Accessing z/OS resources with REST APIs using IBM z/OS Connect

Mitch Johnson mitchj@us.ibm.com
John Brefach John.J.Brefach@ibm.com
Washington System Center

Agenda

- An Introduction and Overview of using REST API

- Enabling RESTful API to z/OS resources, e.g.
  - CICS
  - Db2
  - IMS/TM
  - IMS/DB
  - MQ

- A brief overview of z/OS Connect API Security 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 related workshops. . .

Accessing REST APIs from z/OS using IBM z/OS Connect

Mitch Johnson mitchj@us.ibm.com
John Brefach John.J.Brefach@ibm.com
Washington System Center

WebSphere Liberty Profile Administration

Managing WebSphere Liberty Profile servers for IBM z/OS Connect (OpenAPI 2 and OpenAPI 3), IBM MQ REST Console and zOSMF

IBM Washington Systems Center

z/OS Connect OpenAPI 3

Designer and z/OS Native server Experiences and Observations

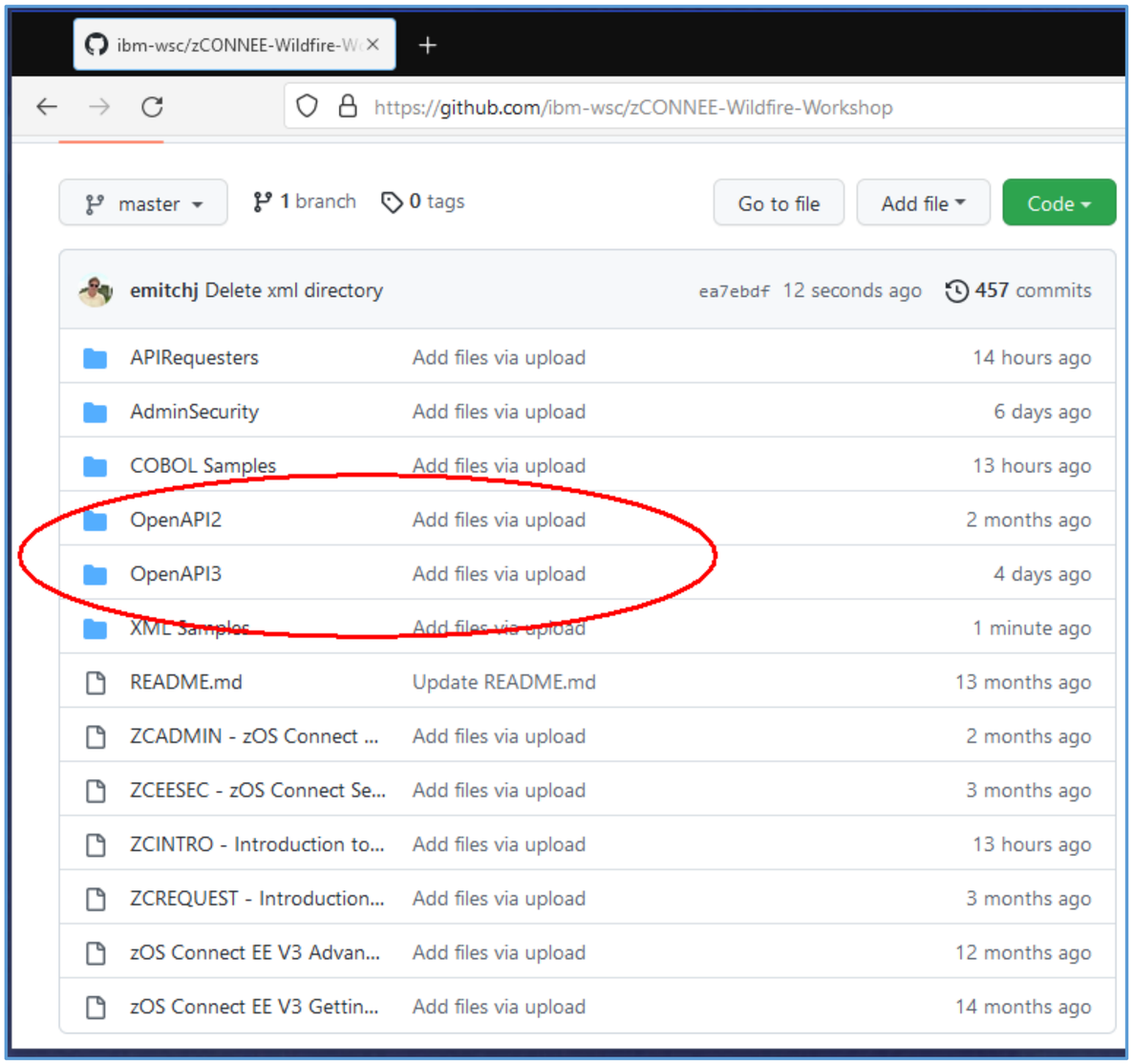
Mitch Johnson
mitchj@us.ibm.com
Washington Systems Center

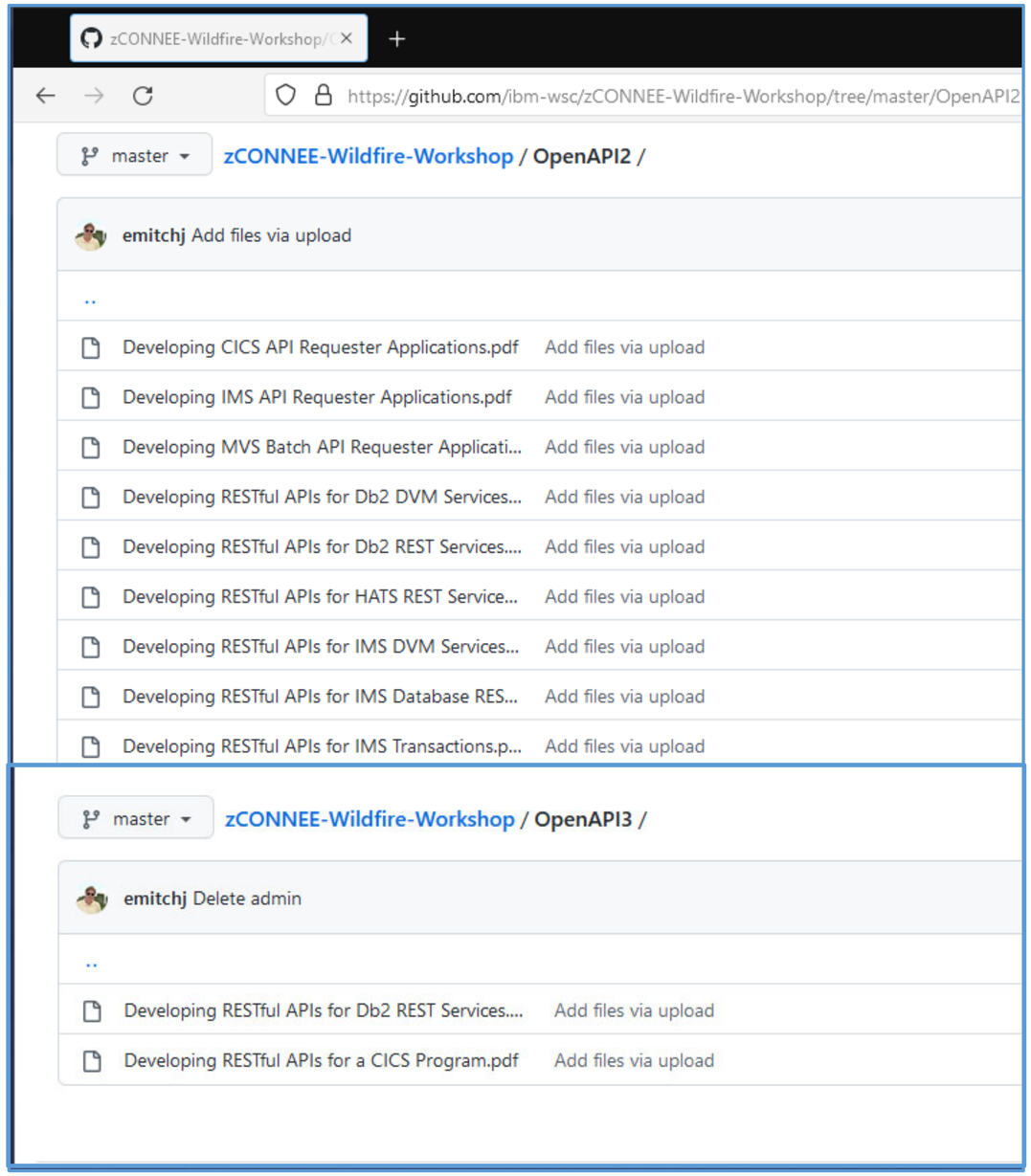
IBM Washington Systems Center

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

z/OS Connect Wildfire Github Site

https://ibm.biz/zCEEWorkshopMaterial





mitchj@us.ibm.com

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

z/OS Connect EE exposes z/OS resources to clients in the “cloud” using RESTful APIs

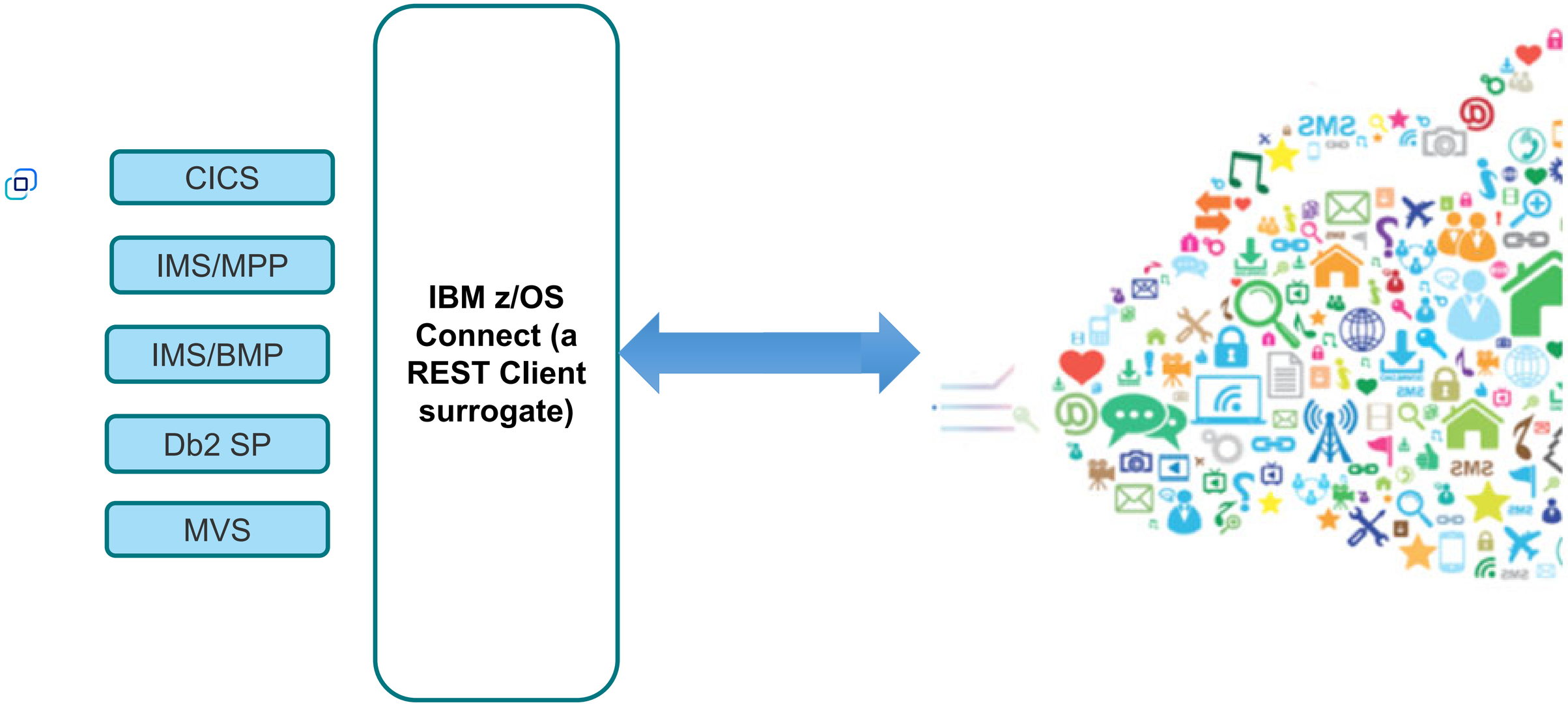
IBM z/OS Connect (an API Gateway)

- 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

Note that z/OS Connect also exposes external REST APIs in the “cloud” to z/OS applications



mitchj@us.ibm.com

Let’s start by defining REST

/what_is_REST?

And what makes an API “RESTful”?

mitchj@us.ibm.com

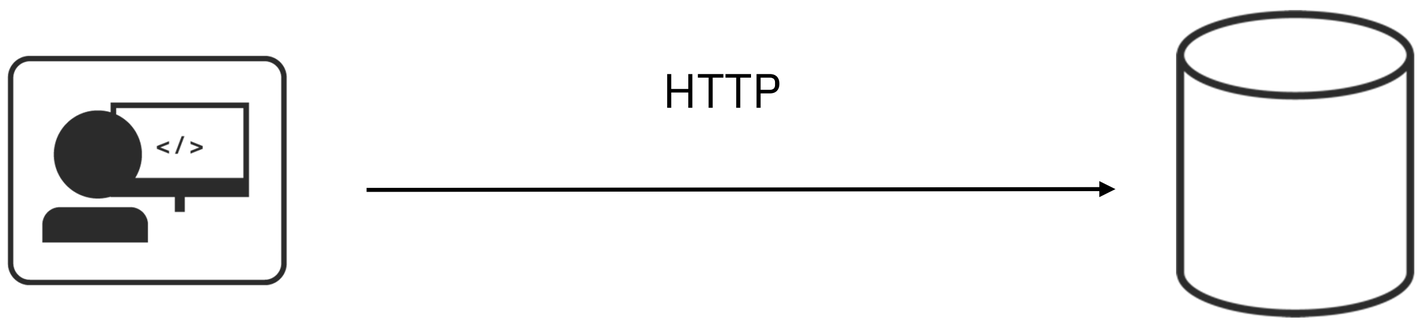
REST is architectural programming style

REST is an 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

Uniform Resource Identifier (URI) identifies the resource (s)

Uses HTTP verbs to specify Create, Read, Update, Delete (CRUD) operations,
for background on CURD, see URL
https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

POST
GET
PUT
DELETE

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

Use Path and Query parameters to refine the request

Uniform Resource Locator (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

Why is REST popular today?

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

RESTful Examples

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

GET `/account?number=1234`

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

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

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

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

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

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

mitchj@us.ibm.com

Not every REST API is a RESTful API

(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 { “*/89:N$3)”: “J5)”,
       “2$9:N$3)” : “B25++9”,
       “$((8”     : “^[ O2( S:8)):”,
       “6.54)N5”  : “[^_‘a [^_‘abc” }
```

→

```
RESPONSE HTTP 201 CREATED
BODY { “id”     : “12345”,
       “4$3)”   : “J5) B25++9”,
       “$((8)99”: “^[ N)= S:8)):”
       “:)2”    : “[^_‘a [^_‘abcx}
```

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

```
POST http://www.acme.com/customers/12345
BODY { “;6($:)F/)2(”: “$((8)99”,
       “4)=V$2;)”   : “^[ N)= S:8)):”}
```

→

```
RESPONSE HTTP 200 OK
BODY { “id”     : “12345”,
       “name”   : “Joe Bloggs”,
       “address”: “^[ N)= S:8)):”
       “tel”    : “01234 123456” }
```

mitchj@us.ibm.com

One goal is to try to simplify the development of client applications

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 { "4$3)": "J5) B25++9",
       "$((8)99": "^[ O2( S:8)):",
       "6.54)N5": "[^_'a [^_'abc" }
```

→

```
RESPONSE HTTP 201
BODY { "id"     : "12345",
       "4$3)"   : "J5) B25++9",
       "$((8)99": "^[ N)= S:8)):"
       "6.54)N5": "[^_'a [^_'abcx}
```

3. Use JSON name/value pairs

```
PUT http://www.acme.com/customers/12345
BODY { "$((8)99"   : "^[ N)= S:8)):"}
```

→

```
RESPONSE HTTP 200 OK
```

mitchj@us.ibm.com

Non-REST client code is usually tightly coupled with the resource being accessed

CICS client

*cscinq.java

```
24      COMMAREABUFFER commarea = new COMMAREABUFFER();
25      commarea.setNumb("111111");
26
27      //CICSConnection connection = new CICSConnection();
28      com.ibm.connector2.cics.ECIInteractionSpec interSpec = new com.ibm.connector2.cics.ECIInteractionSpec();
29
30      //      Create an interaction spec
31      interSpec.setFunctionName("CSCVINQ");
32      interSpec.setCommareaLength(commarea.getSize());
33
34      //      Create a Connection Factory (non-Managed)
35      com.ibm.connector2.cics.ECIManagedConnectionFactory mcf = new com.ibm.connector2.cics.ECIManagedConnectionFactory();
36      mcf.setConnectionURL("tcp://ctfmvs09.rtp.raleigh.ibm.com");
37      //mcf.setConnectionURL("local:");
38      mcf.setServerName("NQA11C00");
39      mcf.setPortNumber("2006");
40
41      try {
42          //      Create an ECI connection factory using the nonMCF
43          com.ibm.connector2.cics.ECIConnectionFactory cf =
44              (com.ibm.connector2.cics.ECIConnectionFactory)mcf.createConnectionFactory();
45
46          //      Create a connection spec
47          com.ibm.connector2.cics.ECIConnectionSpec connectionSpec = new com.ibm.connector2.cics.ECIConnectionSpec();
48          connectionSpec.setUserName("userid");
49          connectionSpec.setPassword("p0ssword");
50
51          //      Obtain a connection from the connection factory
52          com.ibm.connector2.cics.ECIConnection connection =
53              (com.ibm.connector2.cics.ECIConnection)cf.getConnection(connectionSpec);
54
55          //      Create a interaction for the connection
56          com.ibm.connector2.cics.ECIInteraction interaction =
57              (com.ibm.connector2.cics.ECIInteraction) connection.createInteraction();
58
59          //      Execute the interaction
60          interaction.execute(interSpec, commarea, commarea);
61
```

MQ client

PutMessage.java

```
109
110             // Create a connection to the queue manager
111
112             if (bindingType.equals("client")) {
113                 // Set MQ Environment variables/parameters
114                 // qManager = "QMZB";
115                 if (qManager.equals("QMZB")) {
116                     com.ibm.mq.MQEnvironment.hostname = "mpx3";
117                     com.ibm.mq.MQEnvironment.channel = "Client.to.QMZB";
118                     com.ibm.mq.MQEnvironment.port = 1416;
119
120                 if (qManager.equals("WMQ8")) {
121                     com.ibm.mq.MQEnvironment.hostname = "localzos";
122                     com.ibm.mq.MQEnvironment.channel = "Client.to.LOCALZOS";
123                     com.ibm.mq.MQEnvironment.port = 1436;
124                 }
125             }
126             com.ibm.mq.MQEnvironment.userID = userName;
127             com.ibm.mq.MQEnvironment.password = passWord;
128
129             qMgr = new MQQueueManager(qManager);
130             System.out.println("Success:  Obtained a " + bindingType + " reference to the Queue manager: "+ qManager);
131
132             // Set up the options on the queue we wish to open...
133             // Note. All WebSphere MQ Options are prefixed with MQC in Java.
134
135             int openOptions = MQC.MQOO_INPUT_AS_Q_DEF | MQC.MQOO_OUTPUT;
136
137             // Now specify the queue that we wish to open,
138             // and the open options...
139
140             system_default_local_queue = qMgr.accessQueue(queueName,openOptions);
141             System.out.println("Success:  Obtained a reference to the Queue: "  + queueName);
142
143             sendMessage();
144         }
145         catch (MQException ex)
146         {
147           System.out.println("A WebSphere MQ error occurred : Completion code " +
148                              ex.completionCode + " Reason code " + ex.reasonCode);
149           if (ex.reasonCode == EMPTY_QUEUE)
150           {
```

One goal for REST simplicity is to have client code that is unaware of the infrastructure

ZceeCICSGet.java

```
55          // Invoke the REST API using a GET method
56          URL url = new URL("https://wg31.washington.ibm.com:9453/cscvinc/employee/" + args[1]);
57          System.out.println("URL: " + url);
58          HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
59          conn.setRequestMethod("GET");
60          conn.setRequestProperty("Content-Type", "application/json");
61          byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
62          conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
63          try {
64              if (conn.getResponseCode() != 200) {
65                  throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
```

ZceeDb2Get.java

```
52          // Invoke the REST API using a GET method
53          URL url = new URL("https://wg31.washington.ibm.com:9453/db2/employee/"+args[1]);
54          System.out.println("URL: " + url);
55          HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
56          conn.setRequestMethod("GET");
57          conn.setRequestProperty("Content-Type", "application/json");
58          byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
59          conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
60          try {
61              if (conn.getResponseCode() != 200) {
62                  throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
```

These Java clients have the same application programming pattern.

