZDFLT.CatalogManager.Manager
REDFINE EJBROLE BBGZDFLT.CatalogManager.Staff
```

Access to use the GET method to invoke /items would require read access to EJB role *BBGZDFLT.CatalogManager.Manager.*
Access to use the GET method to invoke /items/{id} and the POST method to invoke /orders would require read access to EJB role *BBGZDFLT.CatalogManager.Staff.*

mitchj@us.ibm.com

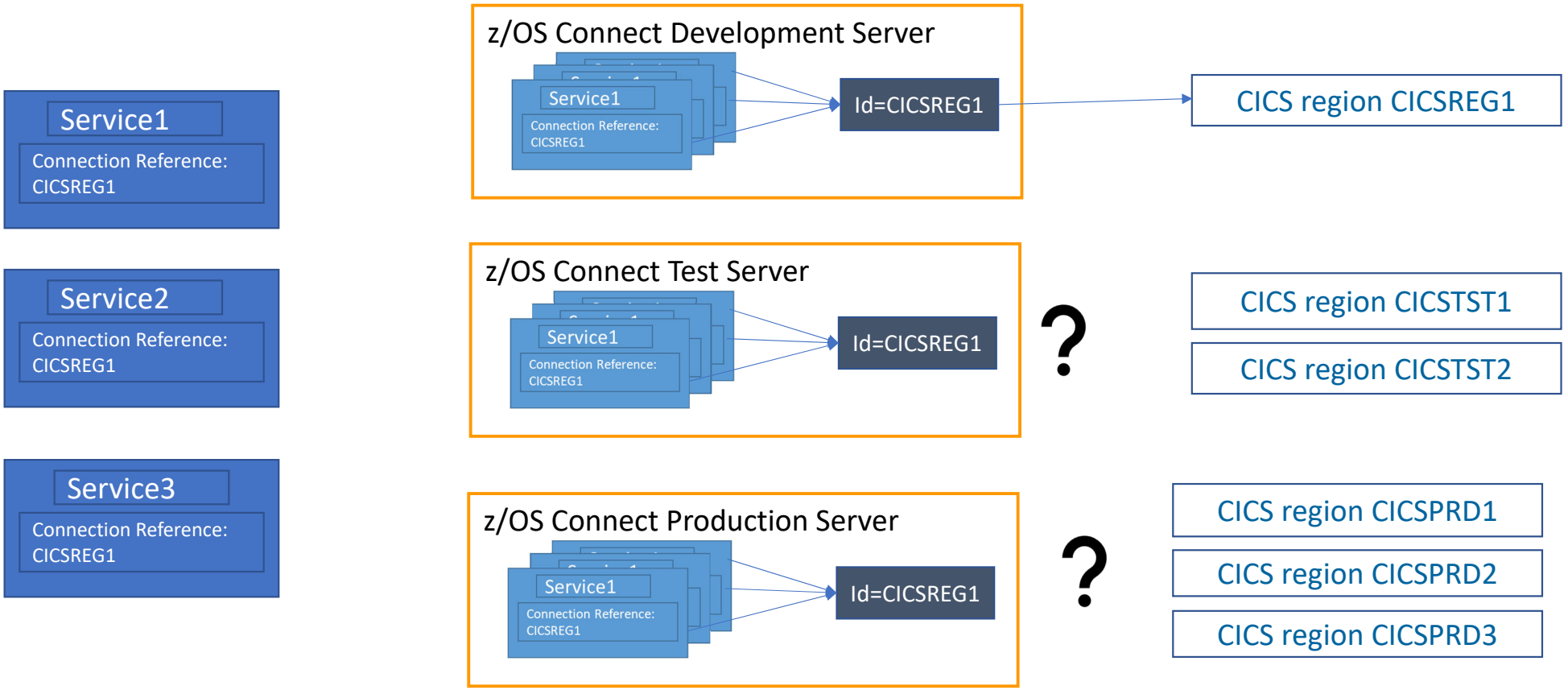
Connection Reference Consideration

Carefully consider the names used for connections

mitchj@us.ibm.com

Use naming conventions for service/endpoint connection references (OpenAPI 2)

Don’t couple service and API requester connection names to specific systems or endpoints

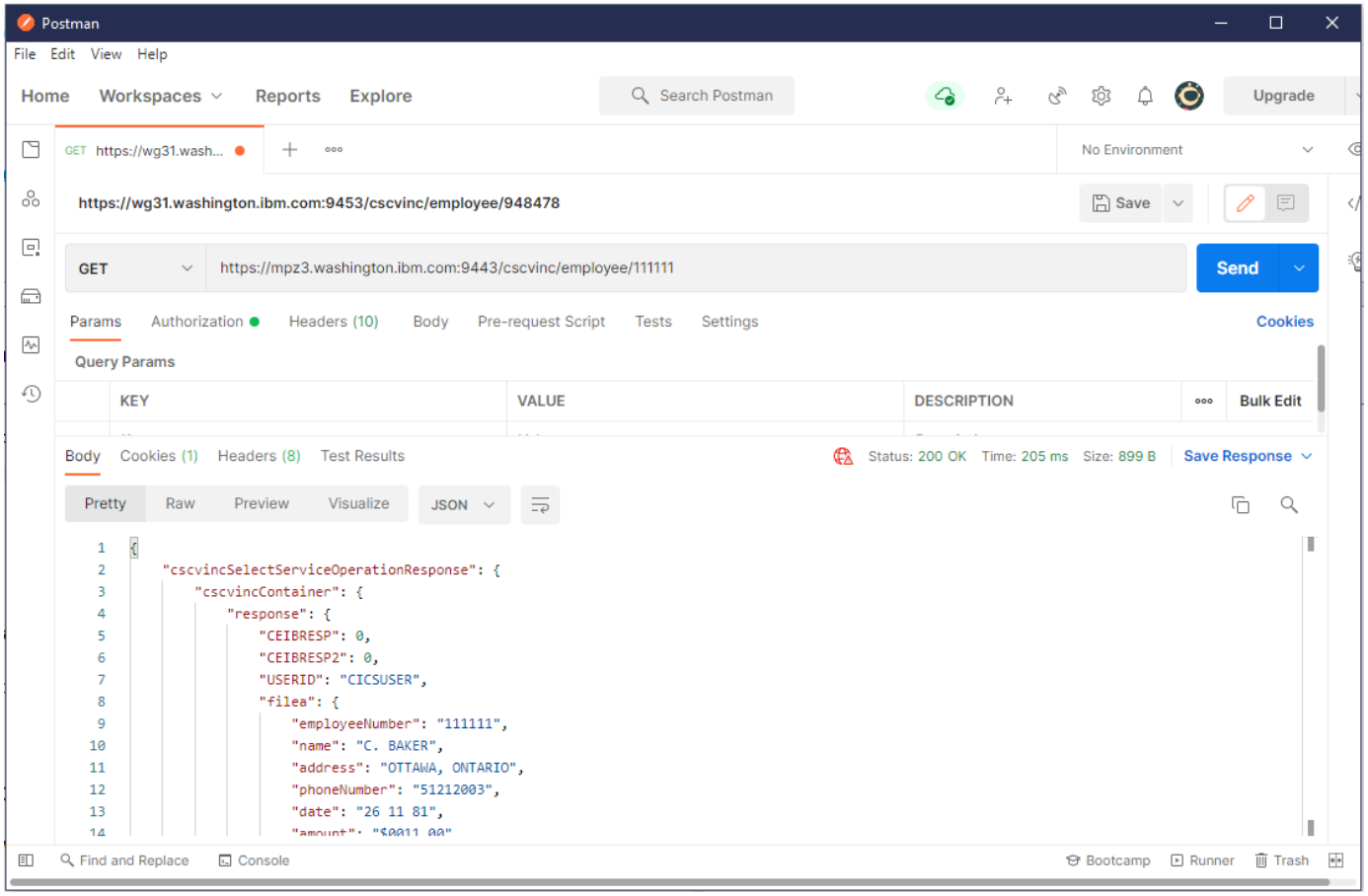


mitchj@us.ibm.com

What REST test tooling is available?

mitchj@us.ibm.com

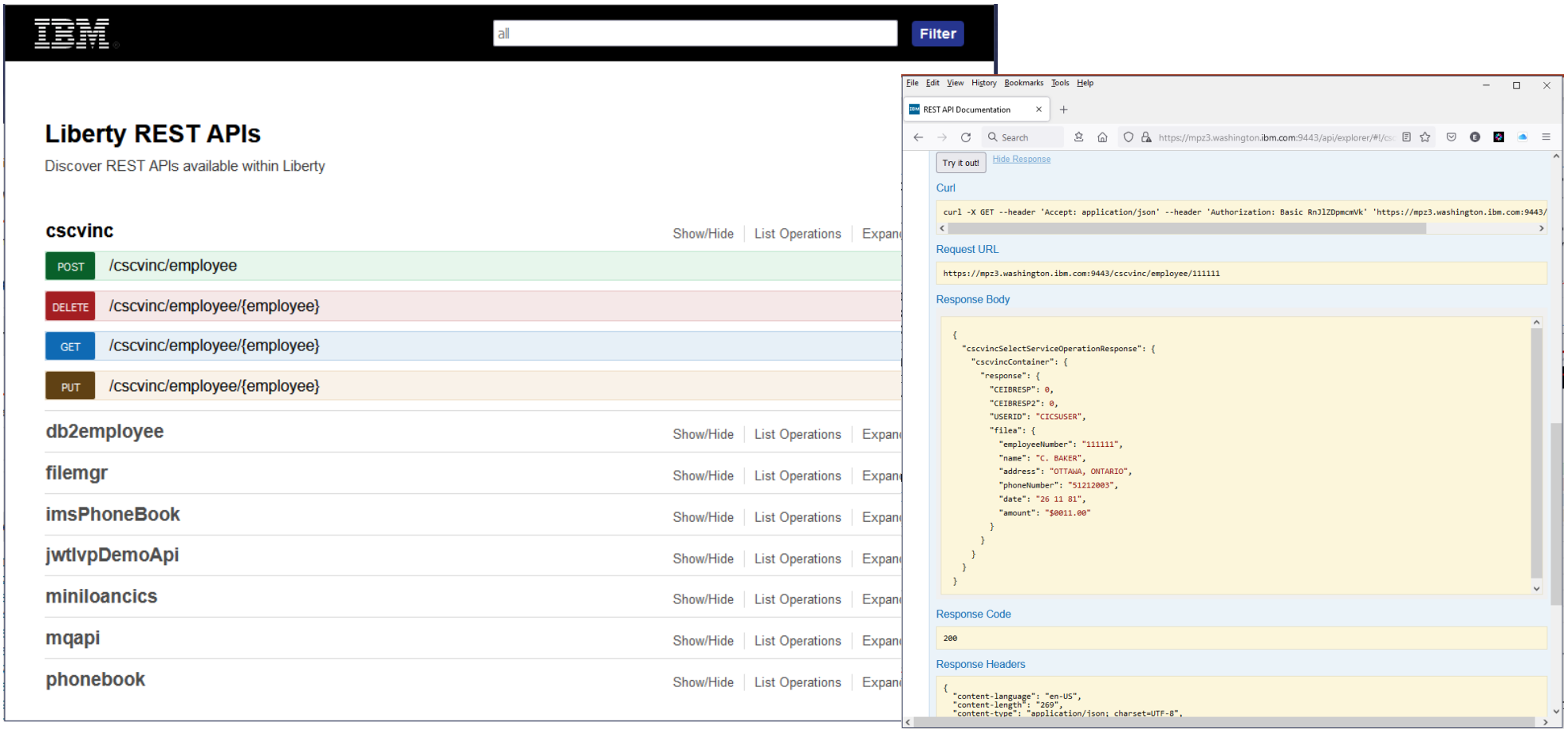
Testing an API with Postman



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

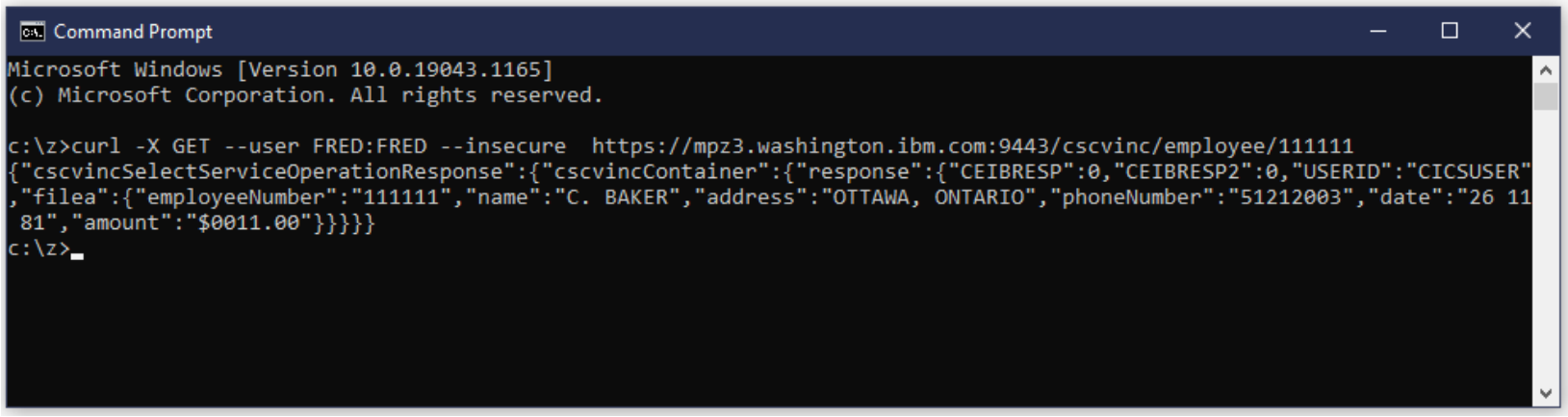
mitchj@us.ibm.com

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



mitchj@us.ibm.com

Testing an API using the cURL tool



https://curl.se/download.html

z/OS Connect administration API

Interface providing meta-data and life-cycle operations for z/OS Connect services, APIs and API requesters.

APIs : Operations for working with APIs

Show/Hide | List Operations | Expand Operations

GET	/apis	Returns a list of all the deployed z/OS Connect APIs
POST	/apis	Deploys a new API into z/OS Connect
DELETE	/apis/{apiName}	Undeploys an API from z/OS Connect
GET	/apis/{apiName}	Returns detailed information about a z/OS Connect API
PUT	/apis/{apiName}	Updates an existing z/OS Connect API

Services : Operations for working with services

Show/Hide | List Operations | Expand Operations

GET	/services	Returns a list of all the deployed z/OS Connect services
POST	/services	Deploys a new service into z/OS Connect
DELETE	/services/{serviceName}	Undeploys a service from z/OS Connect
GET	/services/{serviceName}	Returns detailed information about a z/OS Connect service
PUT	/services/{serviceName}	Updates an existing z/OS Connect service
GET	/services/{serviceName}/schema/{schemaType}	Returns the request or response schema for a z/OS Connect service

API Requesters : Operations that work with API Requesters.

Show/Hide | List Operations | Expand Operations

GET	/apiRequesters	Returns a list of all the deployed z/OS Connect API Requesters
POST	/apiRequesters	Deploys a new API Requester into z/OS Connect and invoke an API Requester call
DELETE	/apiRequesters/{apiRequesterName}	Undeploys an API Requester from z/OS Connect
GET	/apiRequesters/{apiRequesterName}	Returns the detailed information about a z/OS Connect API Requester
PUT	/apiRequesters/{apiRequesterName}	Updates an existing z/OS Connect API Requester

mitchj@us.ibm.com

Examples of curl in JCL (Rocket Software's tooling) and commands files

```
//*************************************************************
//*  SET SYMBOLS
//*************************************************************
//EXPORT EXPORT SYMLIST=(*)
// SET CURL= '/usr/lpp/rocket/curl'
//*************************************************************
//*  CURL Procedure
//*************************************************************
//CURL PROC
//CURL EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
// PEND
//*************************************************************
//*  STEP CURL - use cURL to deploy API cscvinc
//*************************************************************
//DEPLOY EXEC CURL
BPXBATCH SH export CURL=&CURL; +
$CURL/bin/curl -X PUT -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9445/zosConnect/apis/cscvinc?status=sto+
pped > null; +
$CURL/bin/curl -X DELETE -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9445/zosConnect/apis/cscvinc > null; +
$CURL/bin/curl -X POST -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
--data-binary @/u/johnson/cscvinc.aar +
--header "Content-Type: application/zip" +
https://wg31.washington.ibm.com:9445/zosConnect/apis
//*************************************************************
//*  STEP CURL - use cURL to invoke the API cscvinc
//*************************************************************
//INVOKE EXEC CURL
//SYSTSIN  DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; $CURL/bin/curl -X GET -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9445/cscvinc/employee/000100
```

```
@echo off
set TARGET=wg31.washington.ibm.com
set FILE=cscvinc_1.0.0
curl -X PUT --user user1:user1 --insecure
https://%TARGET%:9483/zosConnect/apiRequesters/%FILE%?status=stopped
curl -X DELETE --user user1:user1 --insecure
https://%TARGET%:9483/zosConnect/apiRequesters/%FILE%
curl -X POST --user user1:user1 --data-binary @%FILE%.ara  --header