- They provide an URL and a method and provide a request message.
- Then use HTTP protocol to send a request to the API provider.
- A response message is returned after accessing a remote resource, e.g., a CICS program, a Db2 table, an IMS transaction or a MQ queue. The client application is unaware of the underlying infrastructure. No dependencies on coding for ECI, JDBC, OTMA, JMS, J2C, etc.

```
URL: https://wg31.washington.ibm.com:9453/mqapi/
NAME:   TINA J YOUNG
NUMB:   001781
ADDRX:  SINDELFINGEN,GERMANY
PHONE:  70319990
DATEX:  21 06 77
AMOUNT: $0009.99
```

MQ

```
lastName:  LAST1
firstName: FIRST1
zipCode:   D01/R01
extension: 8-111-1111
message:   ENTRY WAS DISPLAYED
HTTP code: 200
```

IMS

mitchj@us.ibm.com

/oai/open_api_initiative

The industry standard framework for describing REST APIs

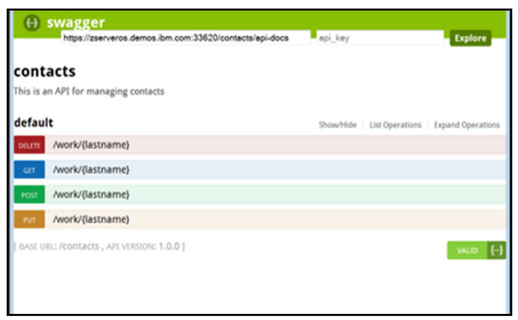
The OpenAPI Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how APIs are described. As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral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

mitchj@us.ibm.com

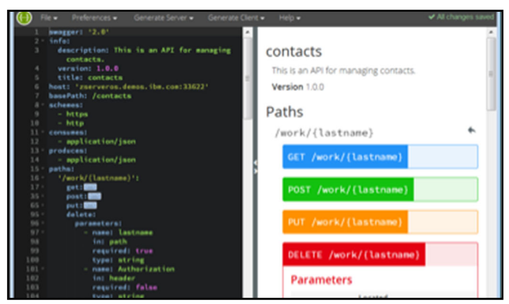
There are a number of tools available to aid in the creation and consumption of APIs

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 specifications.

mitchj@us.ibm.com

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 OpenAPI 2/OpenAPI 3 specification 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).

cscvinc.json - Notepad

File Edit Format View Help

```
{
    "swagger": "2.0",
    "info": {
        "description": "",
        "version": "1.0.0",
        "title": "cscvincapi"
    },
    "basePath": "/cscvincapi",
    "schemes": [
        "https",
        "http"
    ],
    "consumes": [
        "application/json"
    ],
    "produces": [
        "application/json"
    ],
    "paths": {
        "/employee/{employee}": {
            "get": {
                "tags": [
                    "cscvincapi"
                ],
                "operationId": "getCscvincSelectService",
                "parameters": [
                    {
                        "name": "Authorization",
                        "in": "header",
                        "required": false,
                        "type": "string"
                    },
                    {
                        "name": "employee",
                        "in": "path",
                        "required": true,
                        "type": "string",
                        "maxLength": 6
                    }
                ],
                "responses": {
                    "200": {
                        "description": "OK",
                        "schema": {
                            "$ref": "#/definitions/getCscvincSelectService_response_200"
                        }
                    },
                    "404": {
                        "description": "Not Found",
```

Ln 18, Col 7 100% Windows (CRLF) UTF-8

cscvinc.yaml - Notepad

File Edit Format View Help

```
openapi: 3.0.1
info:
  title: cscvinc
  description: ""
  version: 1.0.0
servers:
- url: /cscvinc
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
      - cscvinc
      operationId: postCscvincInsertService
      x-ibm-zcon-roles-allowed:
        - Staff
      parameters:
      - name: Authorization
        in: header
        schema:
          type: string
      requestBody:
        description: request body
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/postCscvincInsertService_request'
        required: true
      responses:
        200:
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/postCscvincInsertService_response_200'
      x-codegen-request-body-name: postCscvincInsertService_request
  /employee/{employee}:
    get:
      tags:
      - cscvinc
      operationId: getCscvincSelectService
      x-ibm-zcon-roles-allowed:
        - Staff
      parameters:
      - name: Authorization
        in: header
        schema:
          type: string
```

Ln 44, Col 16 100% Unix (LF) UTF-8

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

What value does zOS Connect Add?

Truly RESTful APIs to and from your mainframe.

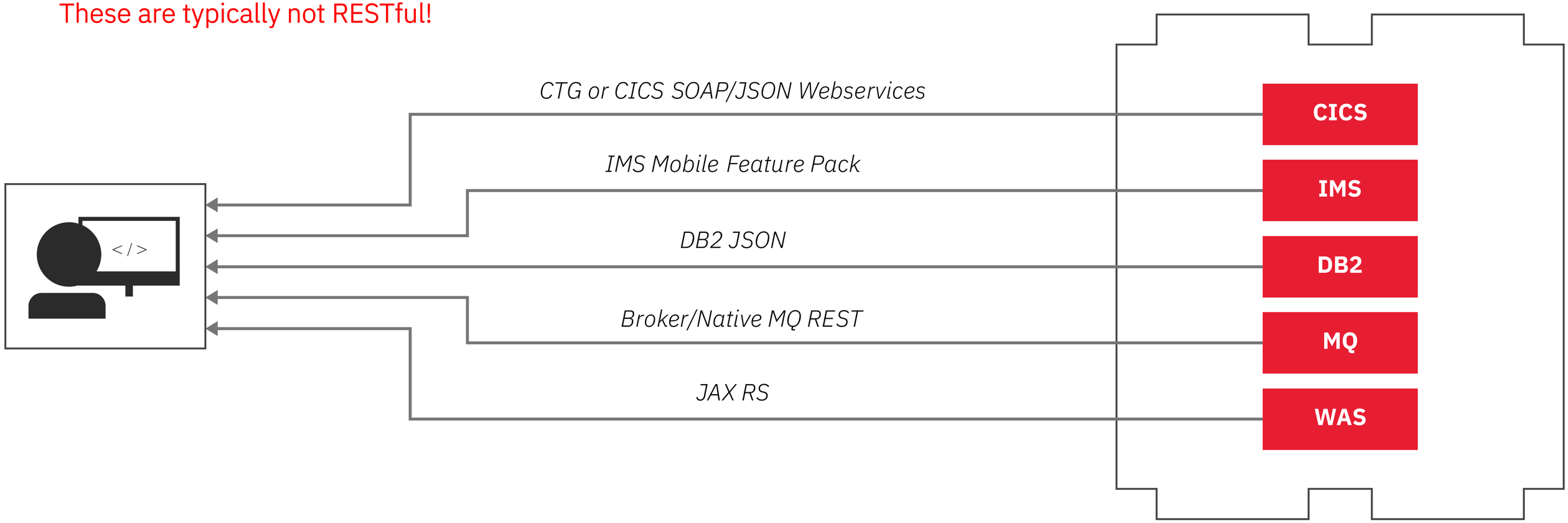
mitchj@us.ibm.com

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 API gateway entry point

- And exposes z/OS resources without writing or changing 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.

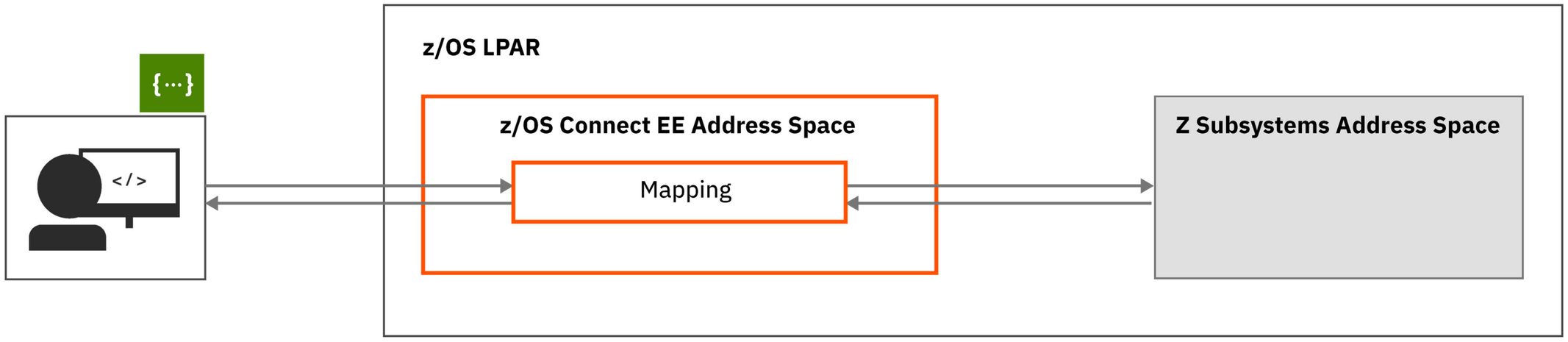
mitchj@us.ibm.com

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

mitchj@us.ibm.com

Data mapping/transformation

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

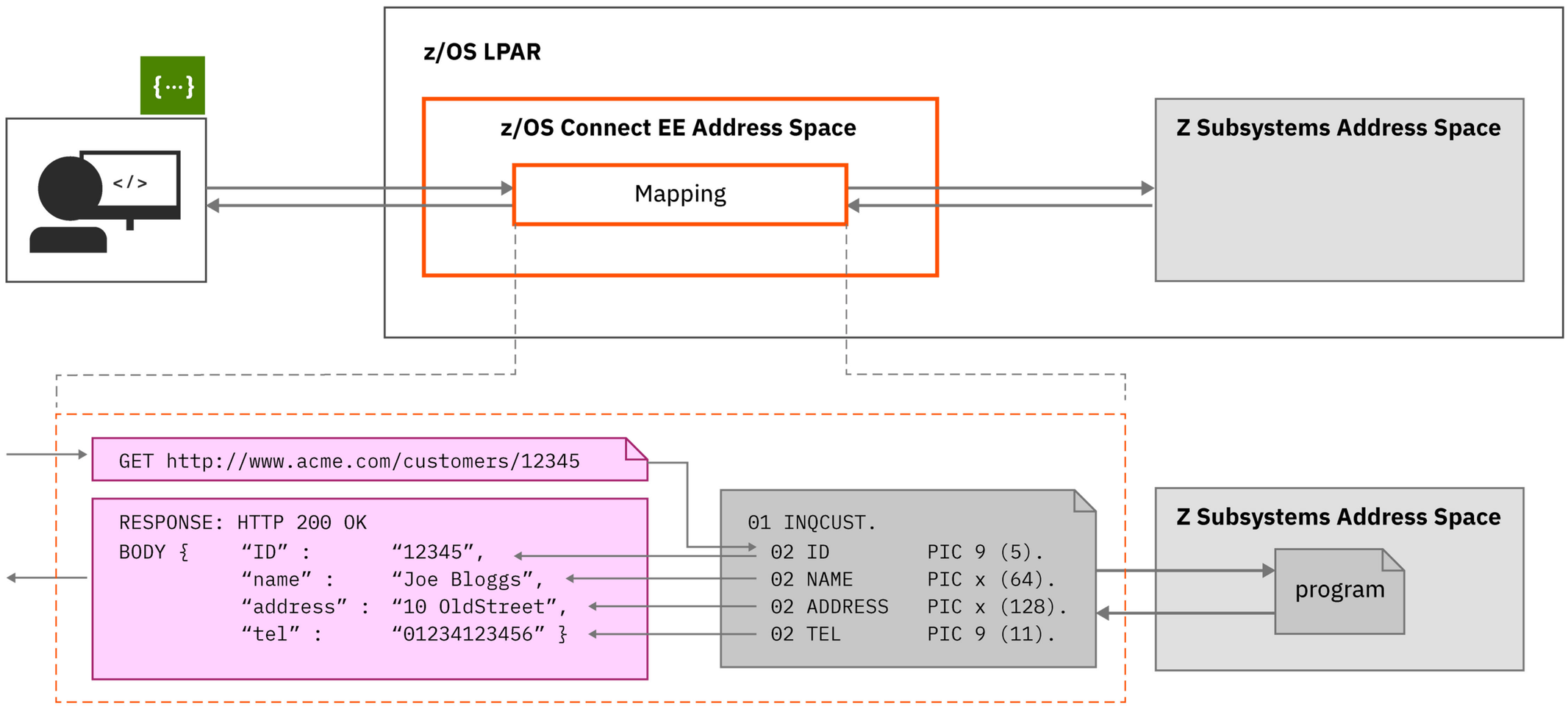


* Most z/OS subsystems depend information in a serial data format and do not normally work with JSON request/response messages. Examples of serialized messages are CICS COMMAREAs and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation refers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

mitchj@us.ibm.com

A data mapping and transformation example

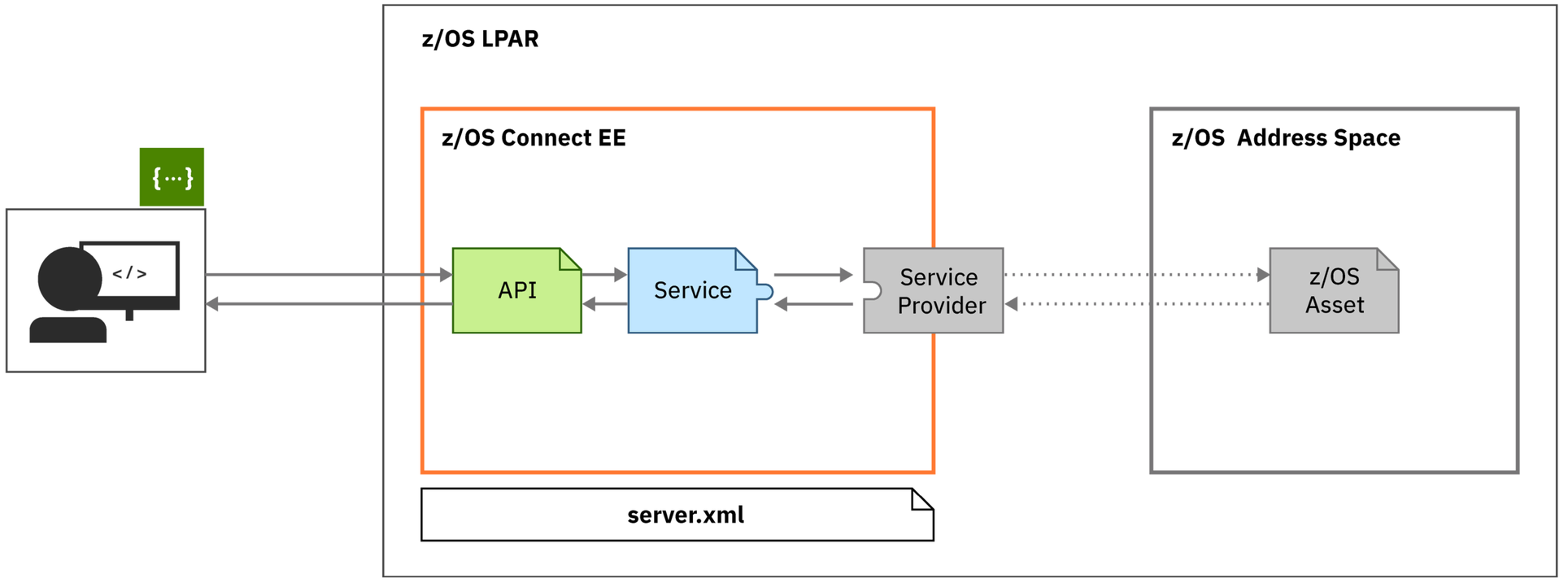
- A closer look at the transformation process



mitchj@us.ibm.com

Accessing an z/OS Asset using an API with Open API 2

- z/OS Connect OpenAPI 2 APIs are developed using an Eclipse based API Toolkit to develop Service Archive (SAR) and API ARchive (AAR) files.