"Content-Type: application/zip" --insecure
https://%TARGET%:9483/zosConnect/apiRequesters
echo ''
```

Change the status of API to stopped

Delete or remove the API from the server

Deploy the API to the server

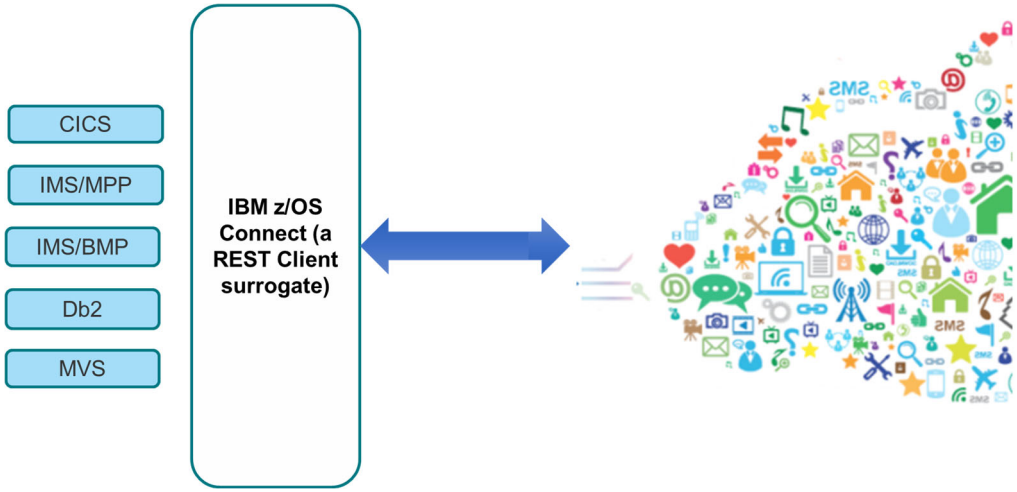
Execute the API

mitchj@us.ibm.com

https://www.rocketsoftware.com/platforms/ibm-z/curl-for-zos

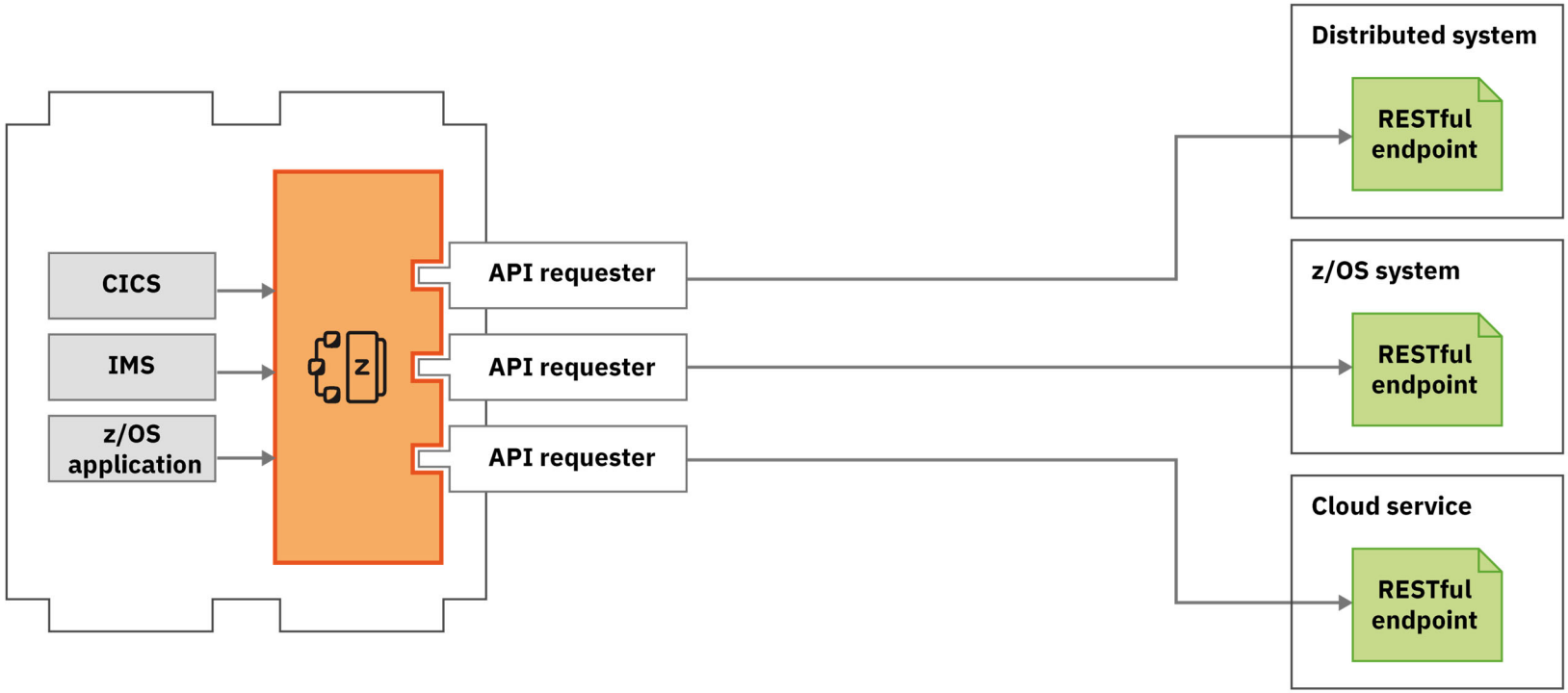
/api_toolkit/apiRequesters

Quick and easy **API mapping.**



mitchj@us.ibm.com

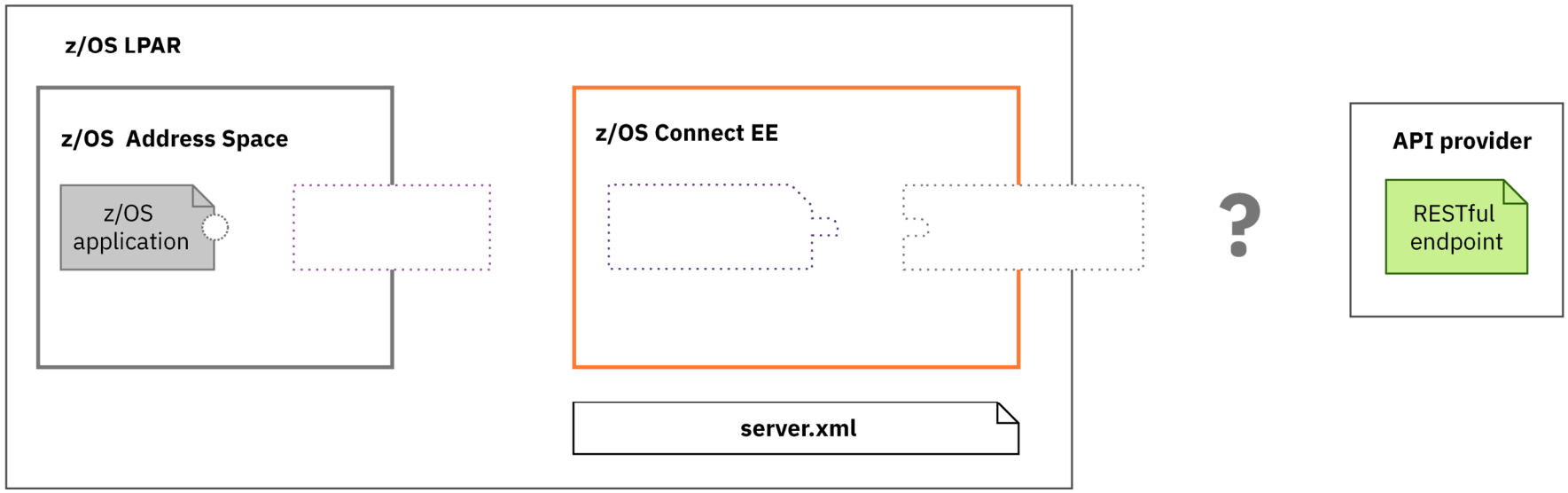
Use API requester to call external APIs from z/OS assets



mitchj@us.ibm.com

Steps to calling an external API

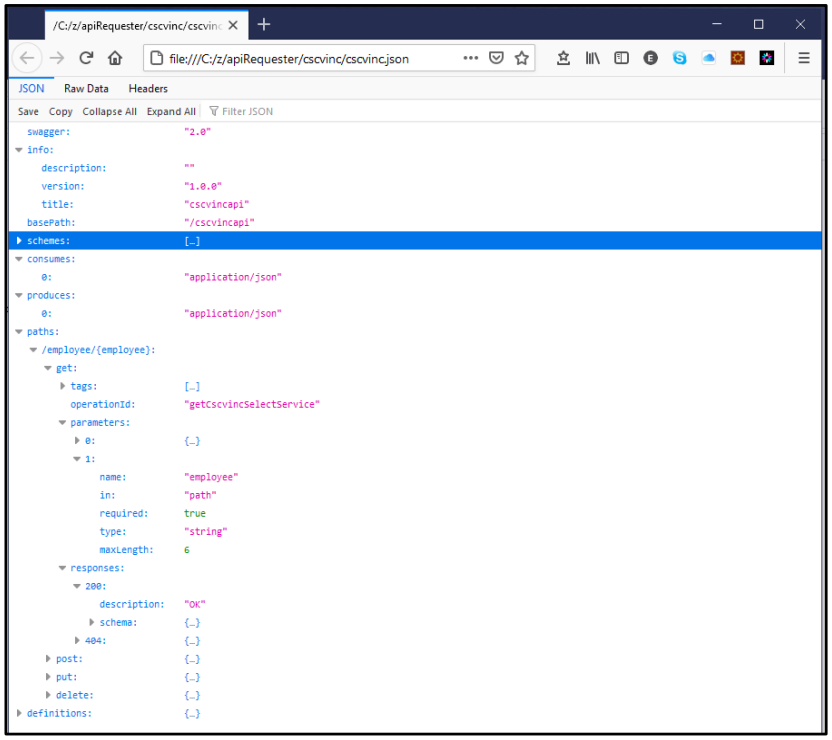
Starting point

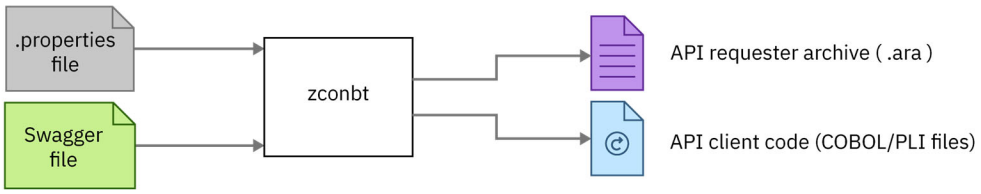


mitchj@us.ibm.com

Calling an external API requires the API's specification document

Start with the API's specification and generate the API requester archive file and API client code





properties file[#]

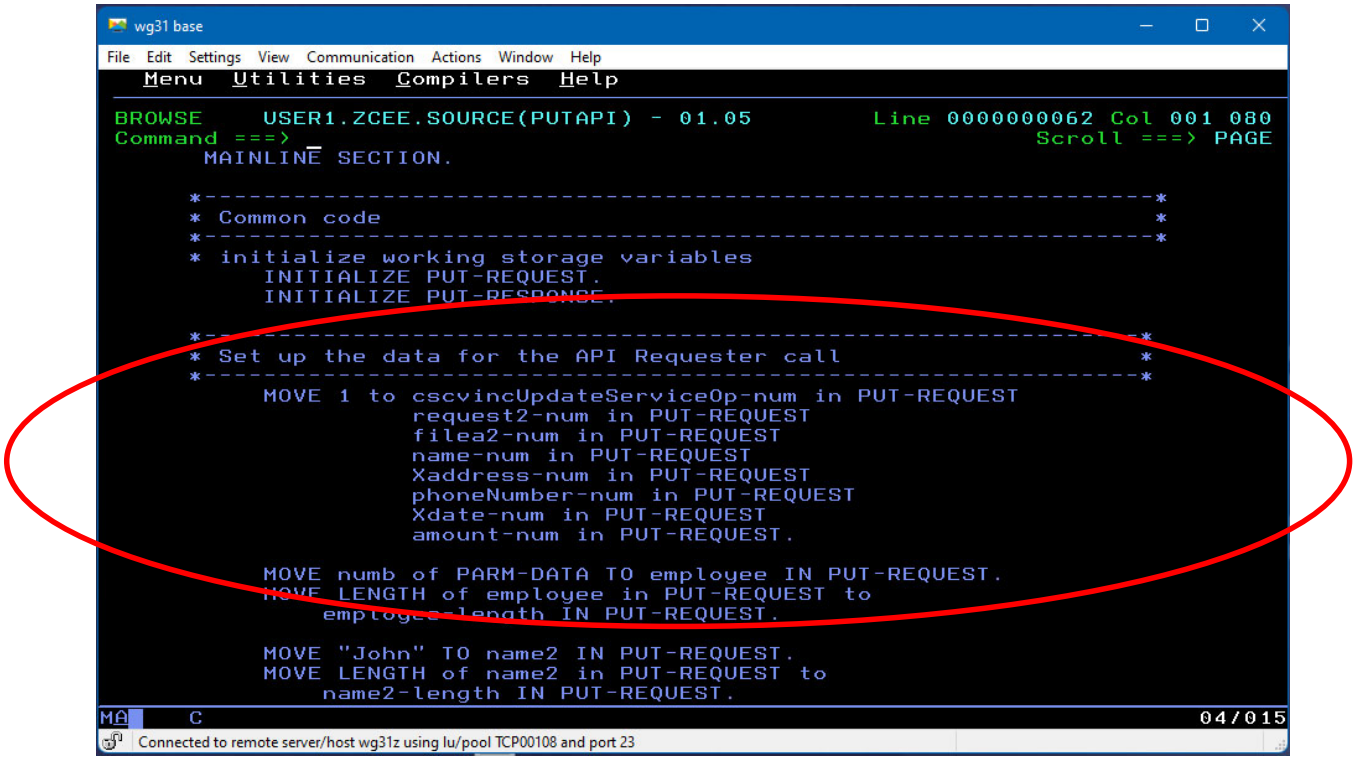
```
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

[#]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

The number of specific entries can be ambiguous

The COBOL copy book will include a counter variable (*variable*-num) for each variable whose number of occurrences is ambiguously defined in the specification document. The number of occurrences of these variables must be provided.



mitchj@us.ibm.com

Steps to calling an external API

Use the z/OS Connect build toolkit, e.g., *zconbt*, to generate an API requester archive file and at most, 3 copy books per method found in the 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.
```

https://www.ibm.com/docs/en/zos-connect/zosconnect/3.0?topic=installing-build-toolkit

mitchj@us.ibm.com

BTW, the z/OS Connect Build Toolkit can be executed on z/OS

```
//JOHNSONS JOB (ACCOUNT),JOHNSON,NOTIFY=&SYSUID,REGION=0M,
// CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1)
//*****************************************************************
//*  SET SYMBOLS
//*****************************************************************
//EXPORT EXPORT SYMLIST=(*)
// SET WORKDIR='/u/johnson/zconbt'
// SET ZCONDIR='/usr/lpp/IBM/zosconnect/v3r0/zconbt/bin'
//ZCONBT   EXEC PGM=IKJEFT01,REGION=0M,MEMLIMIT=4G
//SYSTSPRT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//SYSTSIN  DD *,SYMBOLS=EXECSYS
BPXBATCH SH +
  export WORKDIR=&WORKDIR; +
  export ZCONDIR=&ZCONDIR; +
  cd $WORKDIR; +
  $ZCONDIR/zconbt.zos -p cscvinc.properties -f=cscvinc.ara; +
  cp -v $WORKDIR/syslib/* "//'JOHNSON.ZCONBT.COPYLIB'"
```

cscvinc.properties

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

This assumes the zconbt.zip files was expanded into directory /usr/lpp/IBM/zosconnect/v3r0/zconbt using command *jar –tf zconbt.zip* and that the property file and Swagger JSON document are encoded in ASCII in directory /u/johnson/zconbt.

Tech-Tip: Copy books naming convention

- The naming convention for the generated COBOL copy books is based on the *requesterPrefix* value specified in the properties file. That value was set to CSC in this case, e.g., CSC######. The next 2 characters in the name are assigned sequentially as each API and method is processed, e.g., CSC00### and CSC01###, and CSC02###.

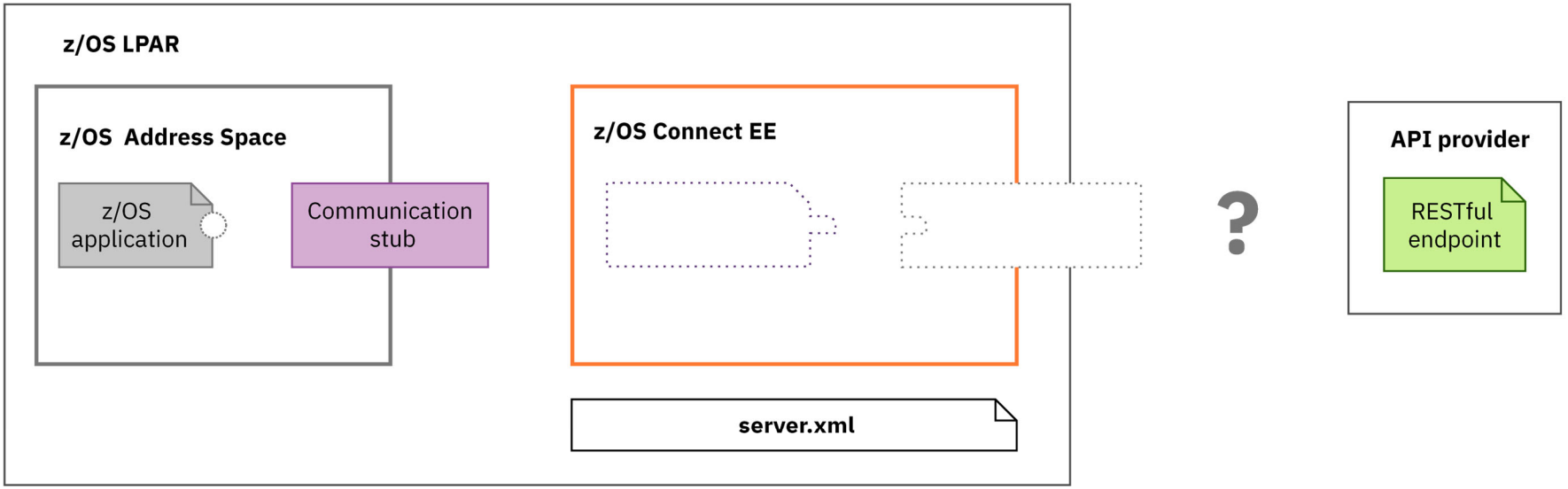
- The next character will be either a Q, P or an I. A “Q” for a request copy book, the “P” for a response copy book and the “I” for the copy book which contains information, e.g., method, path name etc. derived from the Swagger document
- Up to three copy books are generated for each method of each API found in the Swagger document.
  - In the previous example, there were 4 APIs with each having 1 method for a total of 12 copy books.
  - If there is no request message or no response message, then no copy book will be generated. But the messages stills need to have addressable storage in the application's working area.

```
* Request and Response
 01 GET-REQUEST.
    10  FILLER                    PIC X(1).
 01 GET-RESPONSE.
     COPY MQ000P01 SUPPRESS.
* Structure with the API information
 01 GET-INFO-OPER1.
     COPY MQ000I01 SUPPRESS.
```

mitchj@us.ibm.com

Steps for involving an external API from a COBOL progam

Update the application by adding the generated copy books, a common BAQRINFO copy book 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

```
      *-------------------------------------------------------------*
      * Call the communication stub                                 *
      *-------------------------------------------------------------*
      * Call the subsystem-supplied stub code to send
      * API request to zCEE
           CALL  COMM-STUB-PGM-NAME USING
                 BY REFERENCE    GET-INFO-OPER1
                 BY REFERENCE    BAQ-REQUEST-INFO
                 BY REFERENCE    BAQ-REQUEST-PTR
                 BY REFERENCE    BAQ-REQUEST-LEN
                 BY REFERENCE    BAQ-RESPONSE-INFO
                 BY REFERENCE    BAQ-RESPONSE-PTR
                 BY REFERENCE    BAQ-RESPONSE-LEN.
```

mitchj@us.ibm.com

ibm.biz/zosconnect-configure-comms-stub

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

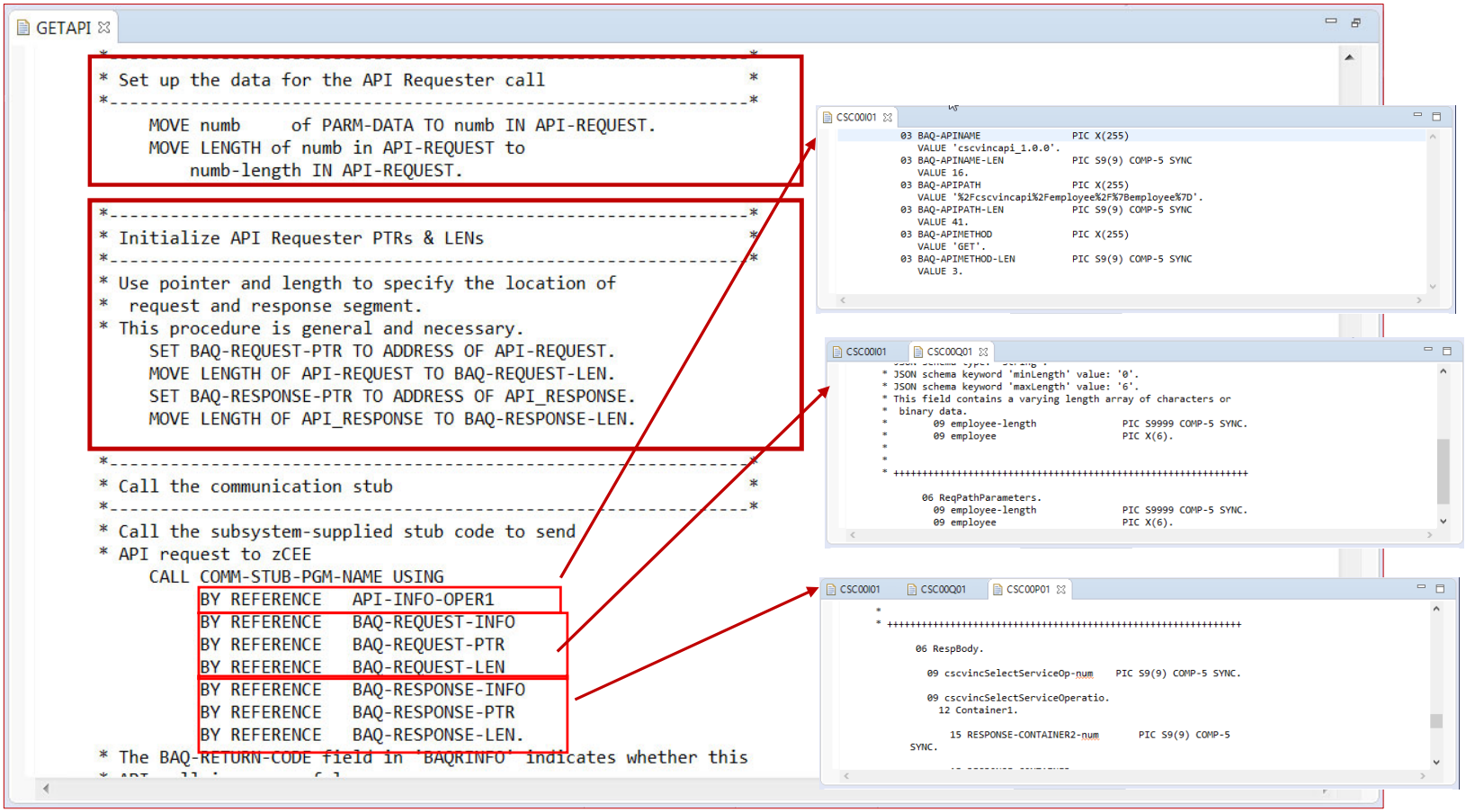
CSC00I01

```
           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 use pointers to pass working storage addresses of the copy books



mitchj@us.ibm.com

Steps to calling an external API

Accessing the results that have been stored in working storage

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.
```

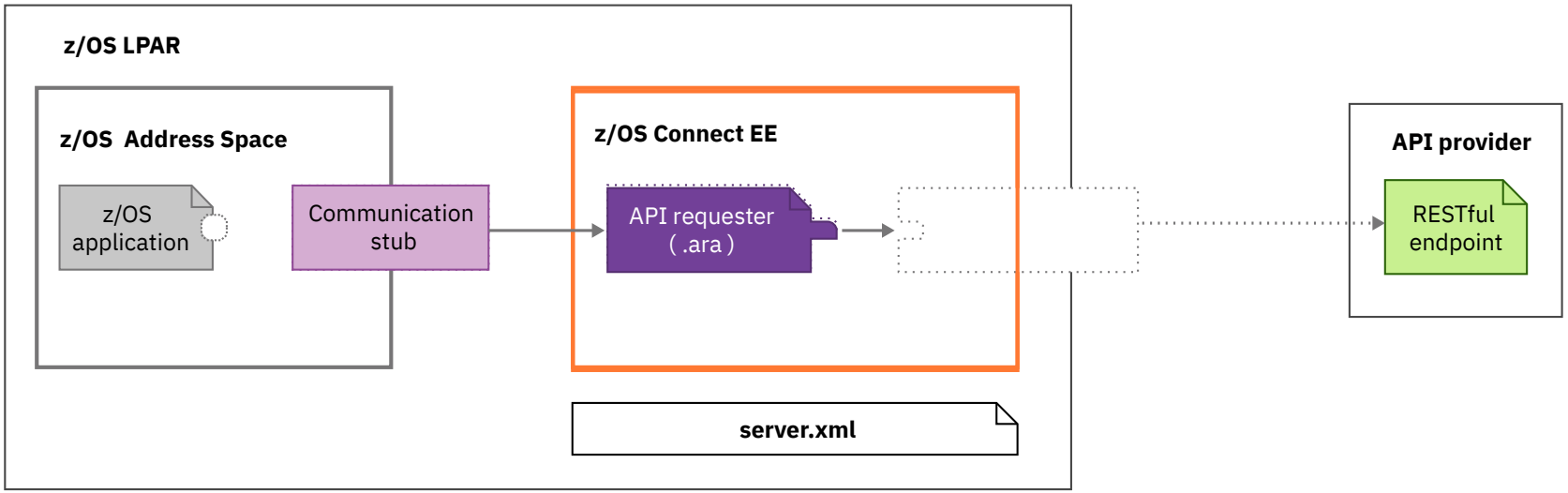
mpz3