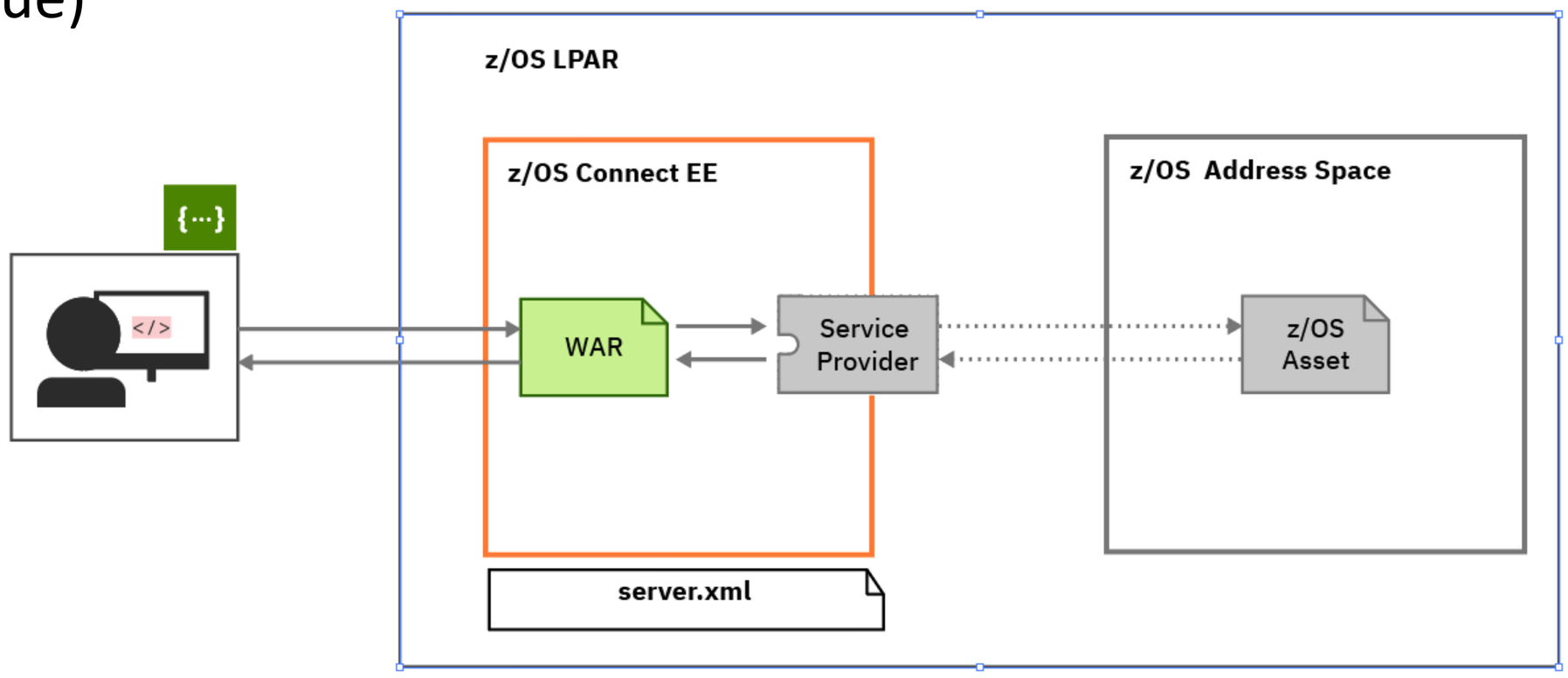


- The API is ready to be consumed and requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port,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

mitchj@us.ibm.com

Accessing a z/PS asset using an API with Open API 3

- z/OS Connect OpenAPI 3 APIs are developed using a z/OS Connect Designer web browser tool to develop 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)

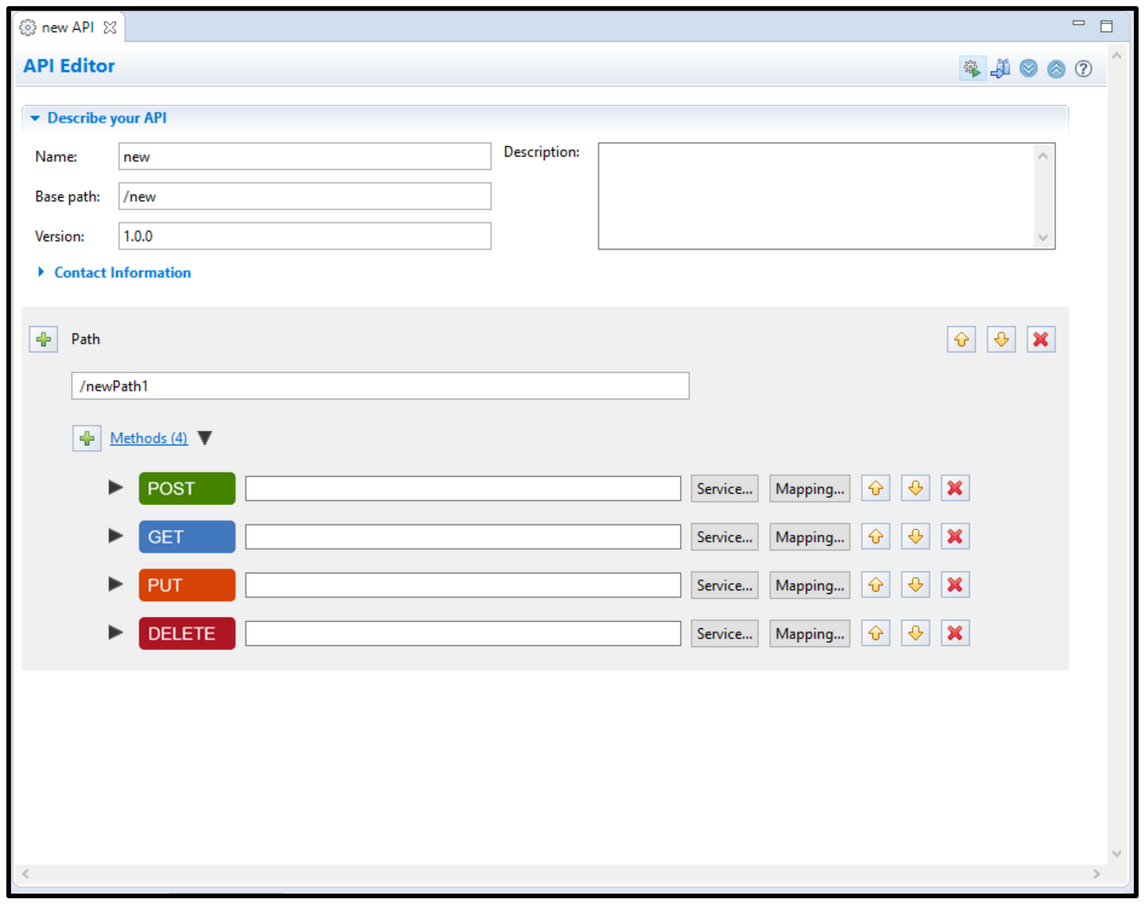
mitchj@us.ibm.com

z/OS Connect OpenAPI Tooling

mitchj@us.ibm.com

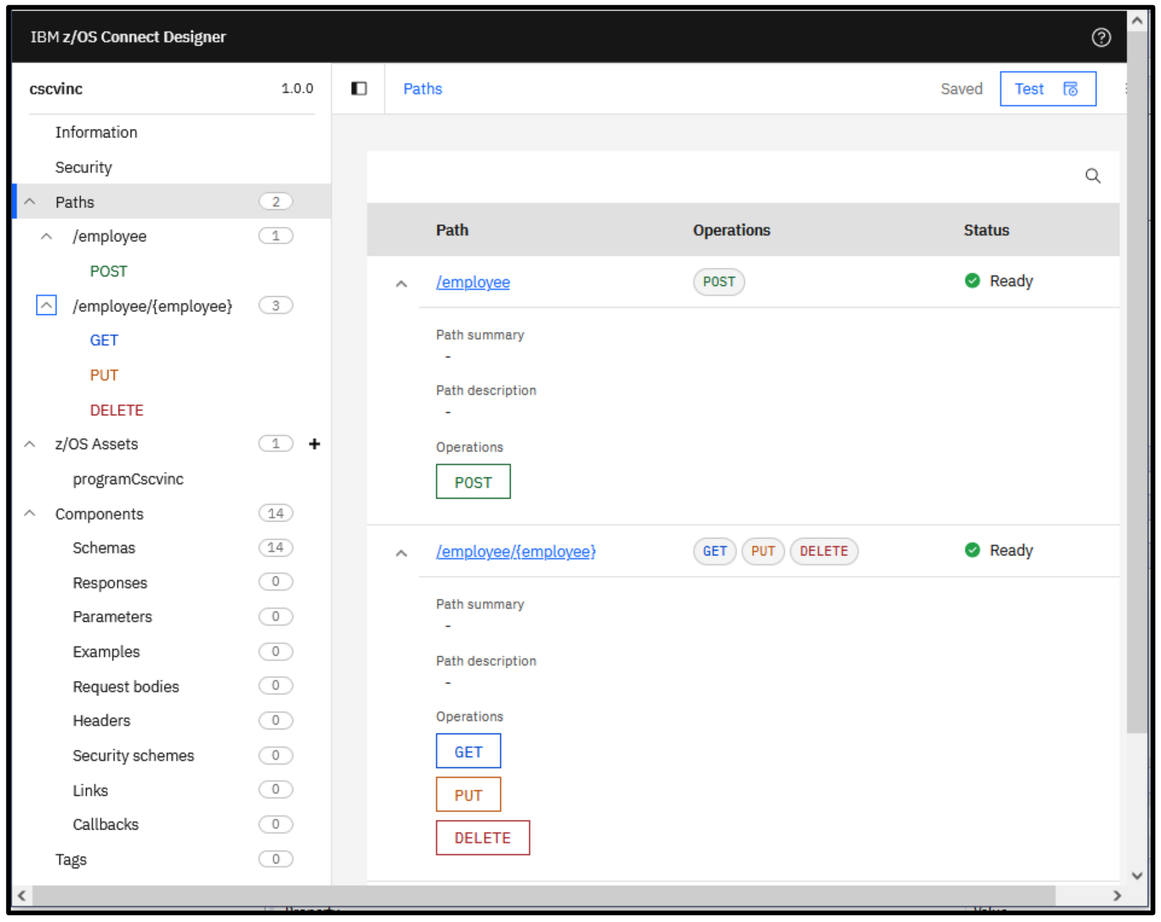
z/OS Connect’s OpenAPI 2 Palette versus the OpenAPI 3 API Designer

z/OS Connect API Toolkit (Eclipse)



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

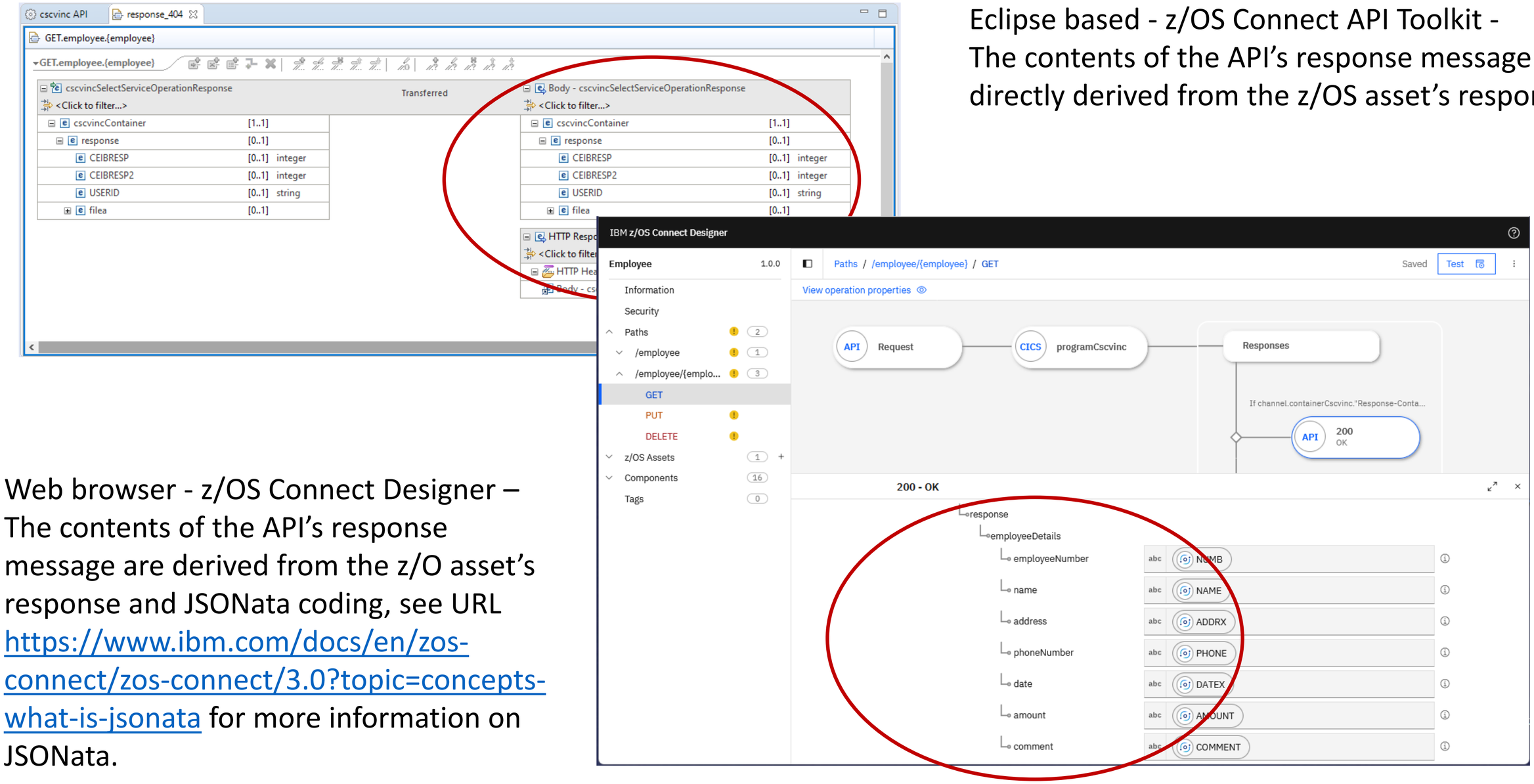
z/OS Connect Designer (Designer Container)



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

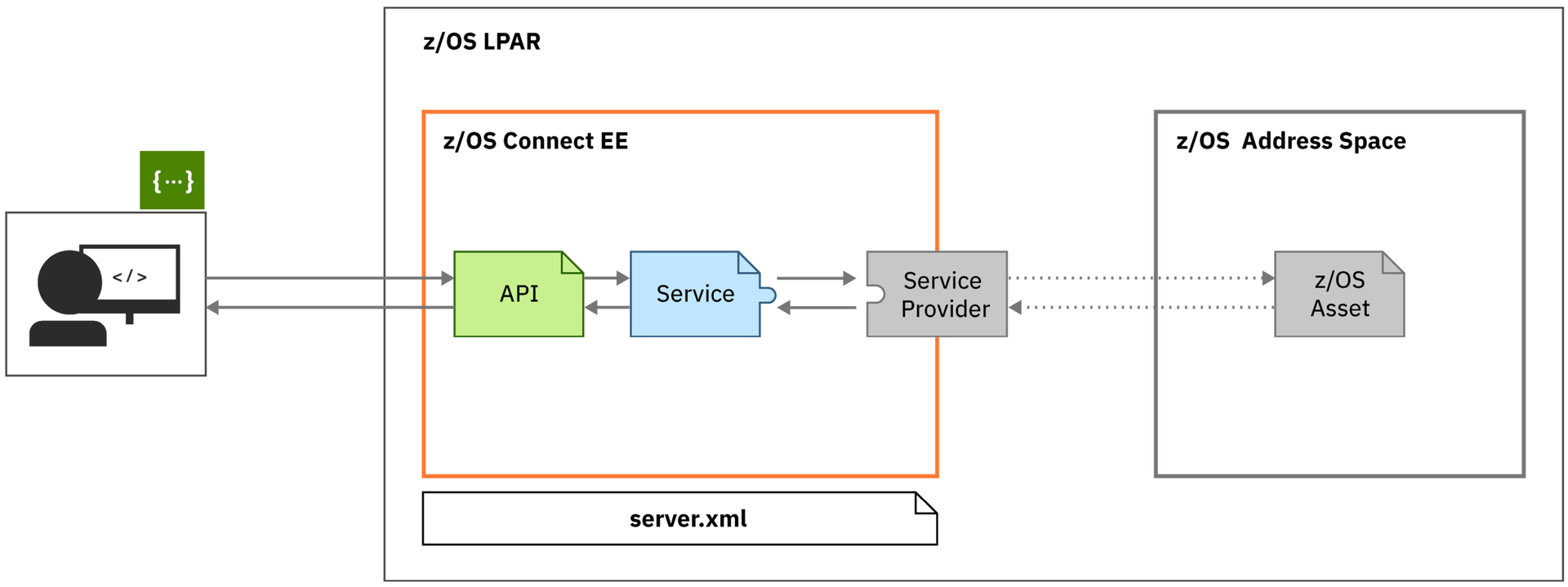
OpenAPI 2

Creating **Services** 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)

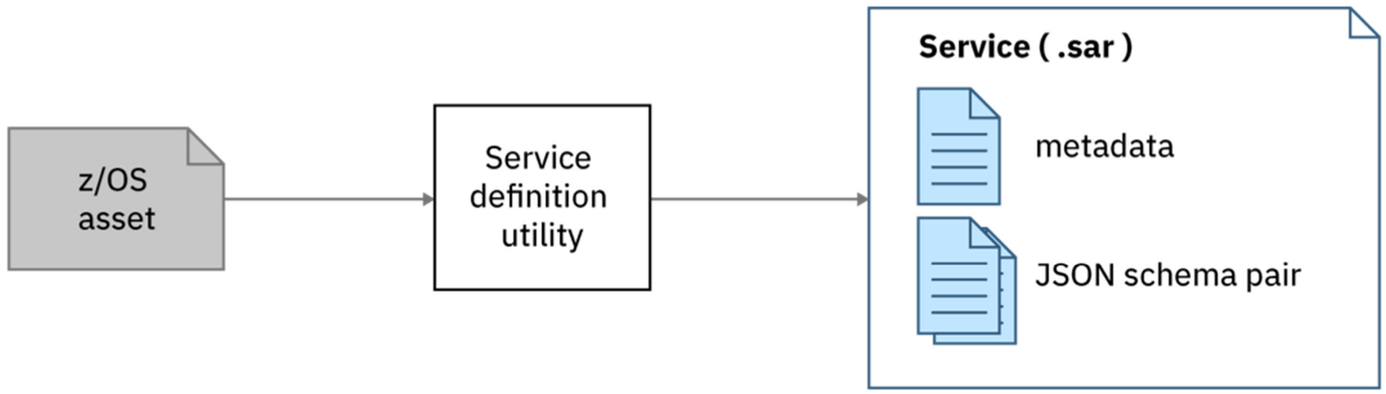
mitchj@us.ibm.com

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



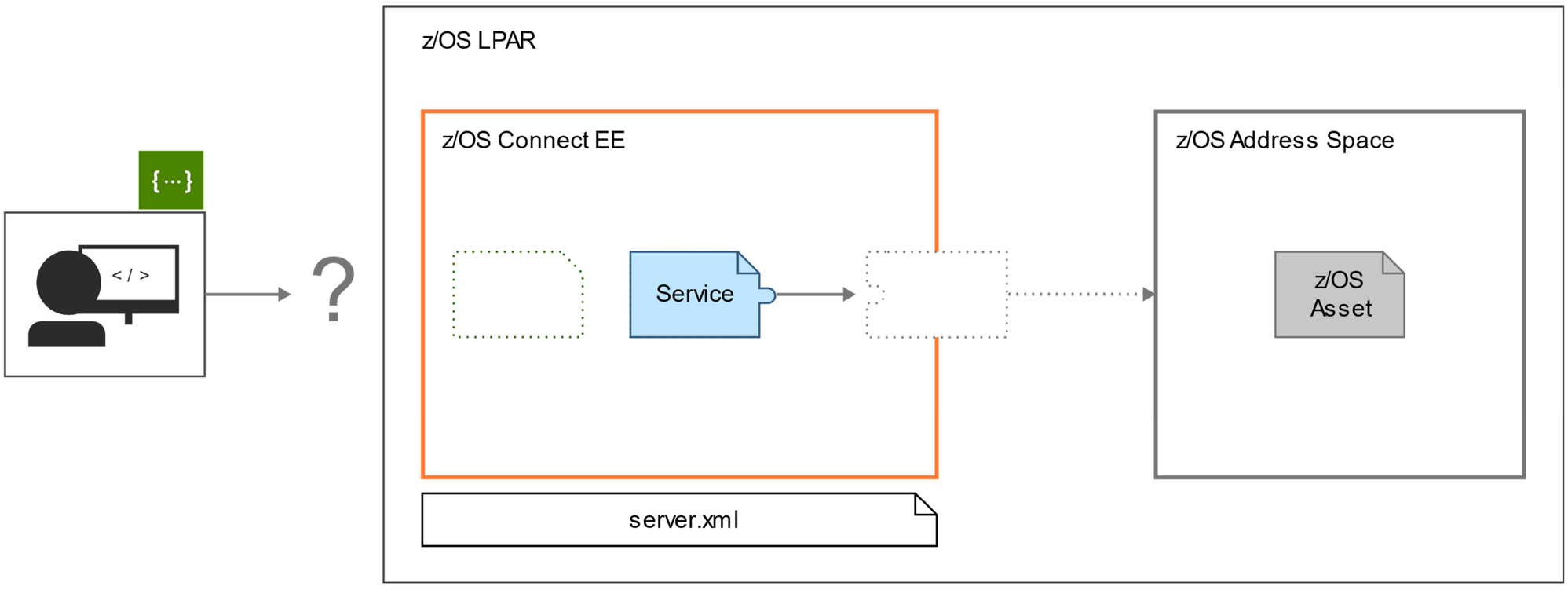

```
        EVALUATE ACTION of Request-Container
            WHEN 'D'
              PERFORM Delete-Record
            WHEN 'I'
              PERFORM Insert-Record
            WHEN 'U'
              PERFORM Update-Record
            WHEN 'S'
              PERFORM Select-Record
        END-EVALUATE.
```

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

Deploy and export the service archive (OpenAPI 2)

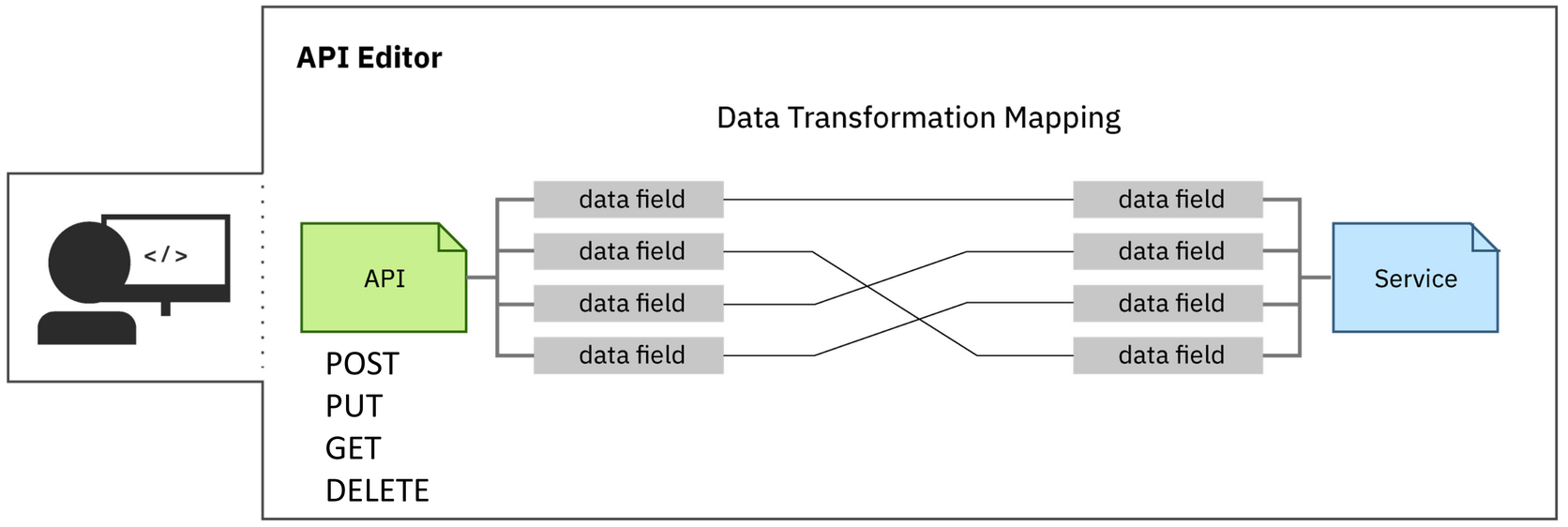
Deploy and export the service archive file



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)

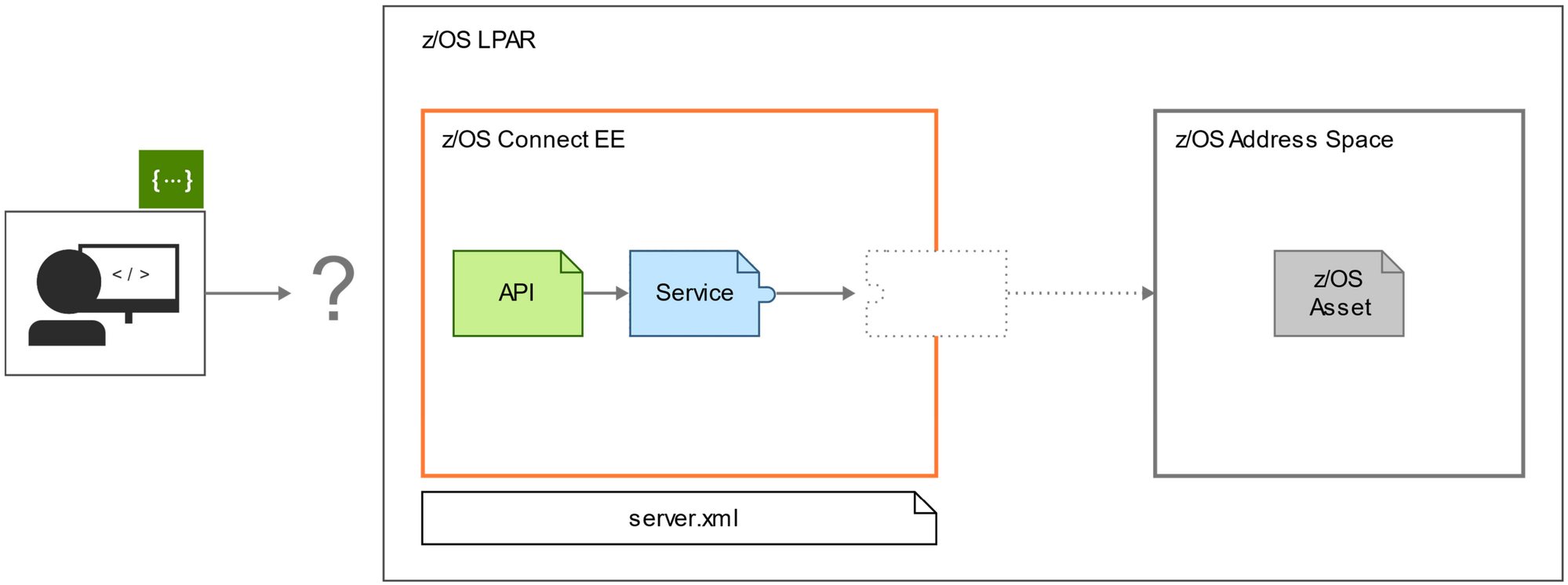
Export the service and then import it to create an API that consumes the service



- 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.

Deploy the API (Open API 2)

Deploy the API archive file

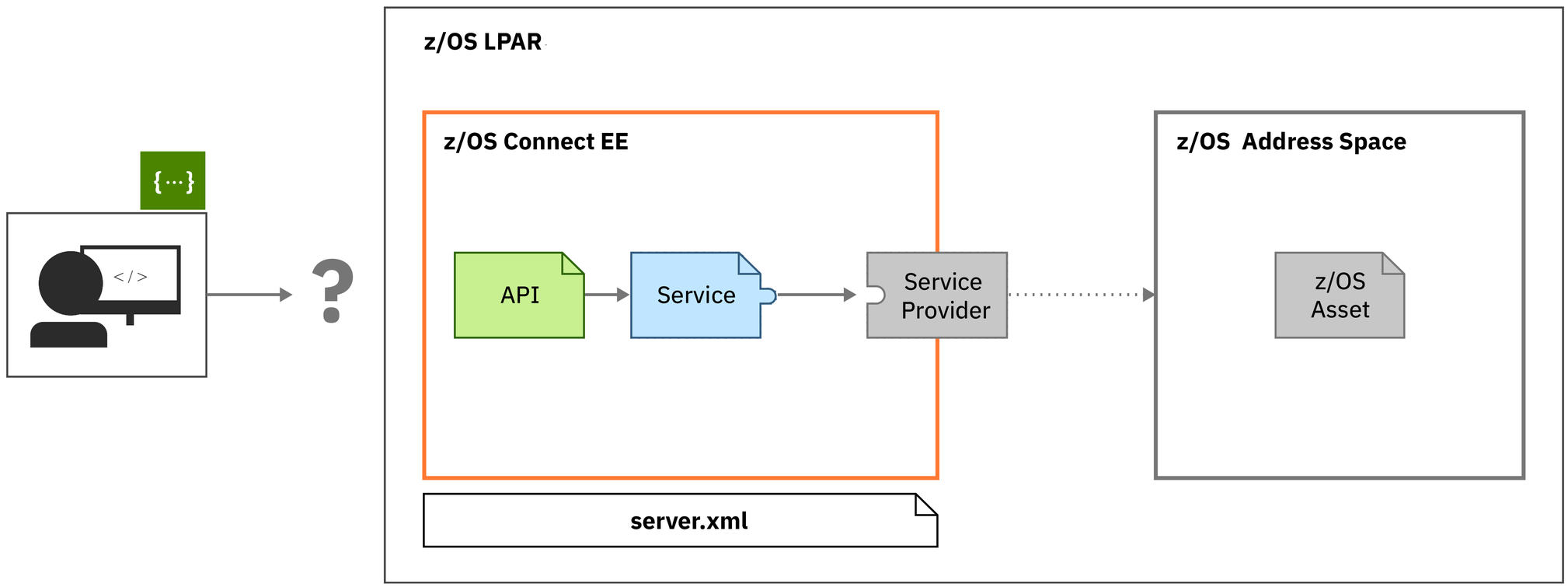


Deploy your API using the right-click deploy in **the API toolkit**

mitchj@us.ibm.com

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

Configure the service provider

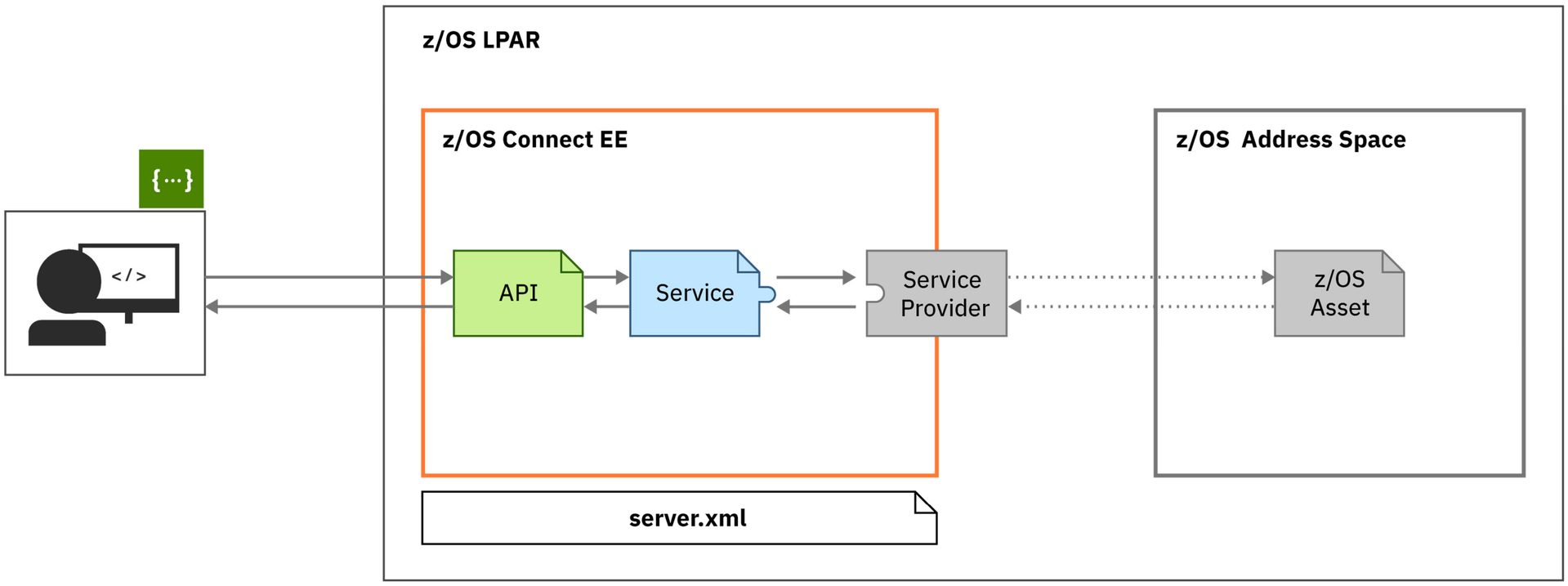


Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.

mitchj@us.ibm.com

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

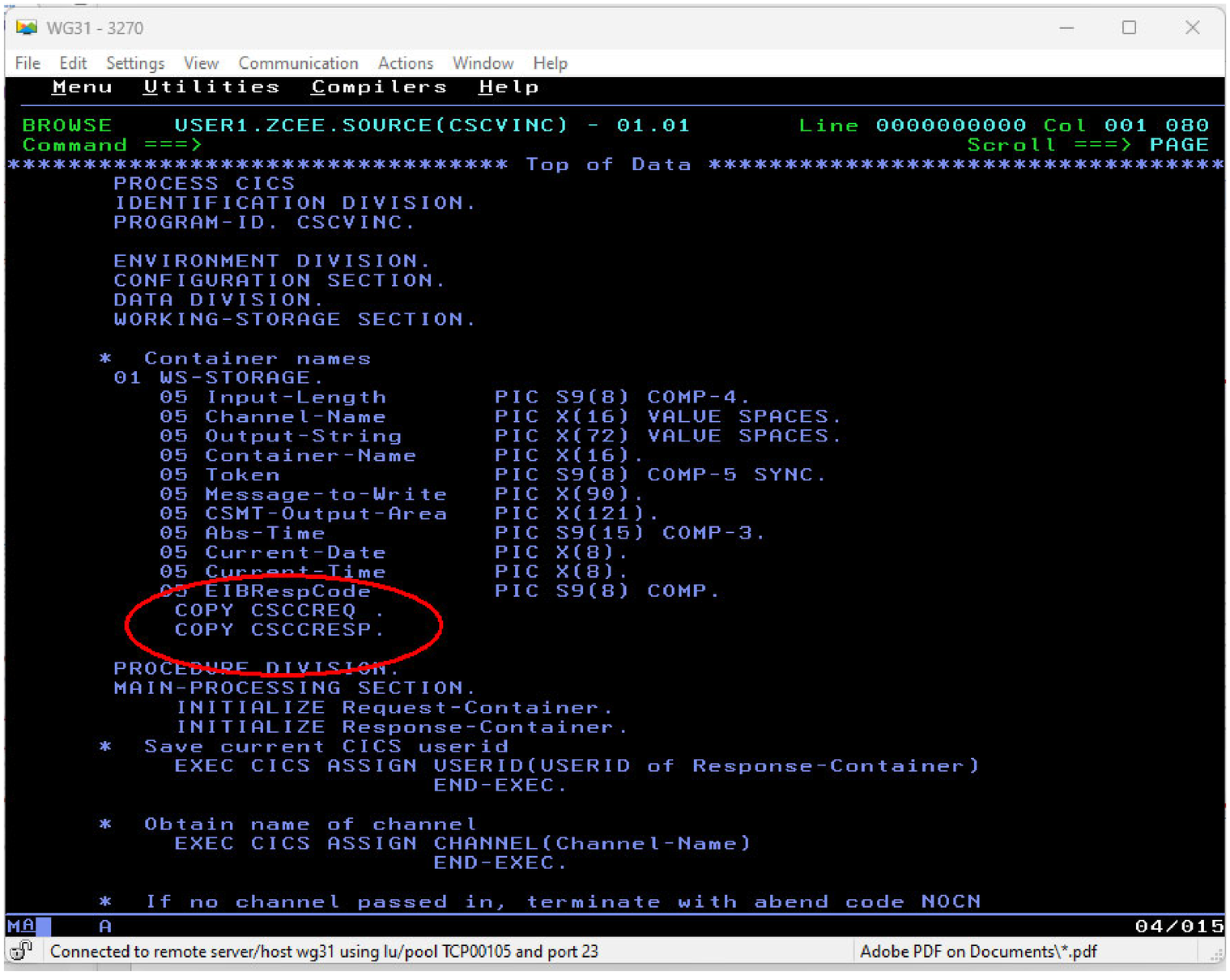
Done



- 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)