```
File  Edit  Settings  View  Communication  Actions  Window  Help
  Menu  Utilities  Compilers  Help
BROWSE    ZCEE30.SBAQCOB(BAQRINFO)                Line 0000000066 Col 001 080
Command ===>                                        Scroll ===> PAGE

      01  BAQ-RESPONSE-INFO.
        03 BAQ-RESPONSE-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 0.
        03 BAQ-STUB-NAME                PIC X(8).
        03 BAQ-RETURN-CODE              PIC S9(9) COMP-5 SYNC.
           88 BAQ-SUCCESS                 VALUE 0.
           88 BAQ-ERROR-IN-API            VALUE 1.
           88 BAQ-ERROR-IN-ZCEE           VALUE 2.
           88 BAQ-ERROR-IN-STUB           VALUE 3.
           88 BAQ-ERROR-NO-RESPONSE       VALUE 4.
        03 BAQ-STATUS-CODE              PIC S9(9) COMP-5 SYNC.
        03 BAQ-STATUS-MESSAGE           PIC X(1024).
        03 BAQ-STATUS-MESSAGE-LEN       PIC S9(9) COMP-5 SYNC.
****************************** Bottom of Data ********************************
MA    B                                                          18/058
Connected to remote server/host mpz3 using lu/pool MPZ30021 and port 23
```

mitchj@us.ibm.com

Steps to calling an external API

Deploy the API requester (.ara) archive

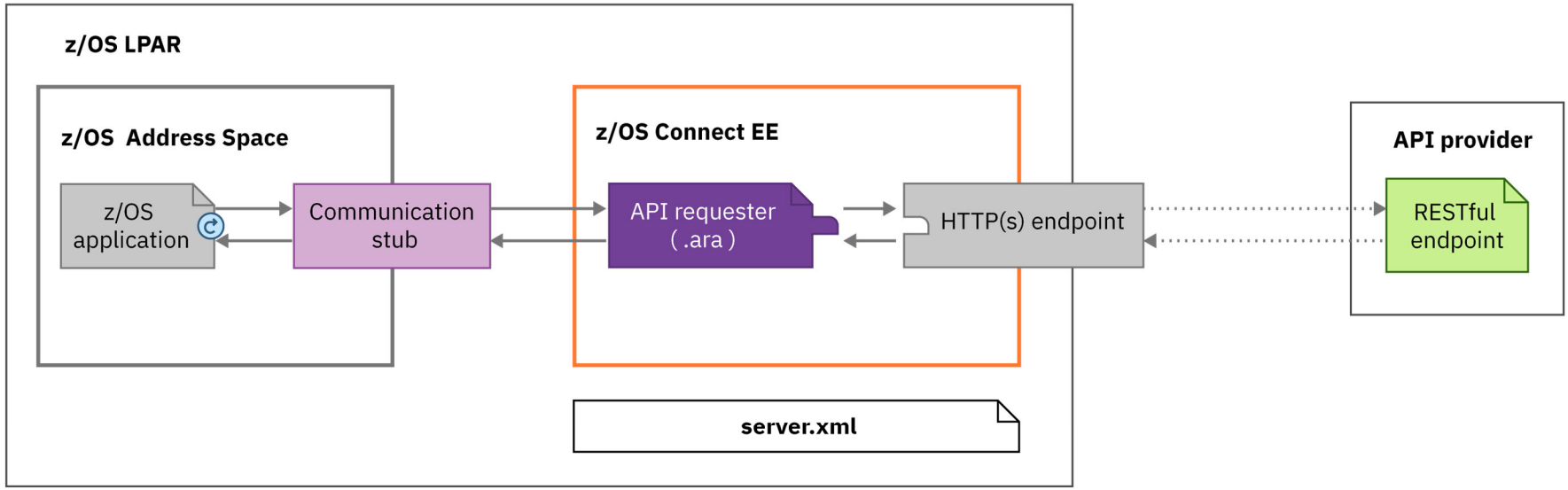


Deploy your API requester archive to the *apiRequesters* directory.

mitchj@us.ibm.com

Steps to calling an external API

Done



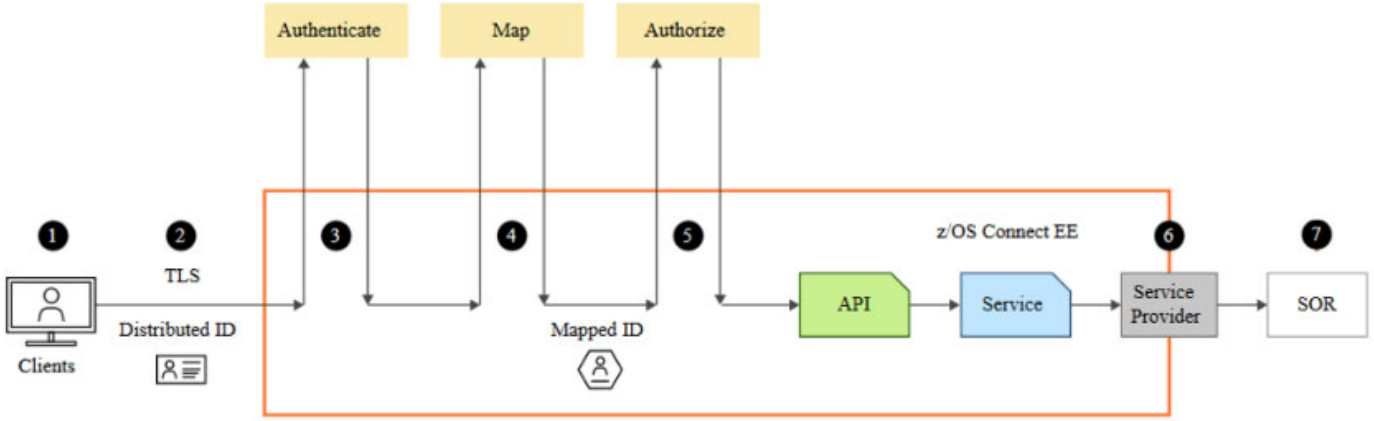
mitchj@us.ibm.com

/security

How is security implement?

mitchj@us.ibm.com

Typical z/OS Connect EE API Provider security flow



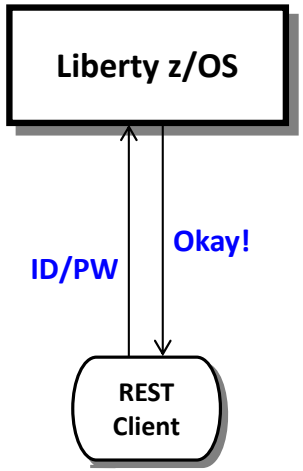
1. The credentials provided by the client
2. Secure the connection to the z/OS Connect EE server
3. Authenticate the client. This can be within the z/OS Connect EE server or by requesting verification from a third-party server
4. Map the authenticated identity to a user ID in the user registry
5. Authorize 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

Several different ways this can be accomplished:

Basic Authentication



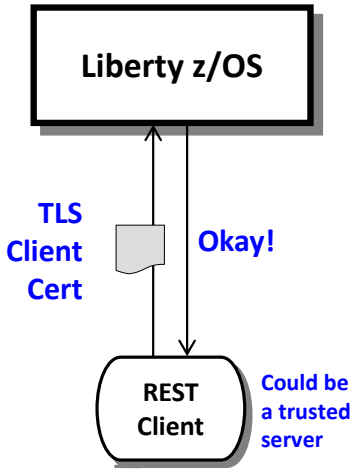
Server prompts for ID/PW

Client supplies ID/PW or ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

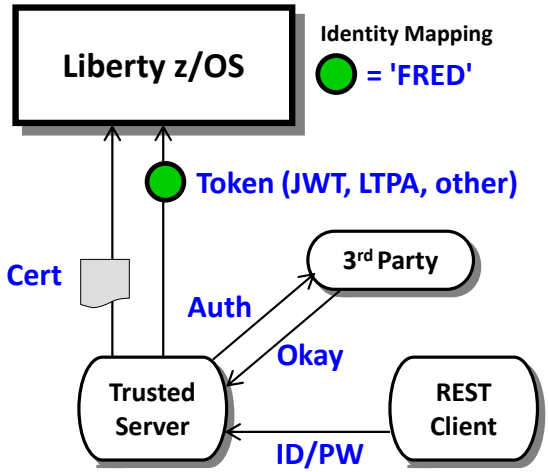
Client supplies certificate

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

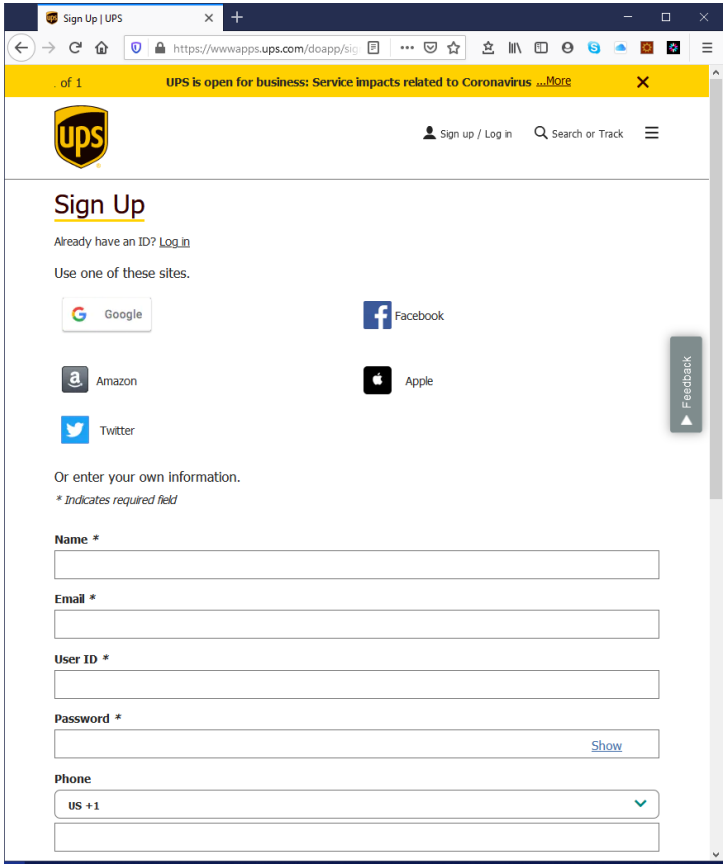
mitchj@us.ibm.com

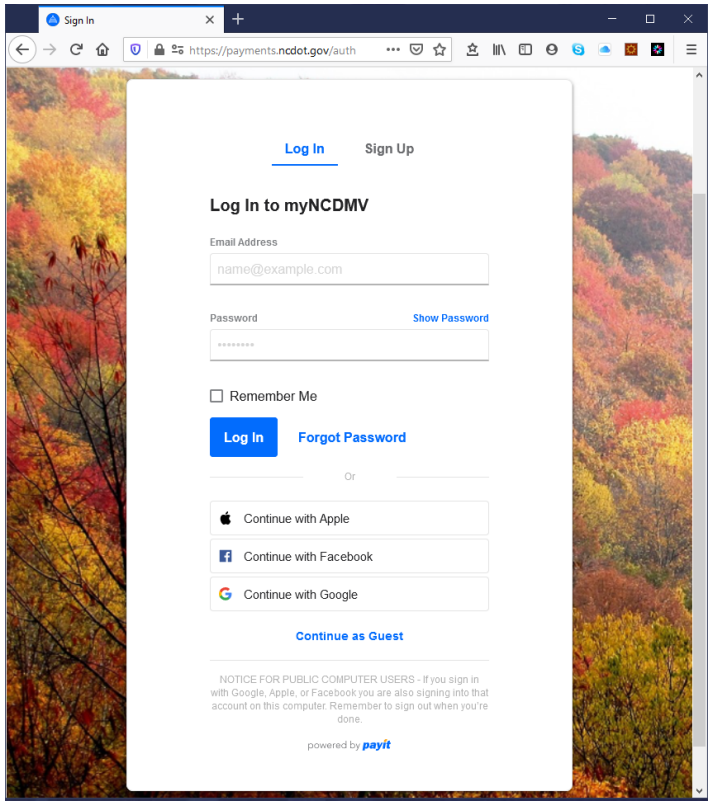
OPENAPI 3 roles


```
<!-- Enable features -->
   <featureManager>
       <feature>appSecurity-2.0</feature>
   </featureManager>
   <webAppSecurity allowFailOverToBasicAuth="true" />

   <basicRegistry id="basic" realm="zosConnect">
       <user name="Fred" password="fredpwd" />
       <user name="user1" password="user1" />
       <user name="user2" password="user2" />
       <user name="user3" password="user3" />
    <group name="Manager">
          <member name="Fred"/>
    </group>
    <group name="Staff">
          <member name="Fred"/>
          <member name="user1"/>
          <member name="user2"/>
    </group>
   </basicRegistry>

   <authorization-roles id="Manager">
       <security-role name="Manager">
           <group name="managerGroup"/>
       </security-role>
    </authorization-roles>
    <authorization-roles id="Staff">
       <security-role name="Staff">
           <group name="staffGroup"/>
       </security-role>
   </authorization-roles>
```

mitchj@us.ibm.com

Third Party Authentication Examples

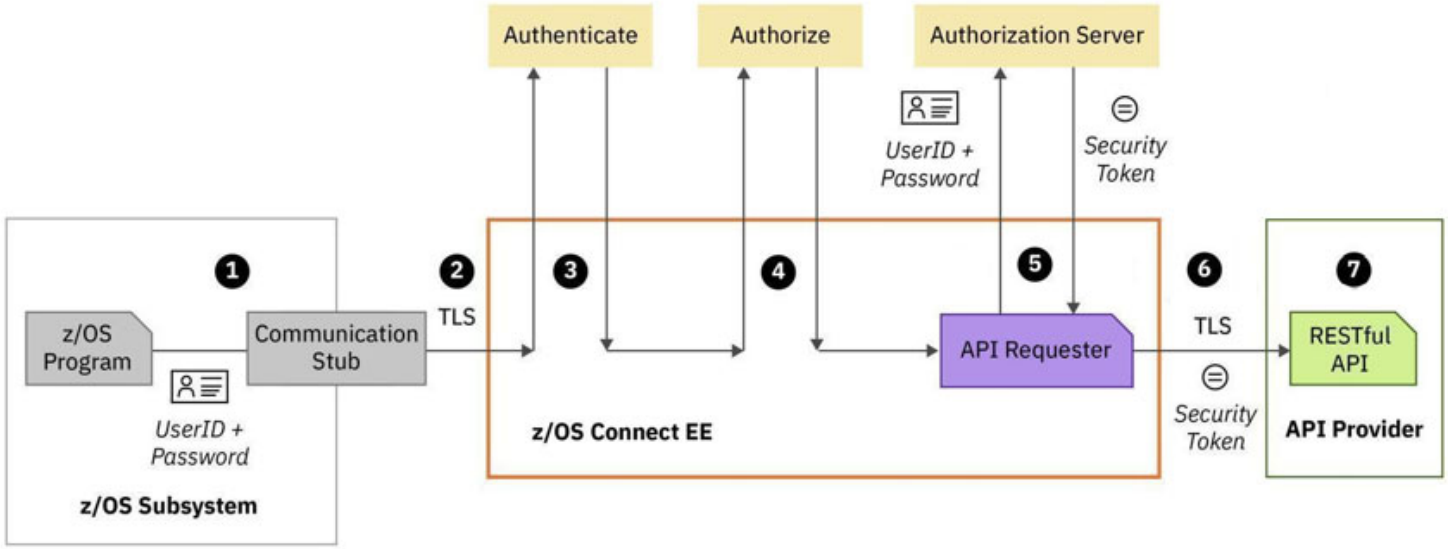




mitchj@us.ibm.com

mitchj@us.ibm.com

Typical z/OS Connect EE API Requester security flow

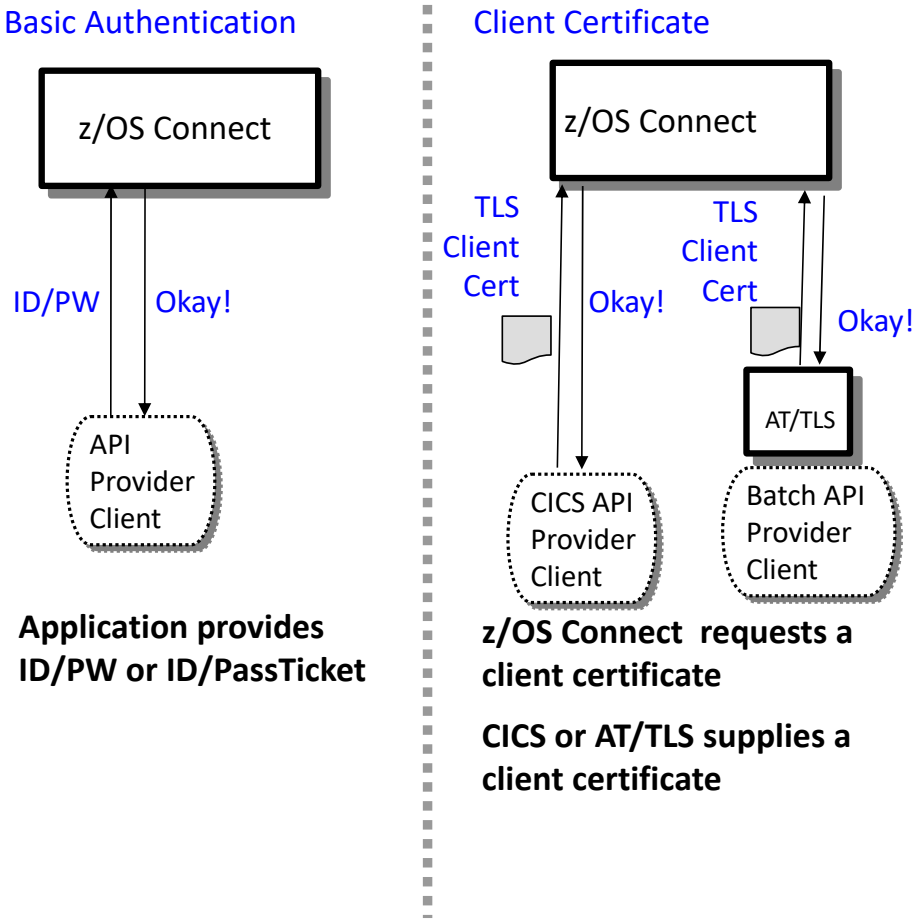


1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS 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

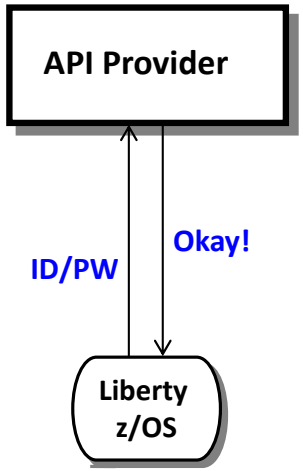


mitchj@us.ibm.com
mitchj@us.ibm.com

API Requester - API Provider Authentication

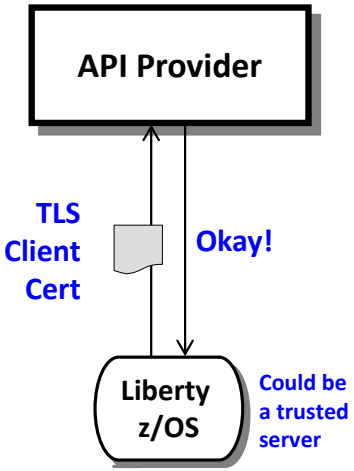
Several different ways this can be accomplished:

Basic Authentication



zCEE supplies ID/PW or ID/Passticket

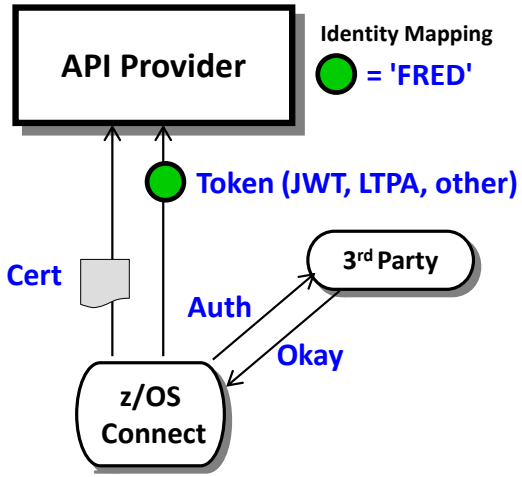
Client Certificate



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

Basic authentication – non-CICS COBOL API Requester

- A MVS batch, IMS or Db2 stored procedure requester application sends basic authentication information (identity and password) by using environment variables.
  - BAQUSERNAME
  - BAQPASSWORD

- The variables can be provided in JCL using CEEOPTS DD statement:

```
//CEEOPTS  DD *
     POSIX(ON),
     ENVAR("BAQURI=wg31.washington.ibm.com",
    "BAQPORT=9080",
    "BAQUSERNAME=USER1",
    "BAQPASSWORD=USER1")
```

- Or, provided by using a CEEROPT or CEEUOPT module:

```
CEEROPT CSECT
CEEROPT AMODE ANY
CEEROPT RMODE ANY
CEEXOPT POSIX=((ON),OVR),                                          +
               ENVAR=(('BAQURI=wg31.washington.ibm.com',           +
               'BAQPORT=9120',                                     +
               'BAQUSERNAME=USER1',                                +
               'BAQPASSWORD=USER1'),OVR),                          +
               RPTOPTS=((ON),OVR)
       END
```

Tech/Tip: This is good opportunity to use a pass ticket rather than a password

mitchj@us.ibm.com

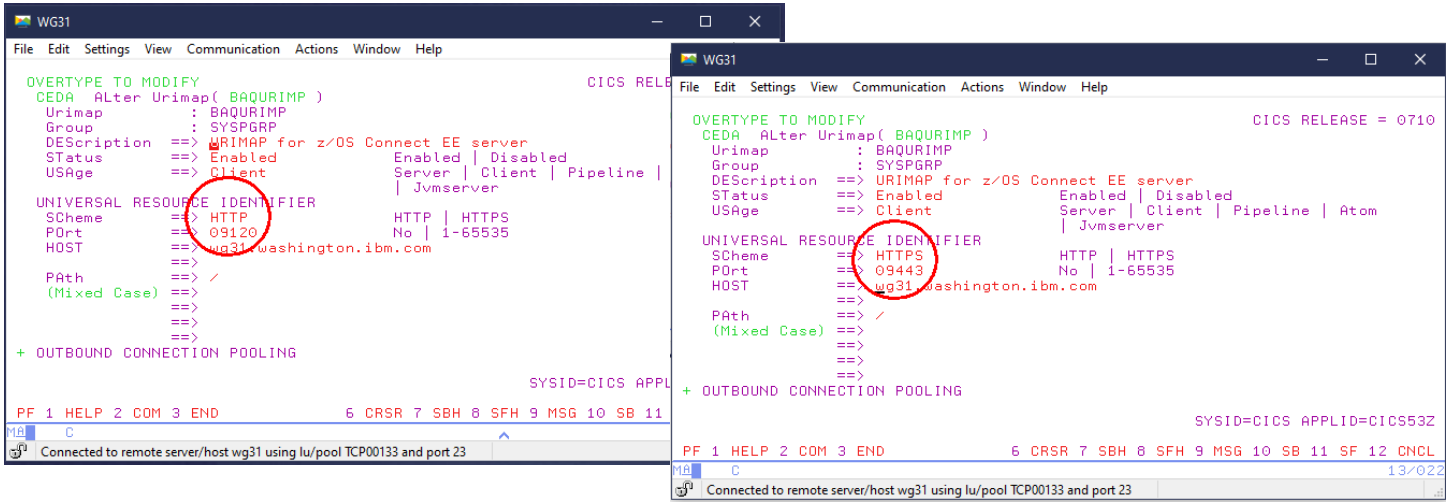
Tech/Tip: API Requester - HTTP v HTTPS

MVS Batch and IMS with and without an outbound AT-TLS policy

```
CEEOPTS DD *
 POSIX(ON),
 ENVAR("BAQURI=wg31.washington.ibm.com",
 "BAQPORT=9080")
```

```
CEEOPTS DD *
 POSIX(ON),
 ENVAR("BAQURI=wg31.washington.ibm.com",
 "BAQPORT=9443")
```

CICS URIMAPs



Field BAQ-ZCON-SERVER-URI was added to BAQRINFO in V3.0.37.

MOVE "URIMAP01" TO BAQ-ZCON-SERVER-URI.

mitchj@us.ibm.com

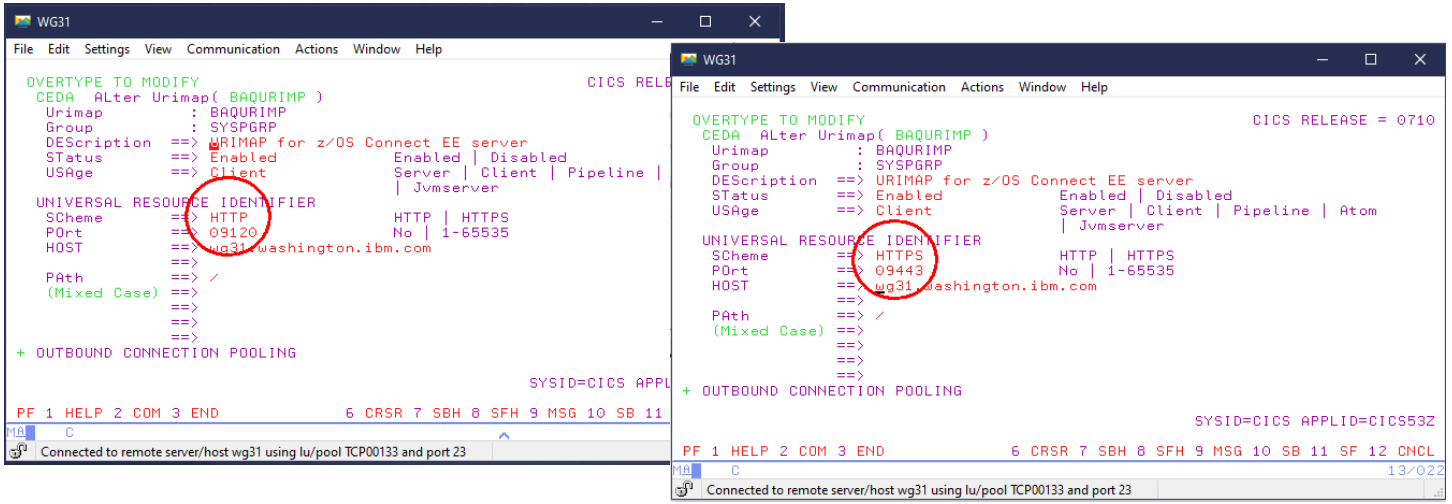
Tech/Tip: API Requester - HTTP v HTTPS

MVS Batch and IMS with and without an outbound AT-TLS policy

```
CEEOPTS DD *
 POSIX(ON),
 ENVAR("BAQURI=wg31.washington.ibm.com",
 "BAQPORT=9080")
```