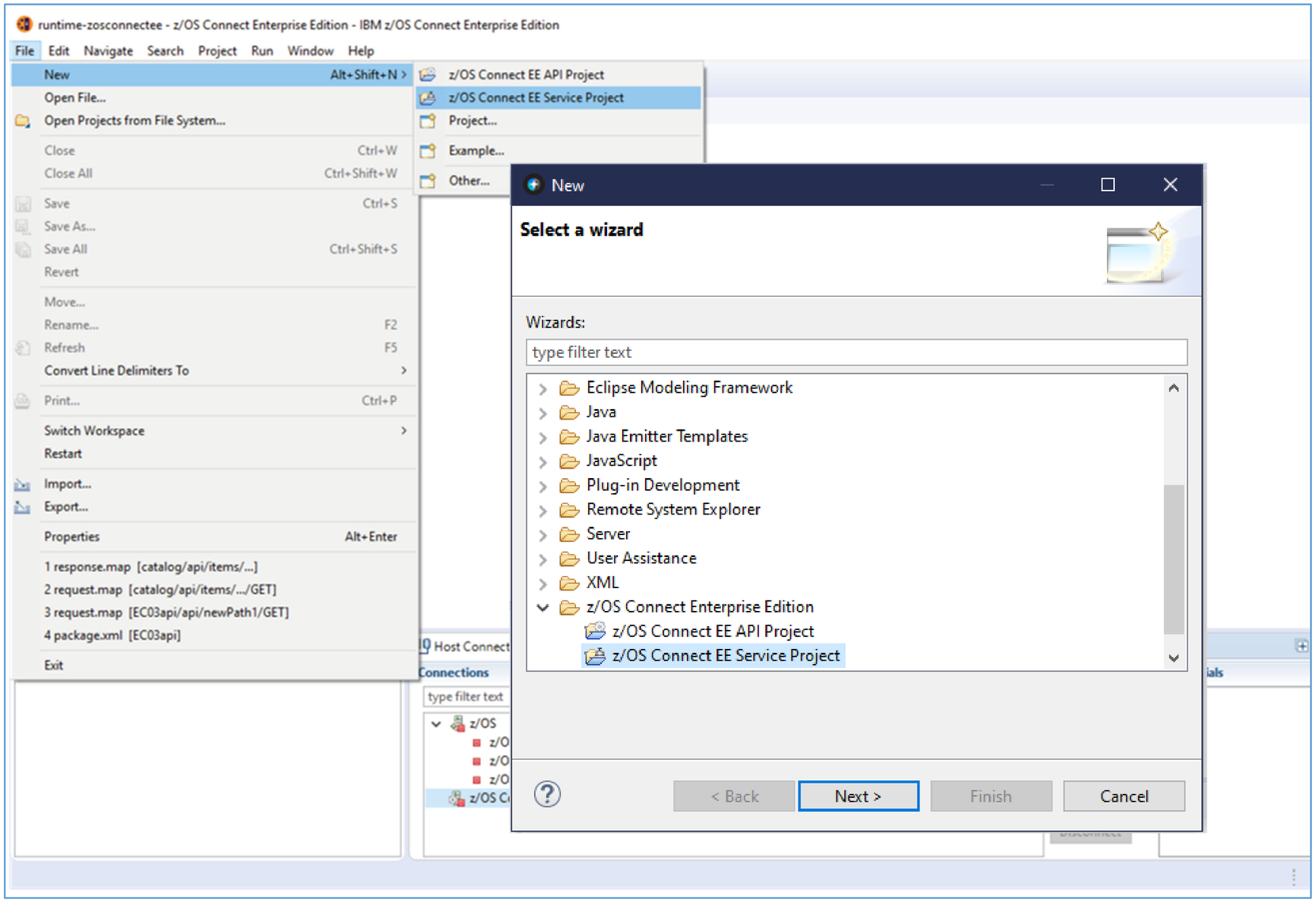
mitchj@us.ibm.com

Start by identifying the format of the messages (e.g., copy books) used by the application



mitchj@us.ibm.com

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



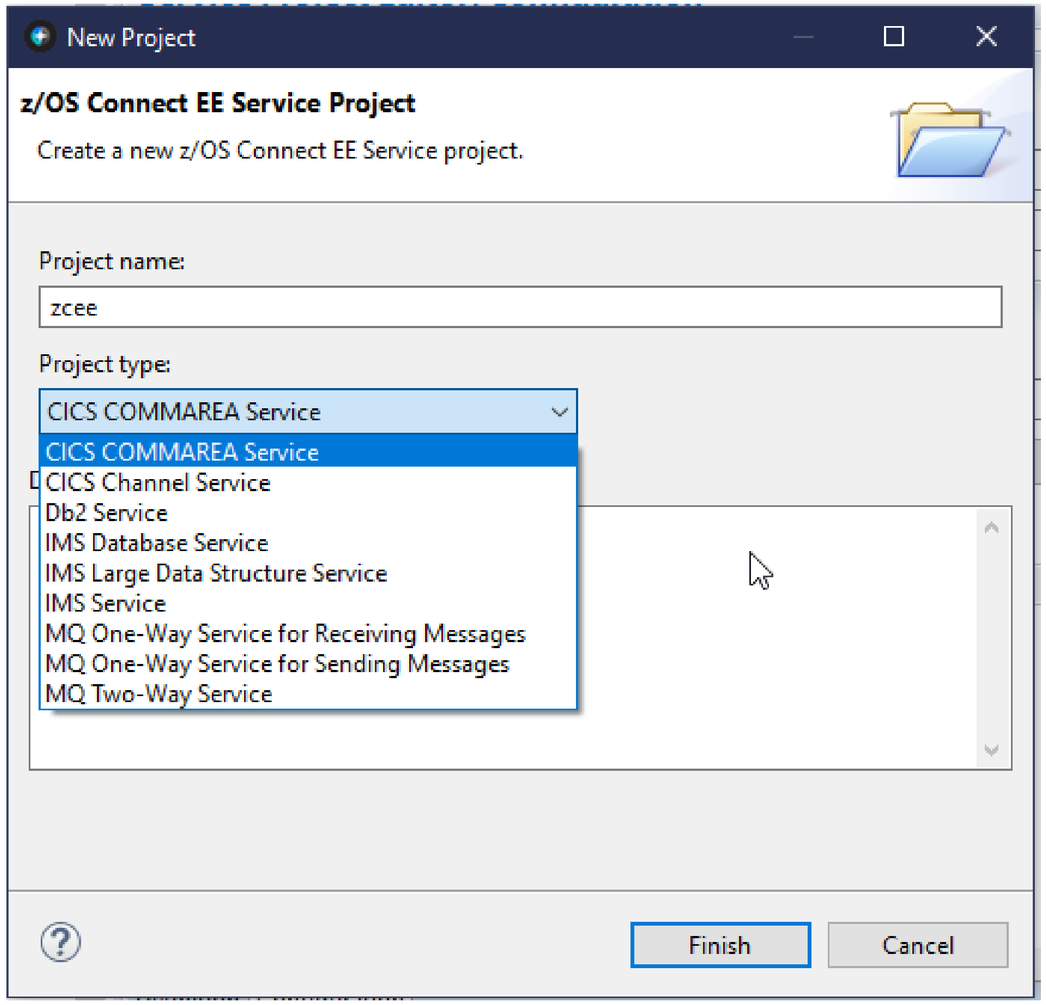
Use the **API toolkit** to create services through Eclipse-based tooling.

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



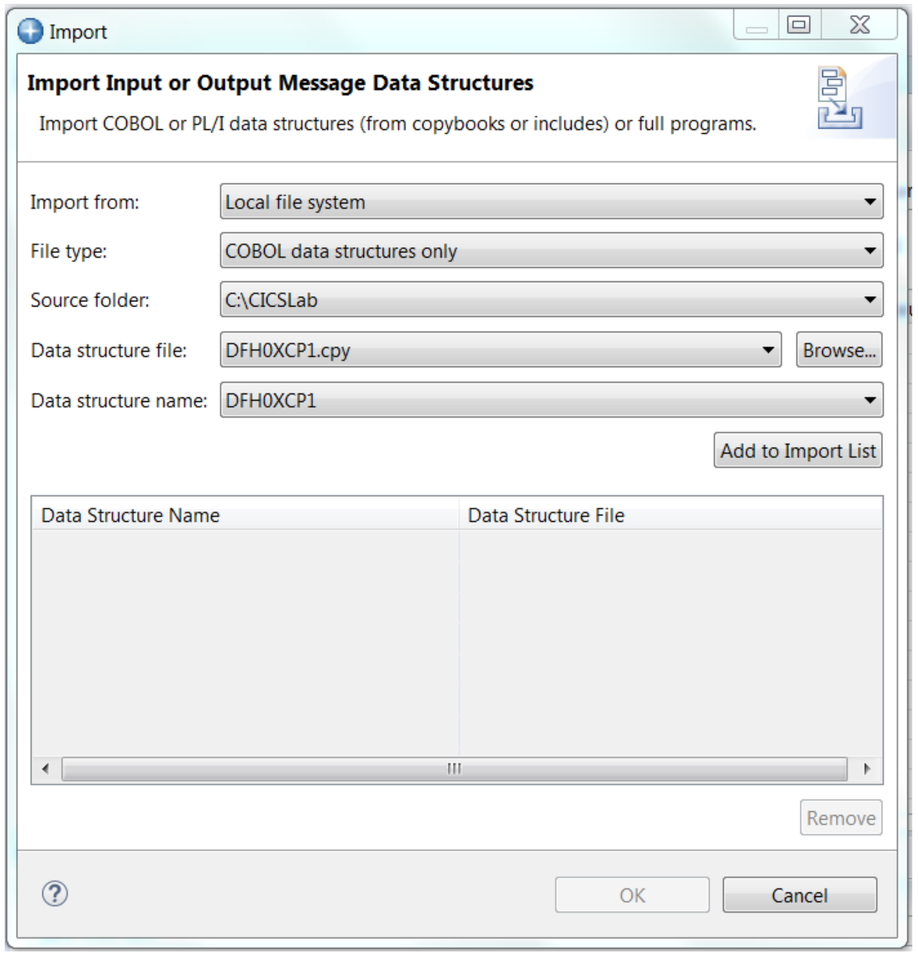
- CICS services that can invoke almost any CICS programs accessed by EXEC CICS LINK request (COMMAREA or CHANNEL). See URL http://www.ibm.com/docs/en/cics-ts/5.6?topic=link-exception-conditions-command for a list of EXEC CICS APIs not allowed in a program when invoked using a CICS Dynamic Program Link 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

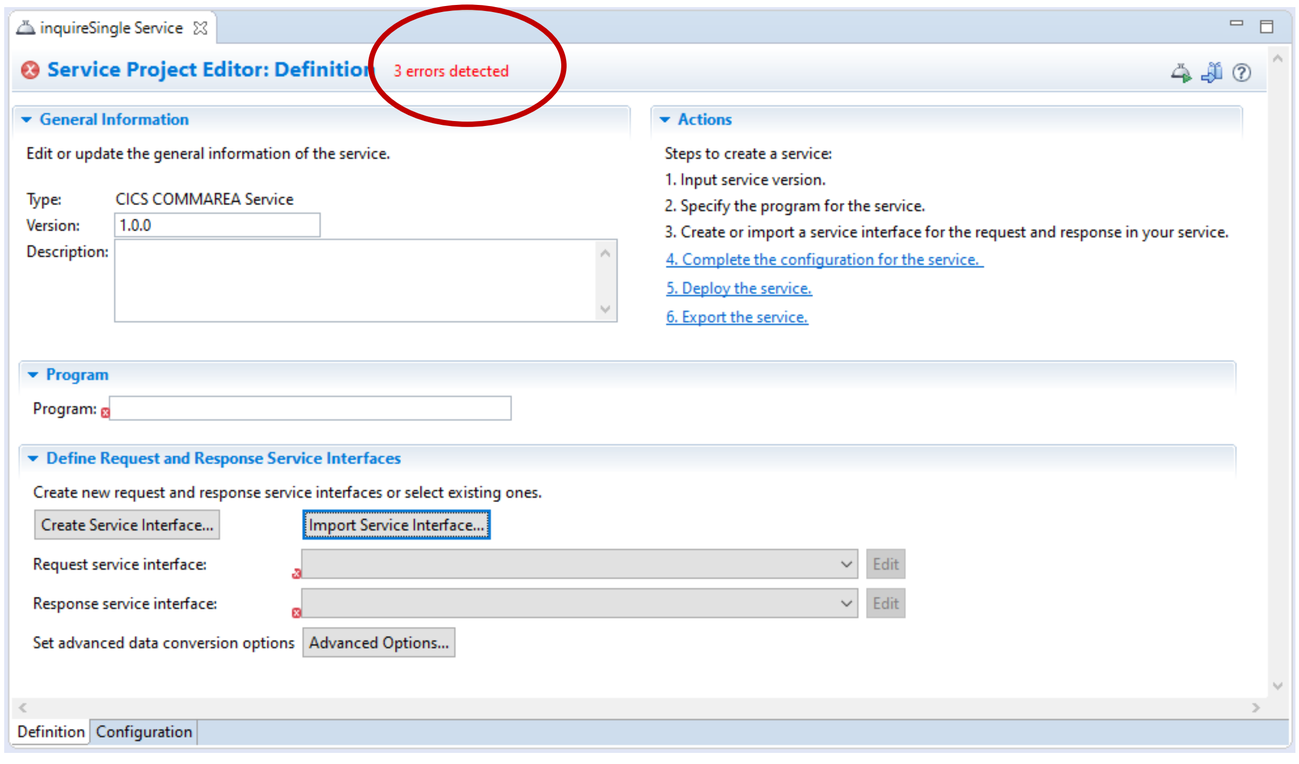
API toolkit – Creating Services for CICS, IMS TM and MQ

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.



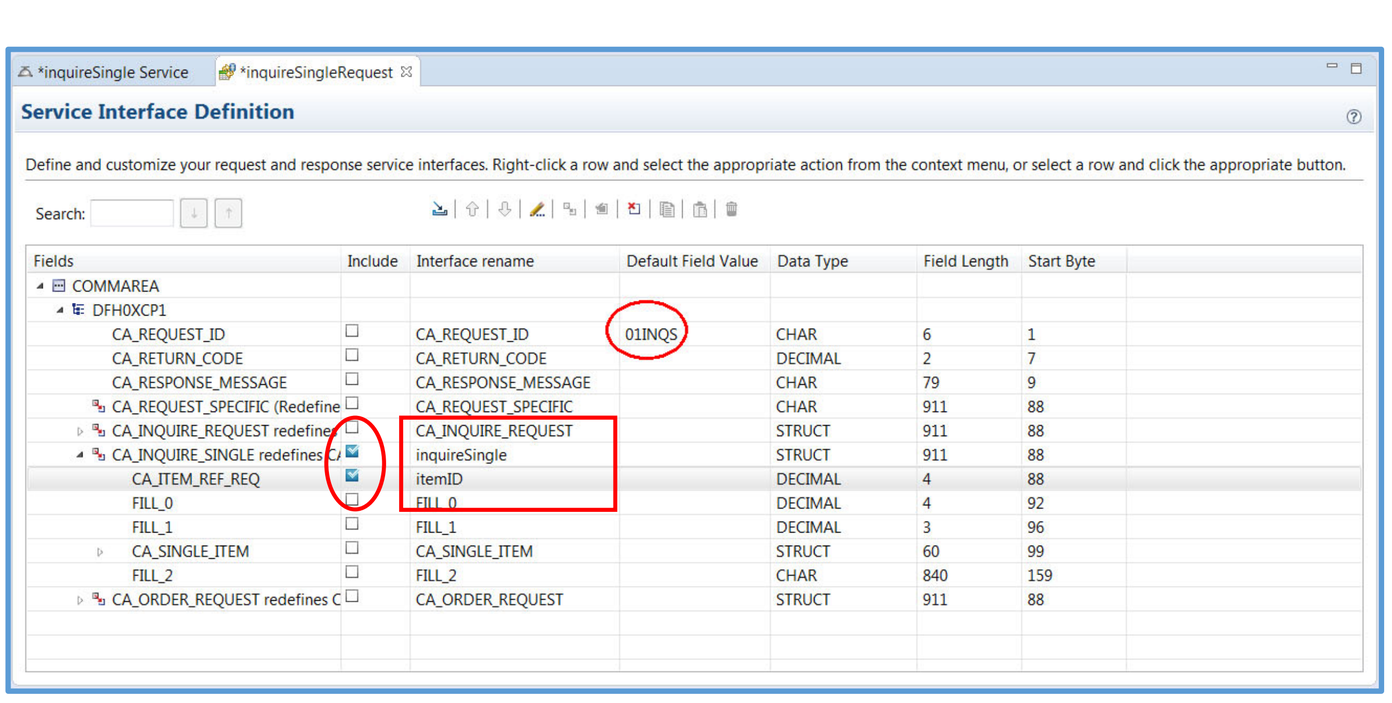
mitchj@us.ibm.com

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

*inquireSingleResponse

Service Interface Definition

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

Search:

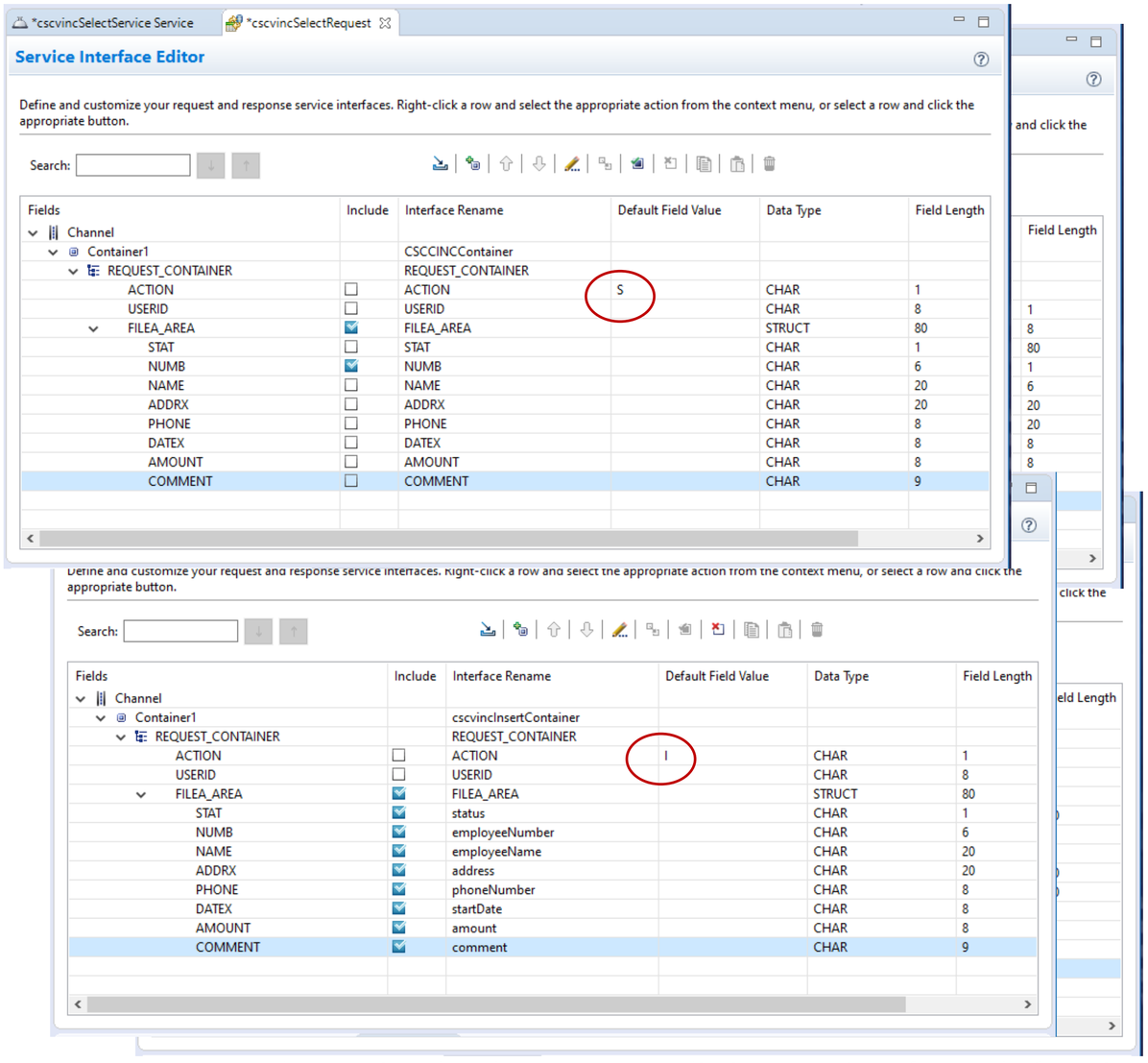
Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID		CHAR	6	1
CA_RETURN_CODE	☑	returnCode		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☑	responseMessage		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefin	☐	CA_REQUEST_SPECIFIC		CHAR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

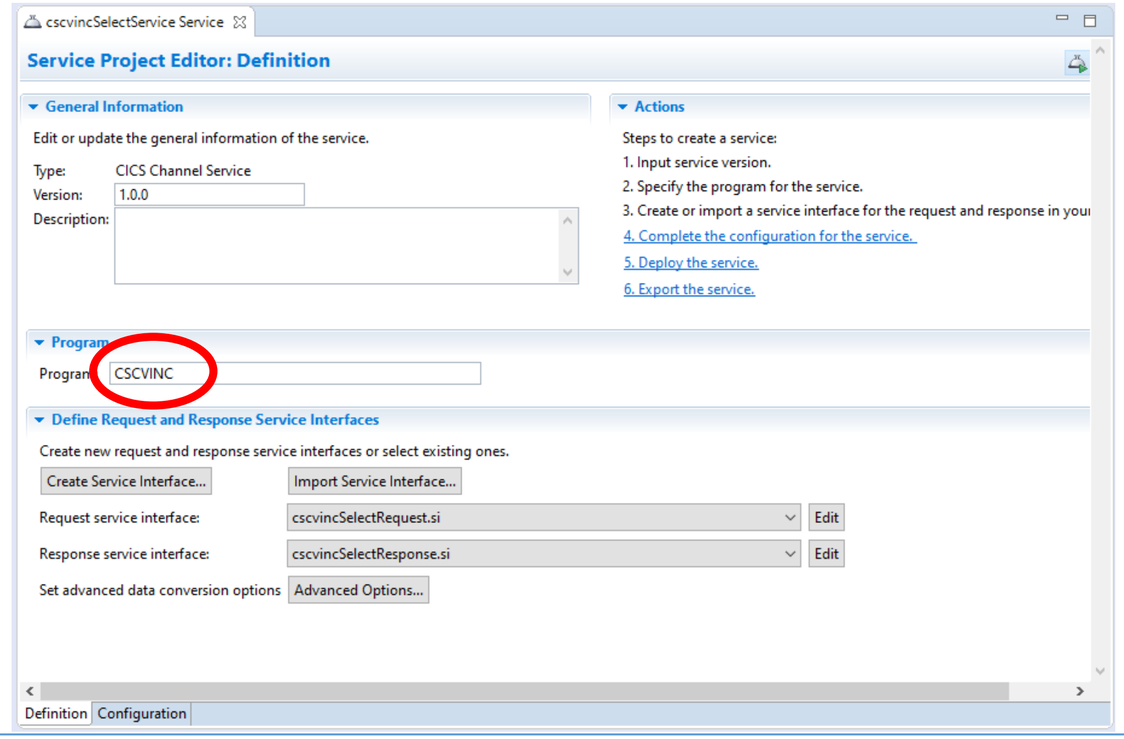
See the imported data structure and can **redact fields** and **rename fields**

mitchj@us.ibm.com

API toolkit – Creating Services for CICS

Creating multiple services definitions to the same resource



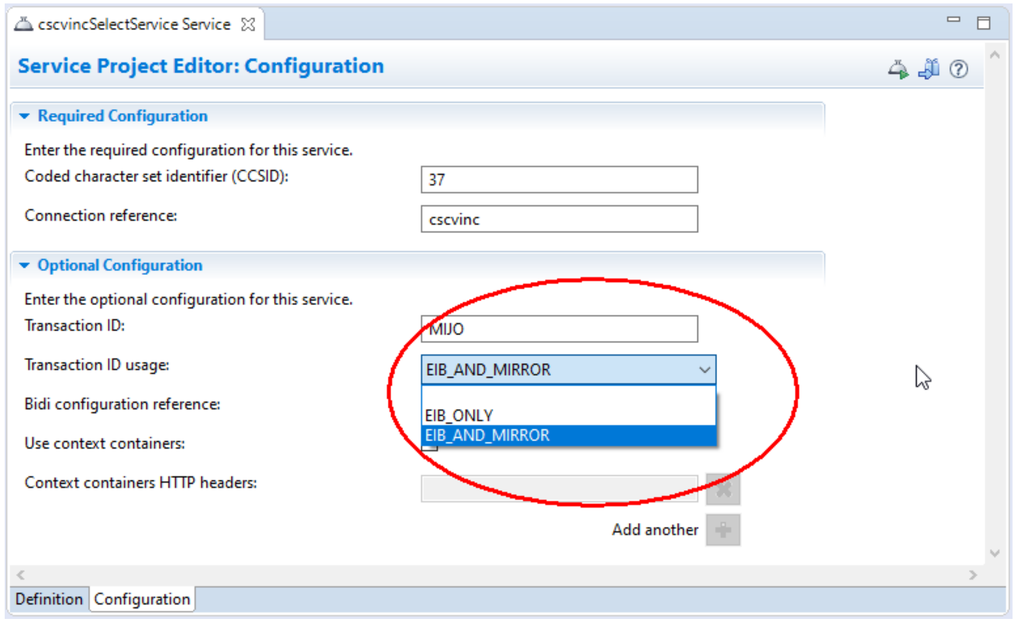



```
EVALUATE ACTION of Request-Container
    WHEN 'D'
      PERFORM Delete-Record
    WHEN 'I'
      PERFORM Insert-Record
    WHEN 'U'
      PERFORM Update-Record
    WHEN 'S'
      PERFORM Select-Record
 END-EVALUATE.
```

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

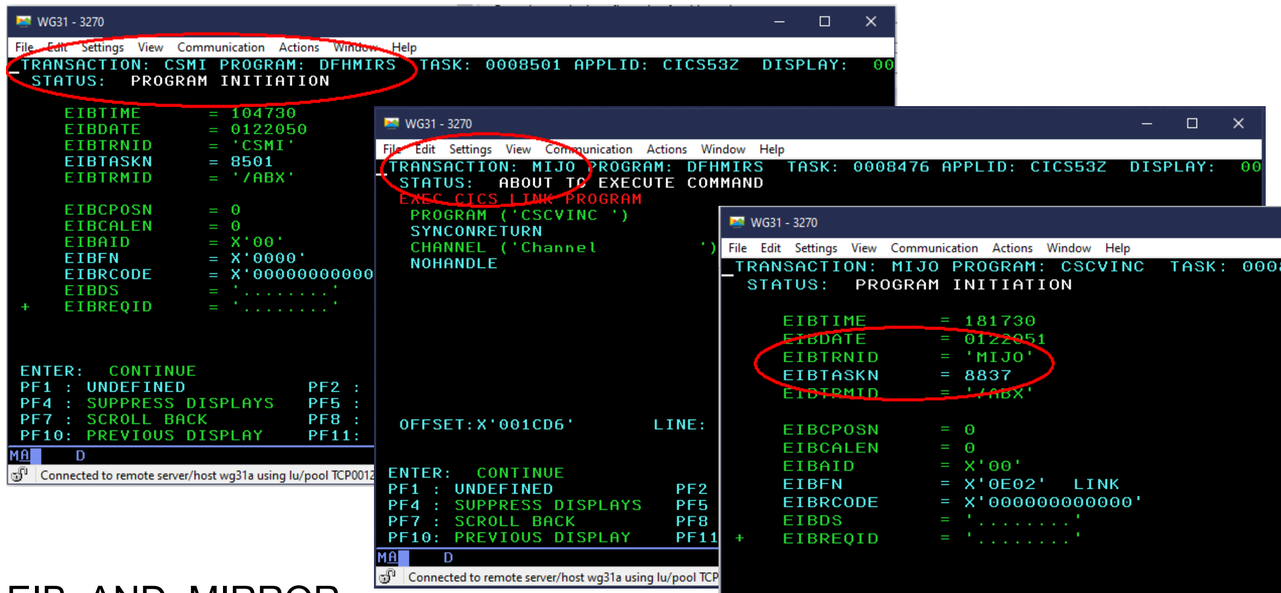
Accessing a CICS program – The significance of Transaction ID Usage



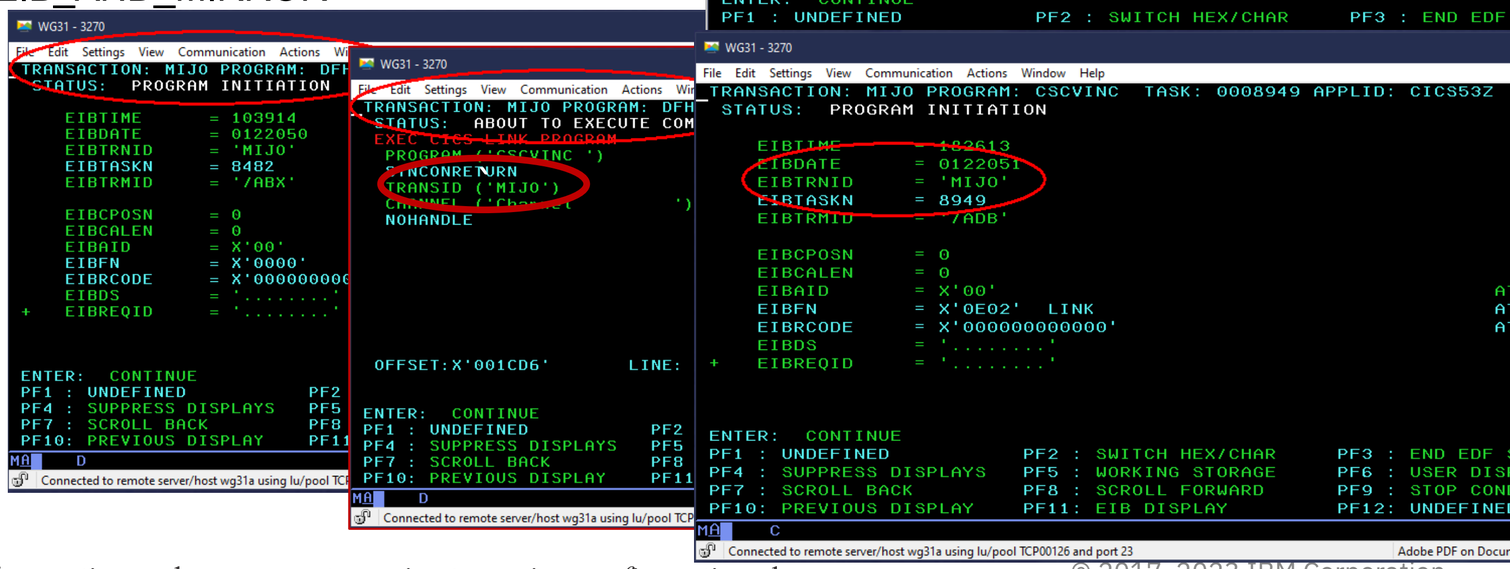
- **Transaction ID** attaches a CICS transaction (CSMI is the default) that starts the CICS DFHMIRS program.