```
CEEOPTS DD *
 POSIX(ON),
 ENVAR("BAQURI=wg31.washington.ibm.com",
 "BAQPORT=9443")
```

CICS URIMAPs

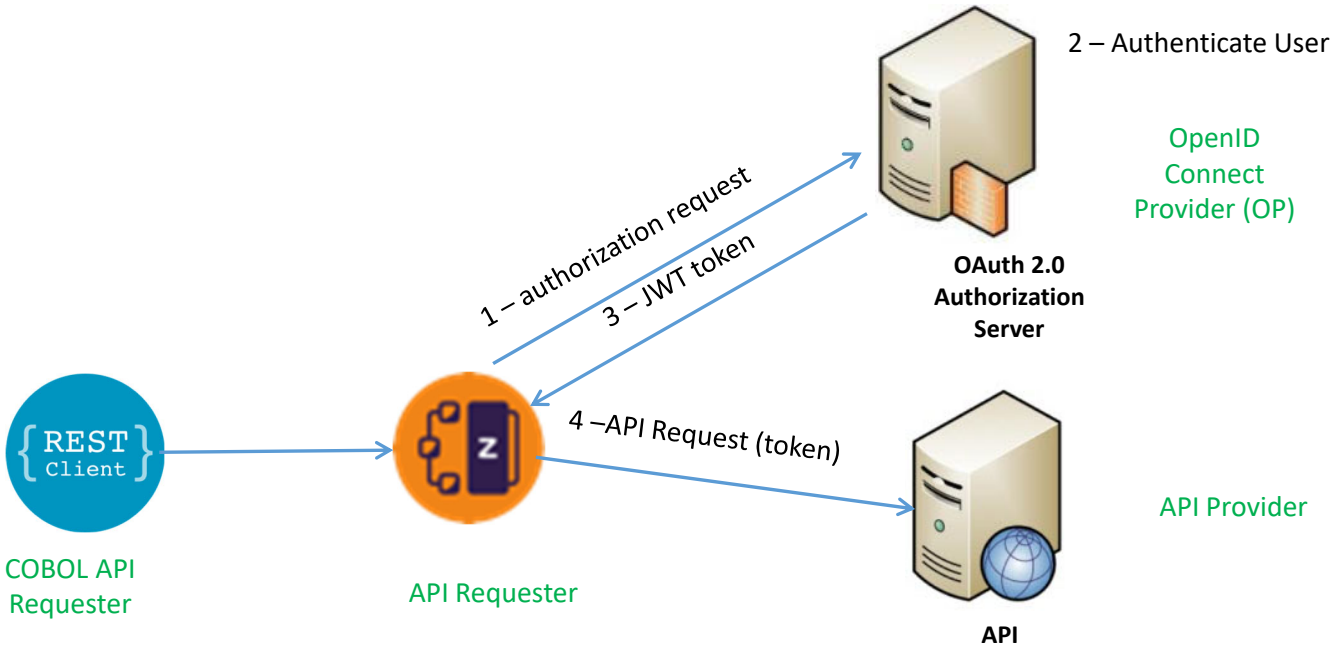


Field BAQ-ZCON-SERVER-URI was added to BAQRINFO in V3.0.37.

MOVE "URIMAP01" TO BAQ-ZCON-SERVER-URI.

mitchj@us.ibm.com

z/OS Connect OAuth Flow for API requester

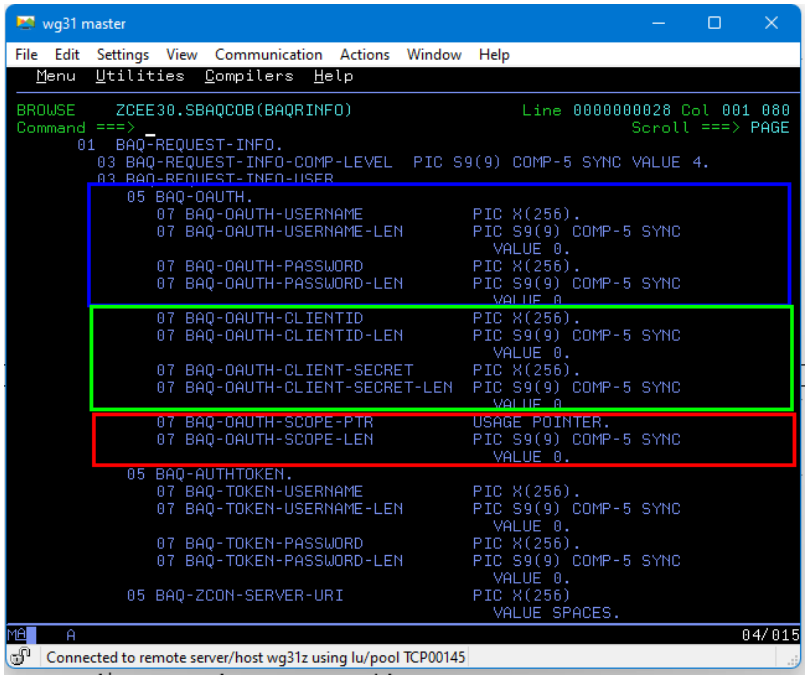


Grant Types:

- client_credentials
- password

mitchj@us.ibm.com

Configuring OAuth support – BAQRINFO copy book



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.

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

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
  - Outbound REST APIs

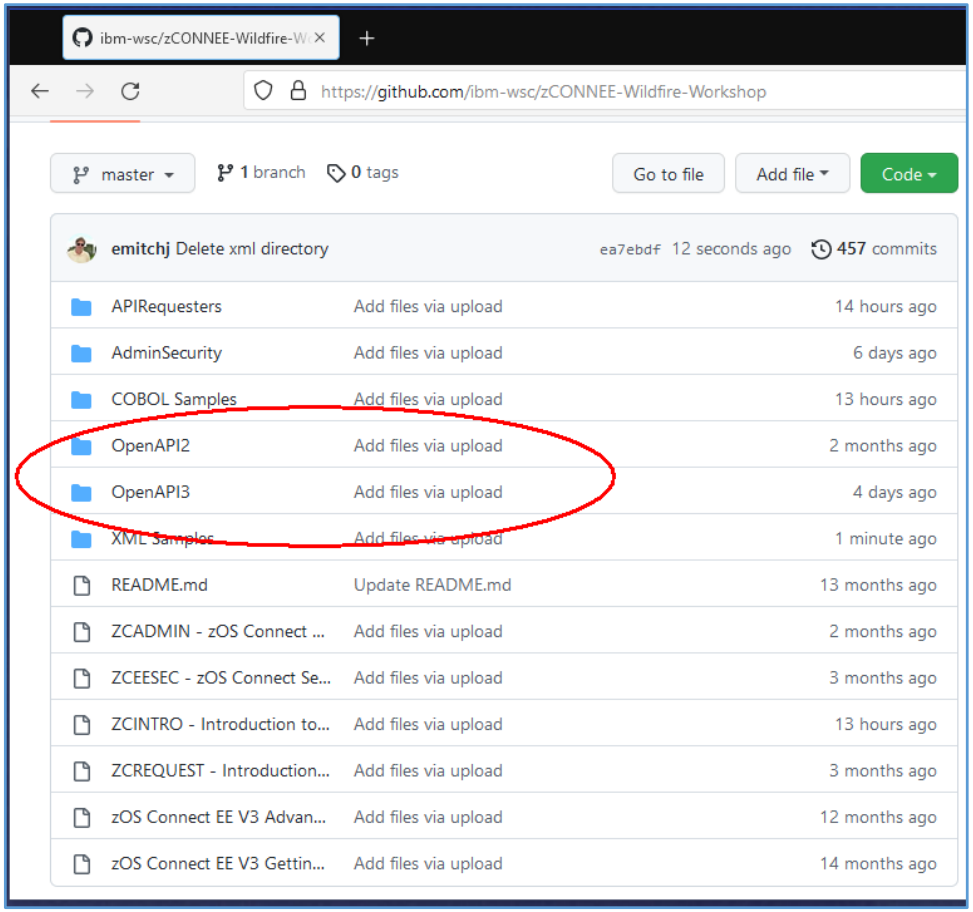
- Accessing RESTful API from z/OS COBOL Applications
- A brief overview of z/OS Connect Security*

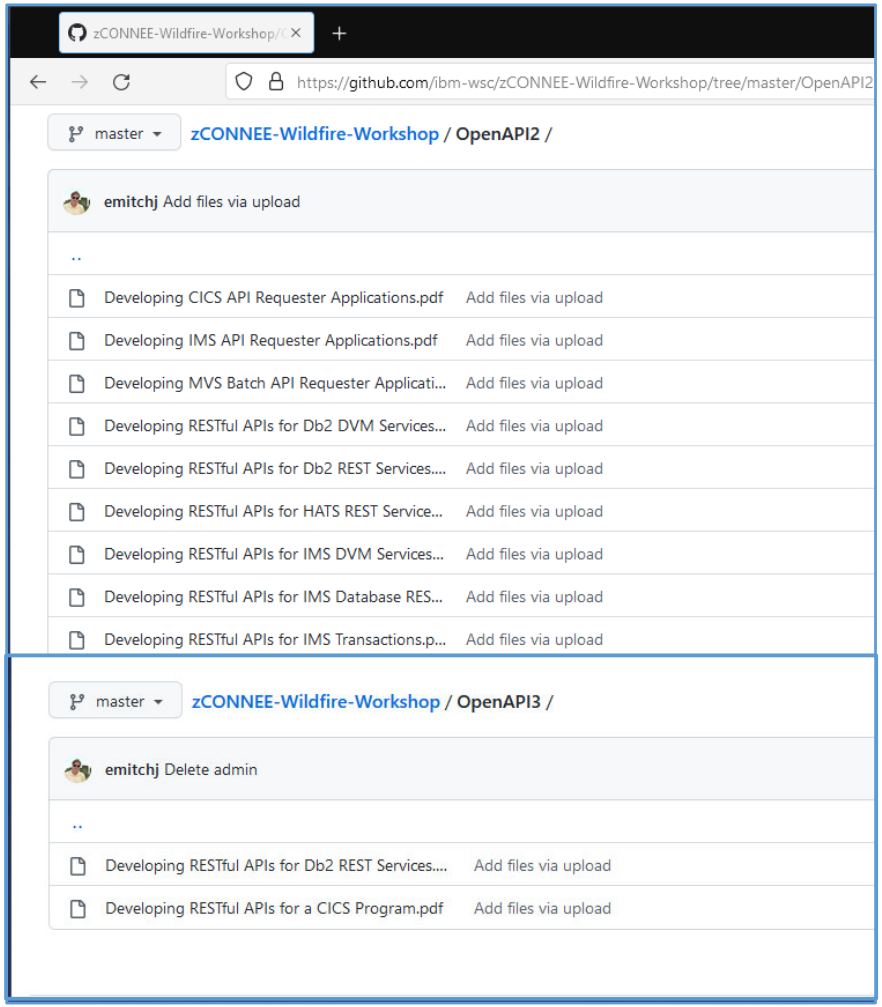
*For more on security, contact your local IBM rep regarding the schedule of workshop *zOSSEC1 IBM z/OS Connect Administration/Security Wildfire Workshop*

mitchj@us.ibm.com

z/OS Connect Wildfire Github Site

https://ibm.biz/zCEEWorkshopMaterial



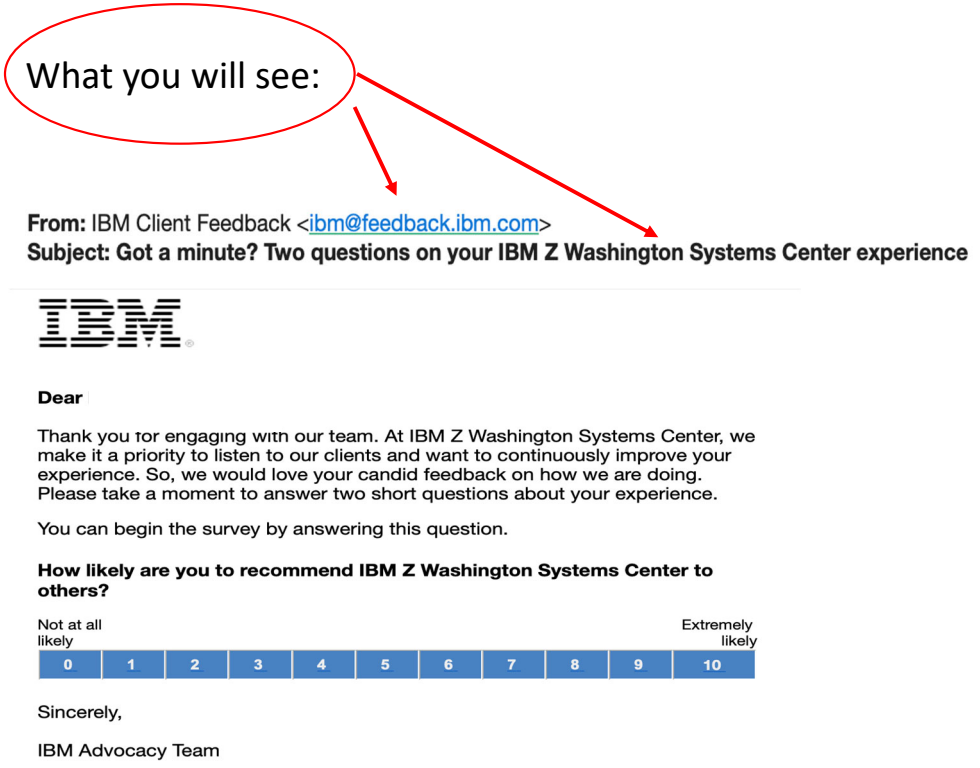


- **Contact your IBM representative to schedule access to these exercises**

mitchj@us.ibm.com

WSC wants your feedback!

What you will see:



From: IBM Client Feedback <ibm@feedback.ibm.com>
Subject: Got a minute? Two questions on your IBM Z Washington Systems Center experience

IBM

Dear

Thank you for engaging with our team. At IBM Z Washington Systems Center, we make it a priority to listen to our clients and want to continuously improve your experience. So, we would love your candid feedback on how we are doing. Please take a moment to answer two short questions about your experience.

You can begin the survey by answering this question.

How likely are you to recommend IBM Z Washington Systems Center to others?

Not at all likely										Extremely likely
0	1	2	3	4	5	6	7	8	9	10

Sincerely,

IBM Advocacy Team

you will NOT receive a new survey if you already responded to an IBM Survey from Medallia in the last **60 days OR if you haven't responded within the last **30 days****

Thank you for listening and your questions

And thank you for completing the Medallia survey

mitchj@us.ibm.com