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

EIB_ONLY



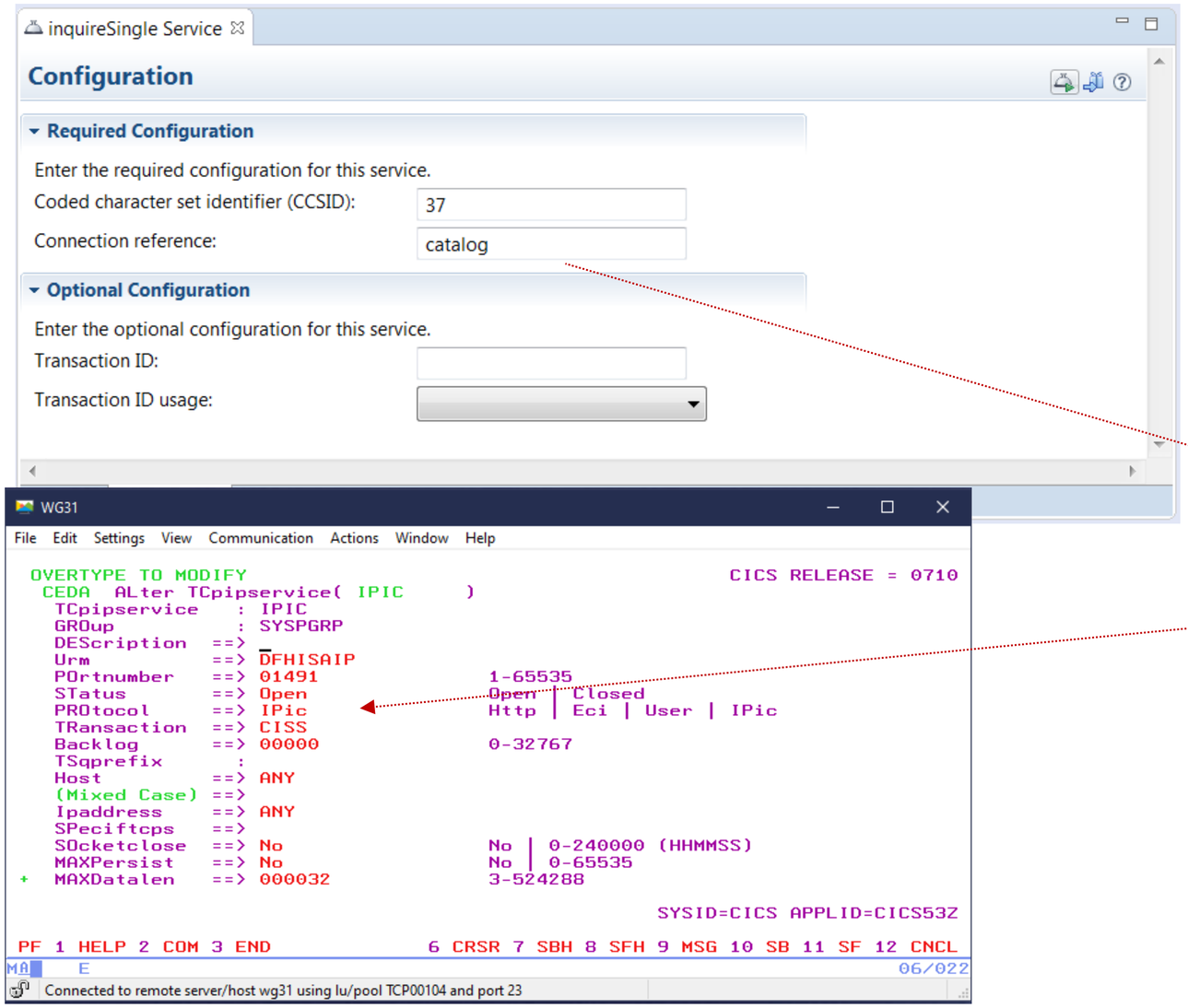
EIB_AND_MIRROR



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

catalog.xml

Design Source

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

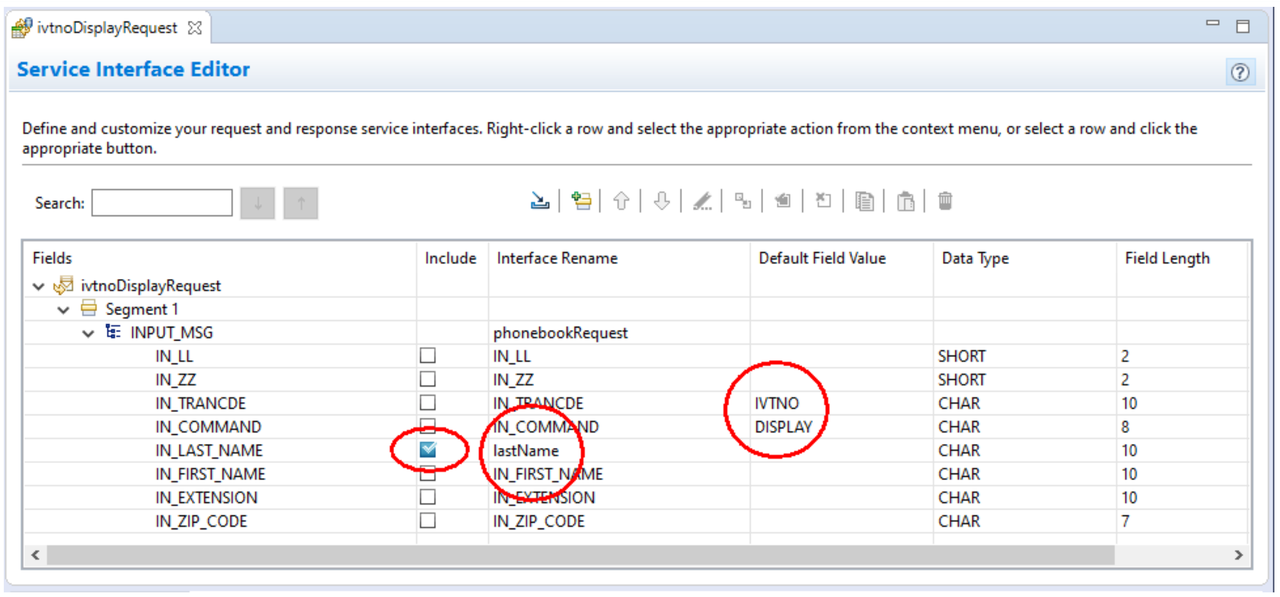
Define IPIC connection to CICS

mitchj@us.ibm.com

API toolkit – Creating Services for IMS

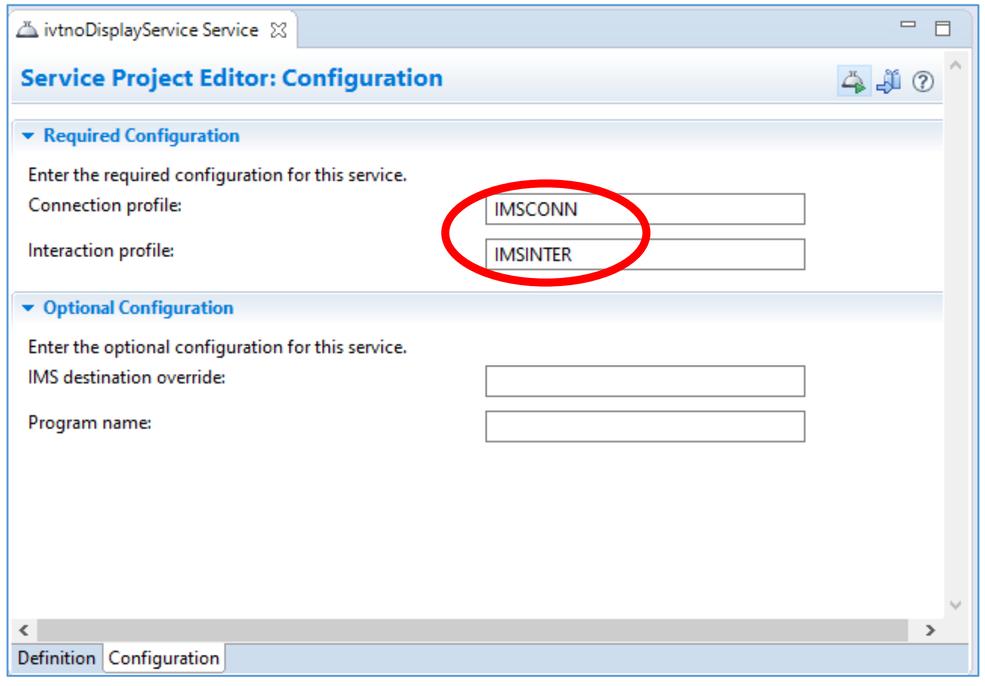
Creating a “GET” service interface request definition

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



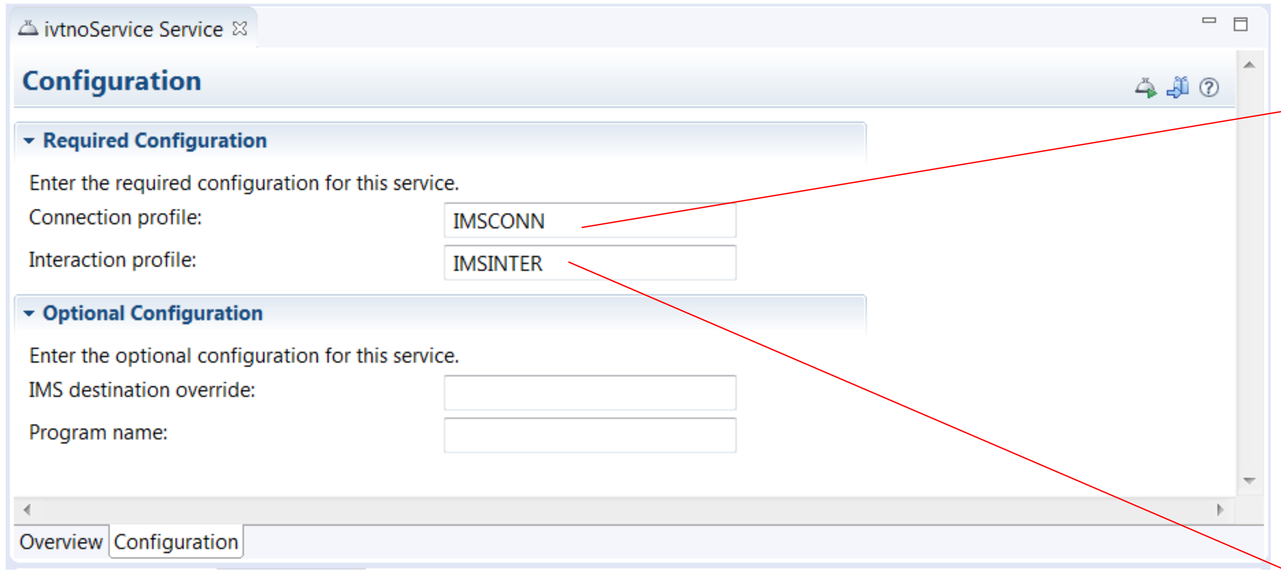
The service developer creates distinct services for each function.

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



mitchj@us.ibm.com

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>
```

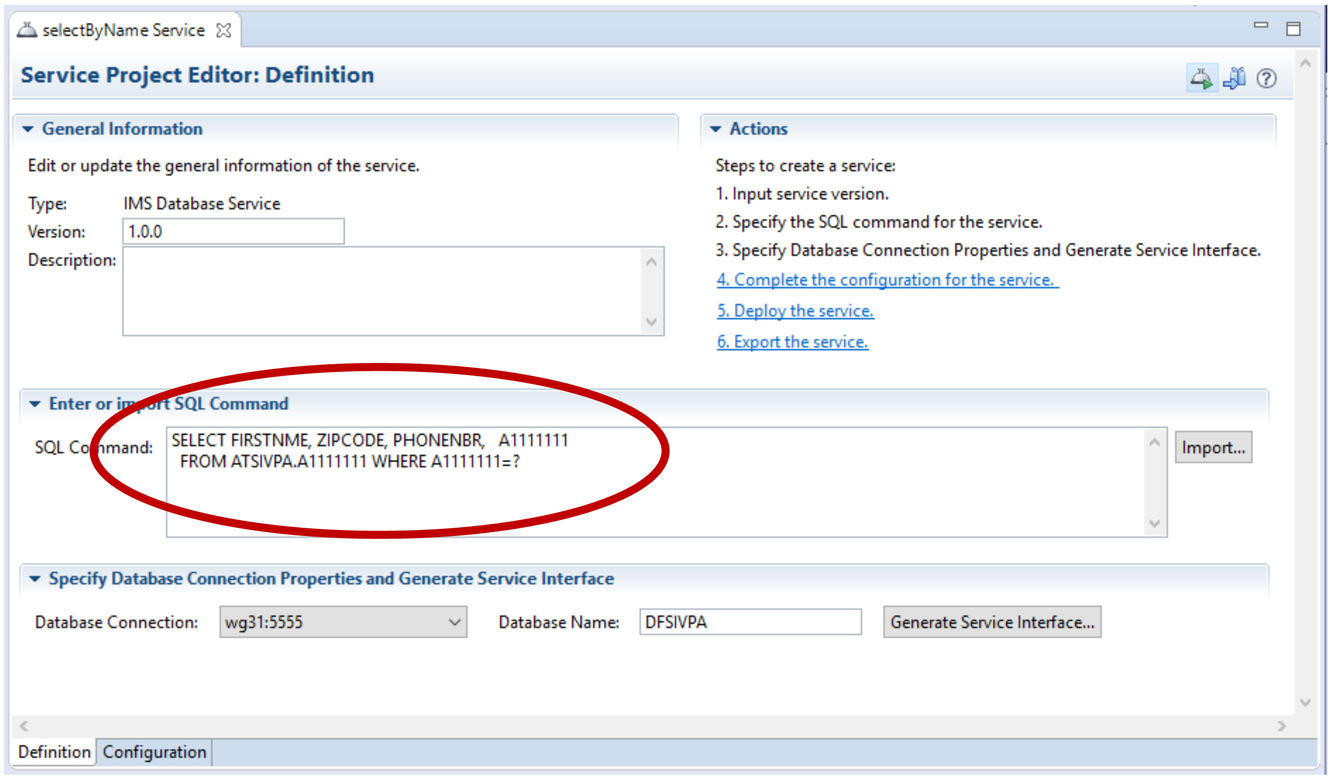
IMS Connect HWSCFG

```
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=OT
MAMEM)
```

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*

response

Service Interface Editor

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

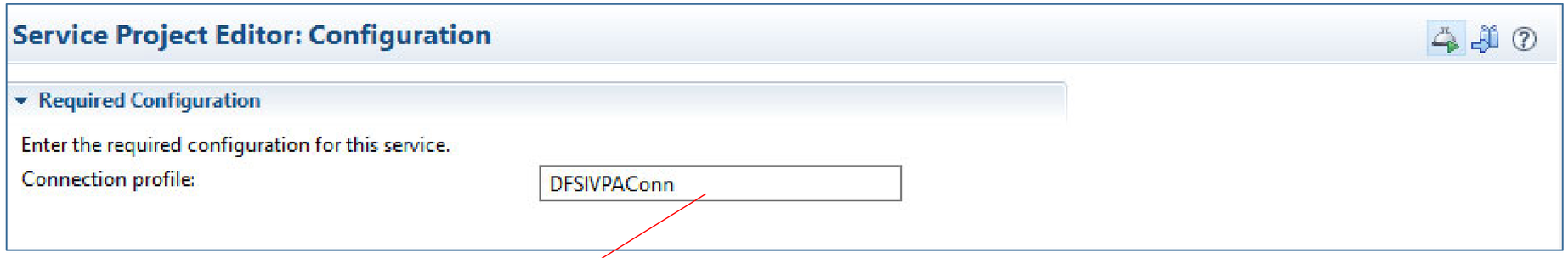
Search:

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
selectByName_Response					
Segment 1					
Output Columns		response			
Result [0..*]	☑	result		ARRAY	0
FIRSTNME	☑	firstName		CHAR	10
ZIPCODE	☑	zipCode		CHAR	7
PHONENBR	☑	phoneNuber		CHAR	10
A1111111	☑	lastName		CHAR	10

*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

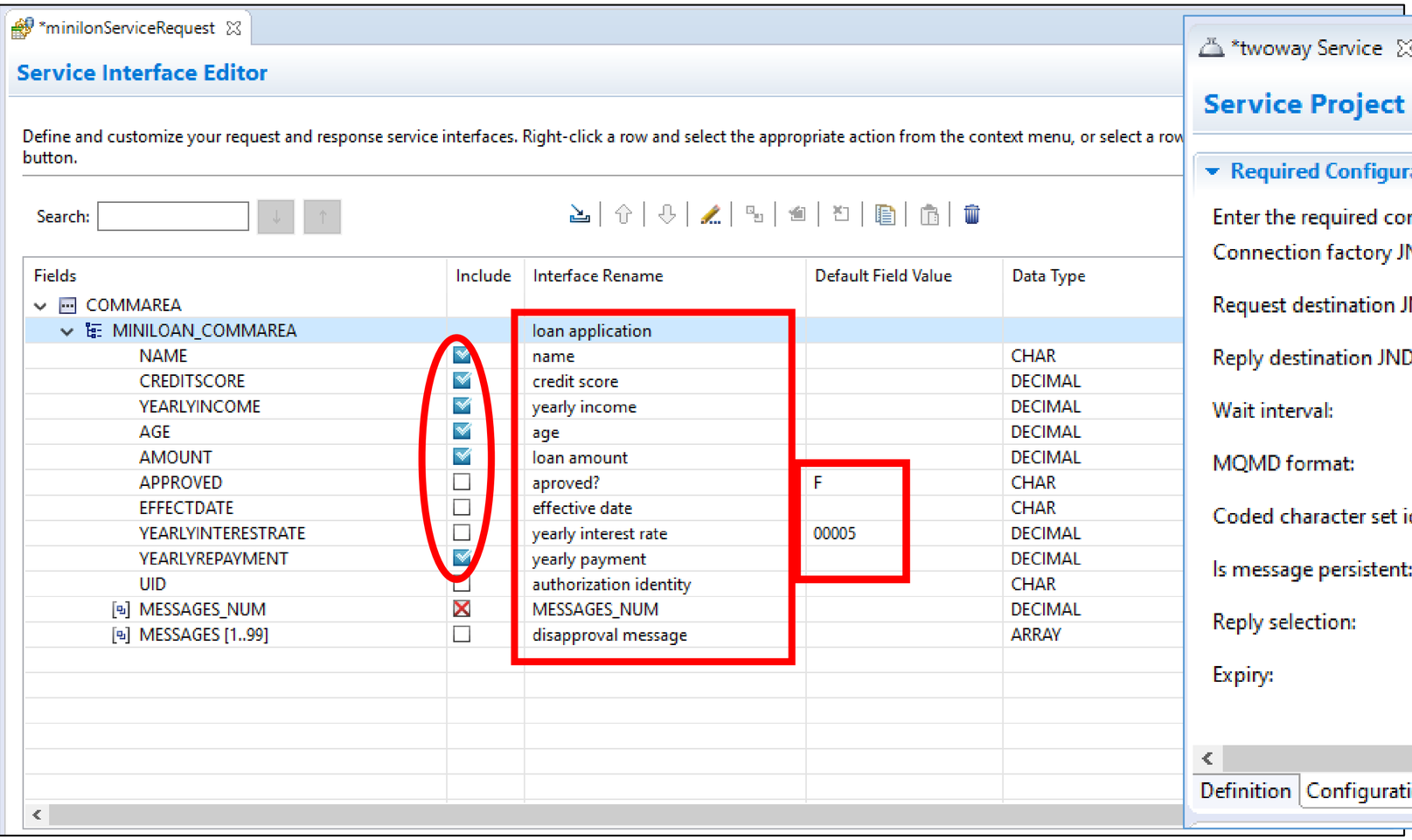
```
HWS=(ID=IMS14HWS,XIBAREA=100,RACF=N,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)
DATASTORE=(GROUP=OTMAGRP,ID=IVP1,MEMBER=HWSMEM,TMEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS,TMEMBER=PLEX1)
ODACCESS=(ODBMAUTOCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000),ODBMTMOT=6000)
```

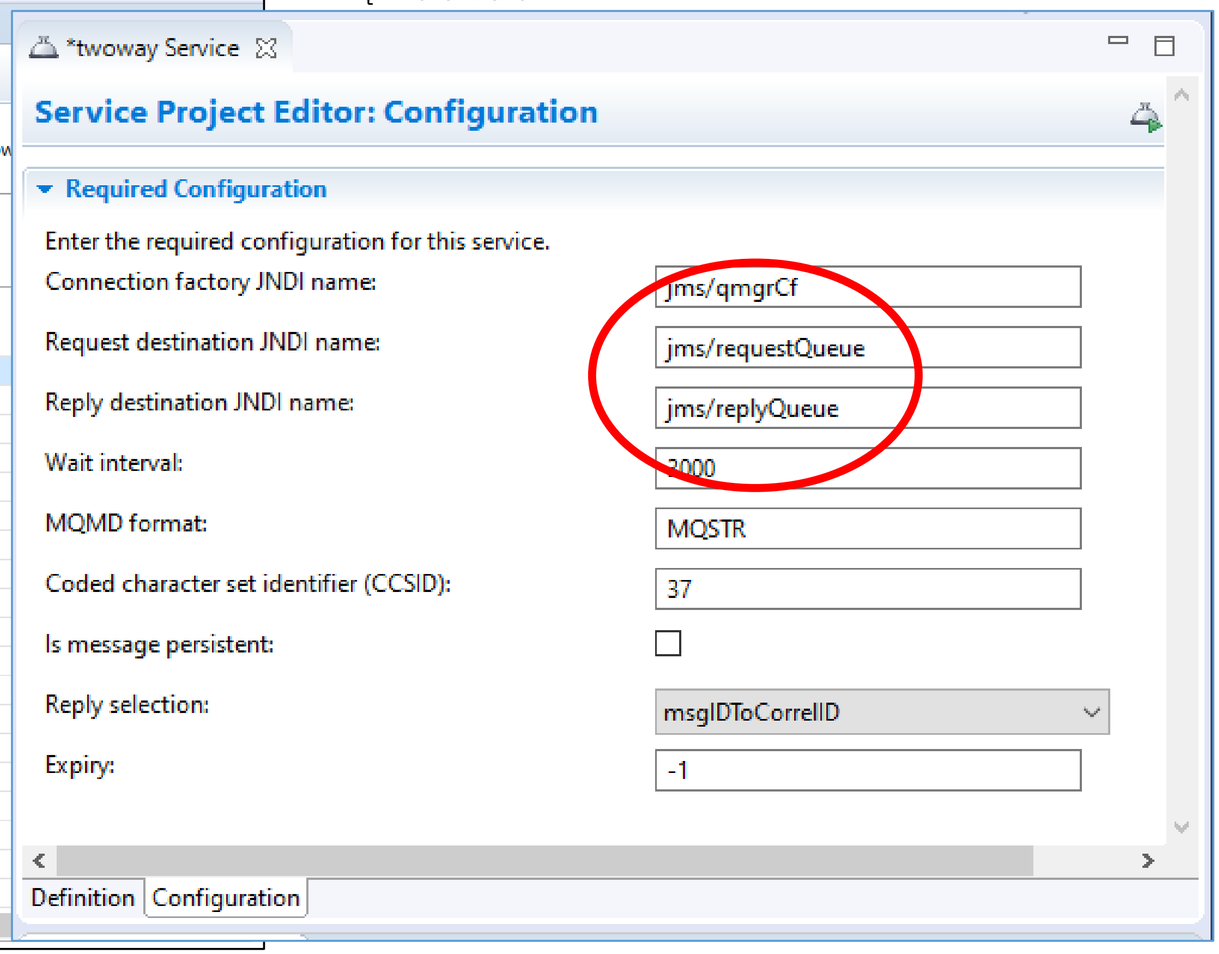
mitchj@us.ibm.com

API toolkit – Creating Services for MQ

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

MQ Meta Data

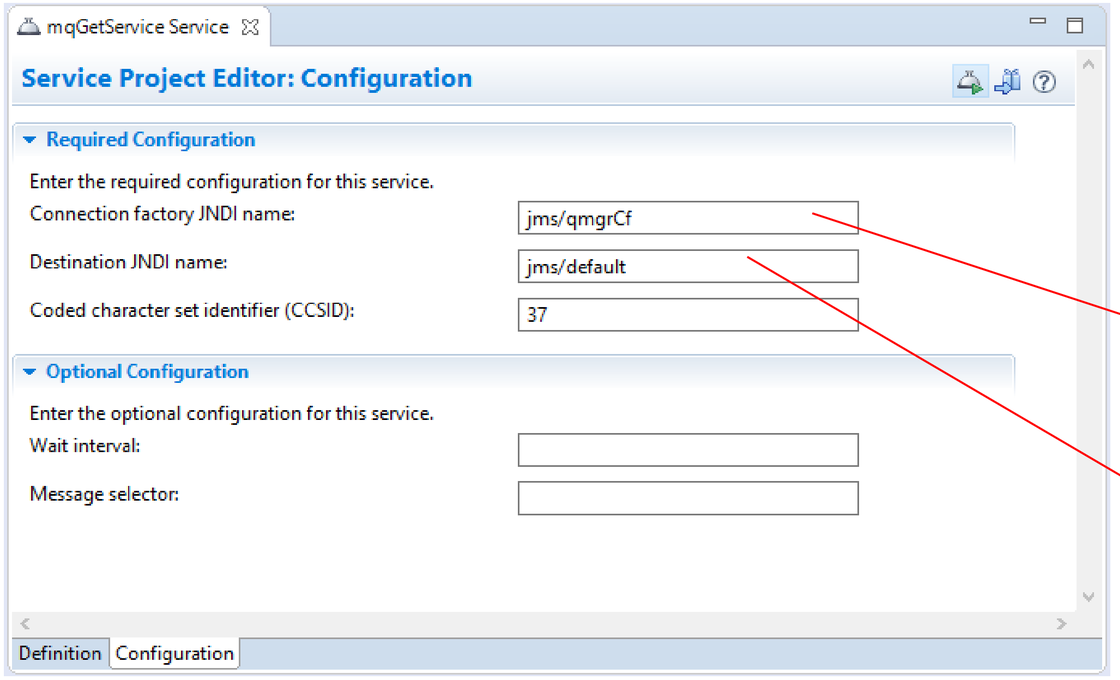


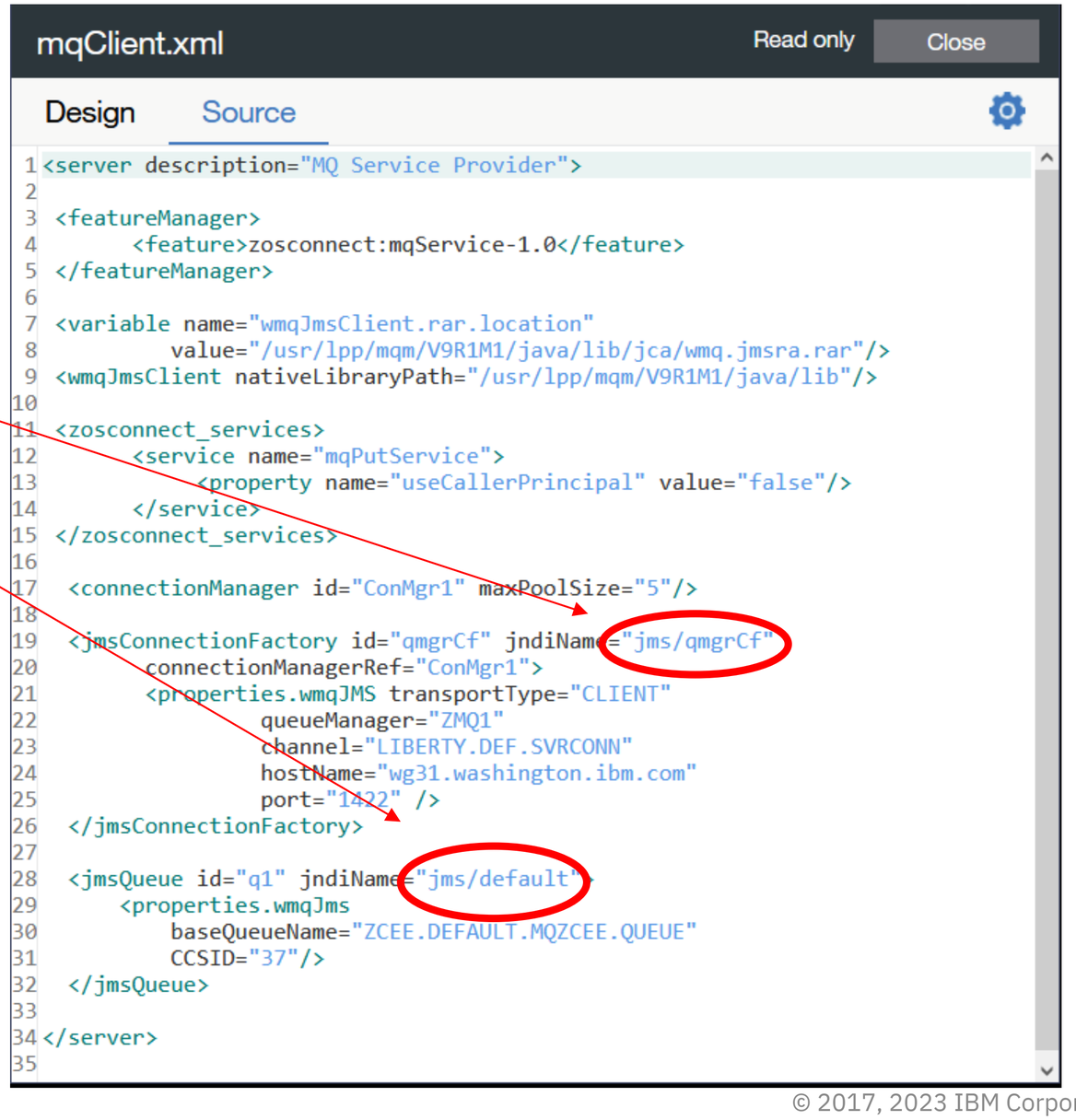


Again the service developer can then see the imported data structure and can **redact fields**, **rename fields**, and **add default values to fields** to make the service more consumable for an API developer.

mitchj@us.ibm.com

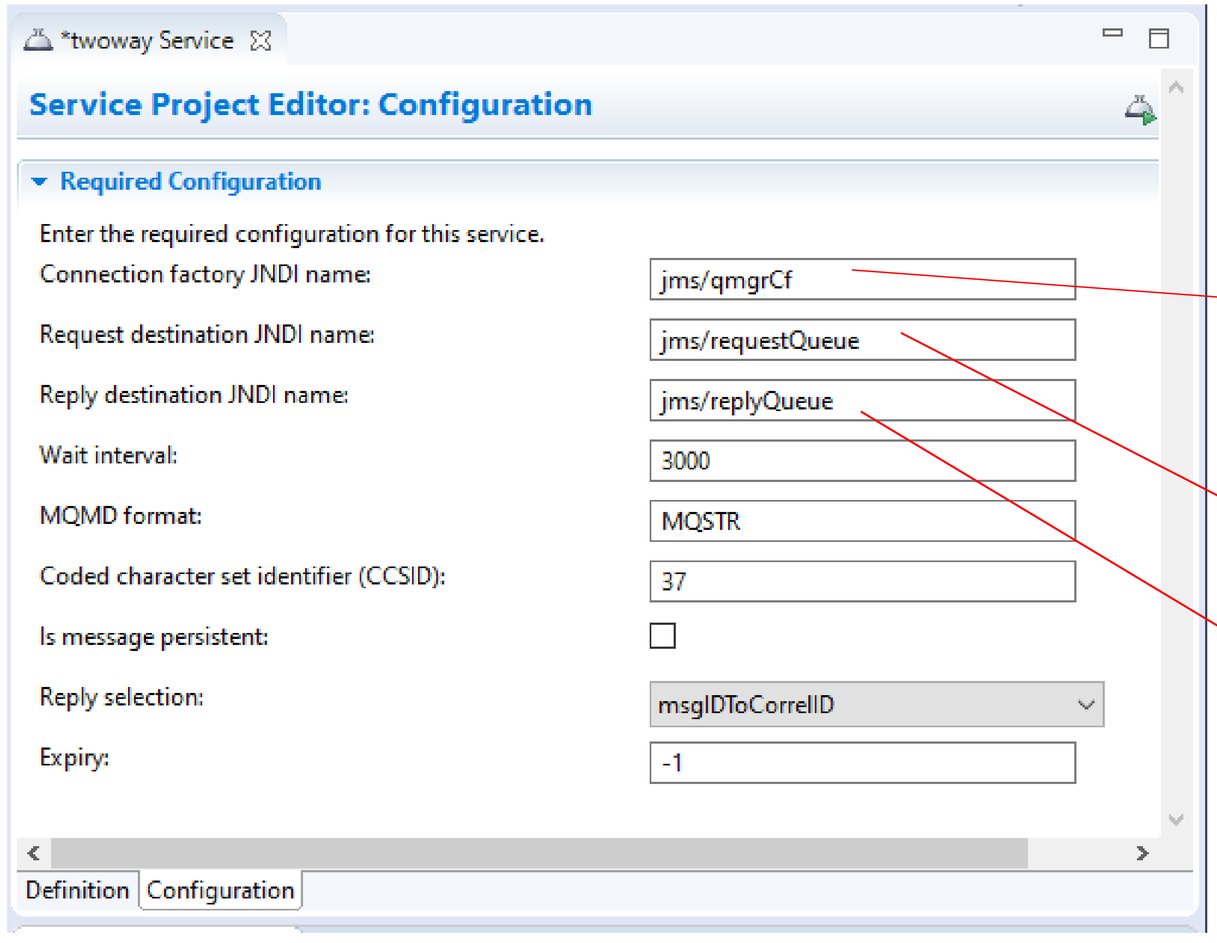
Using JMS to access MQ (One-Way)

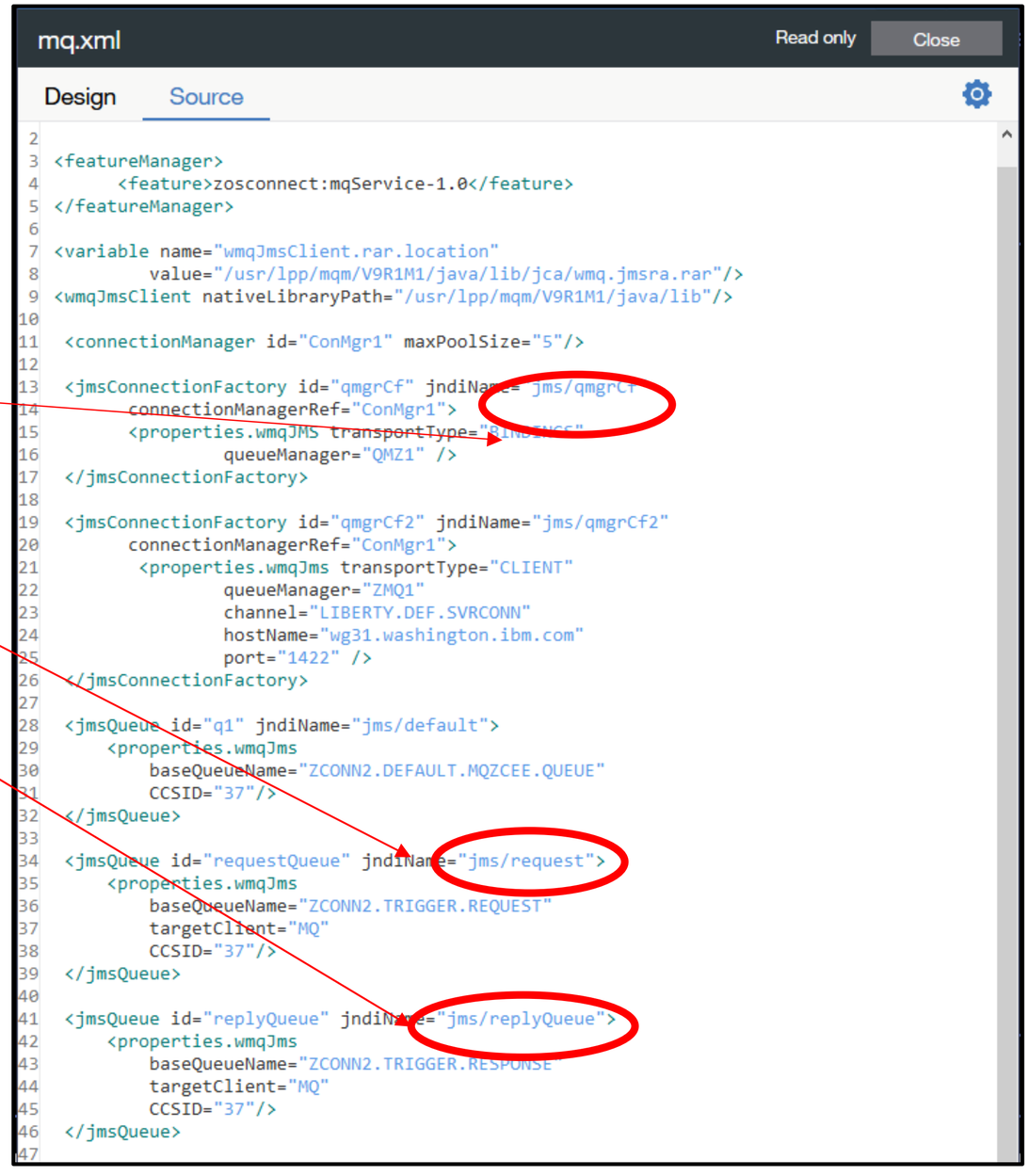




mitchj@us.ibm.com

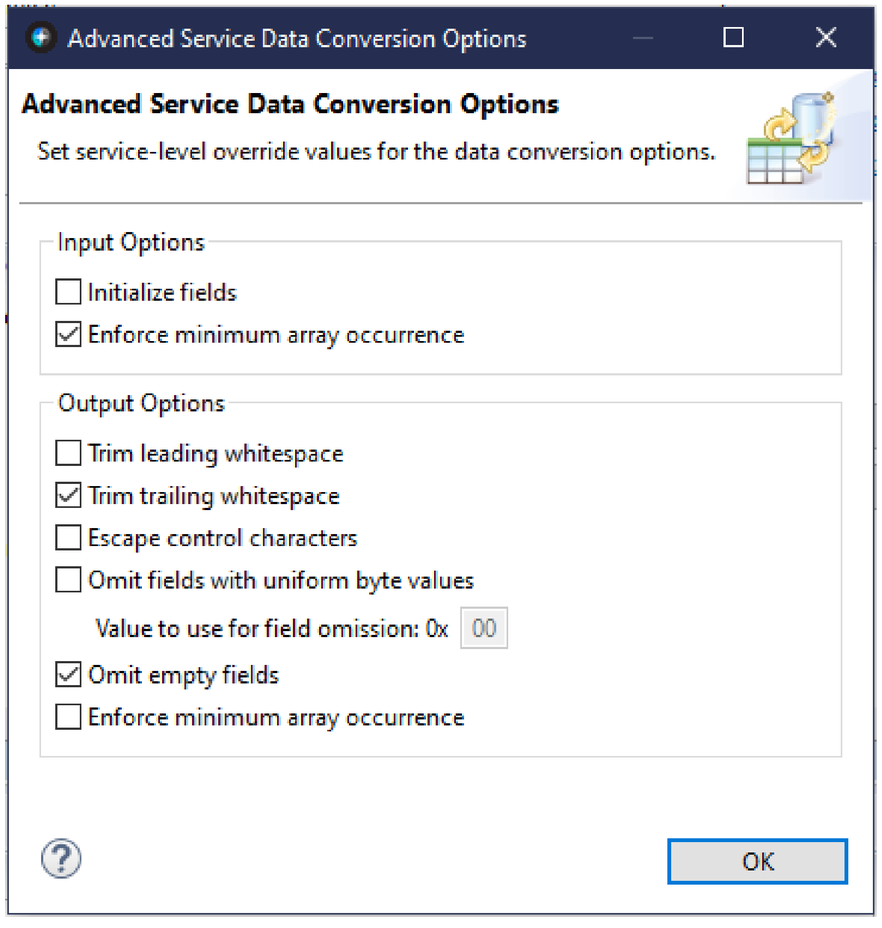
Using JMS to access MQ (Two-Way)





mitchj@us.ibm.com

API toolkit – Advanced Data Conversion Options



Request Messages:

- Initialize fields
- Enforce minimum array occurrence

Response Messages:

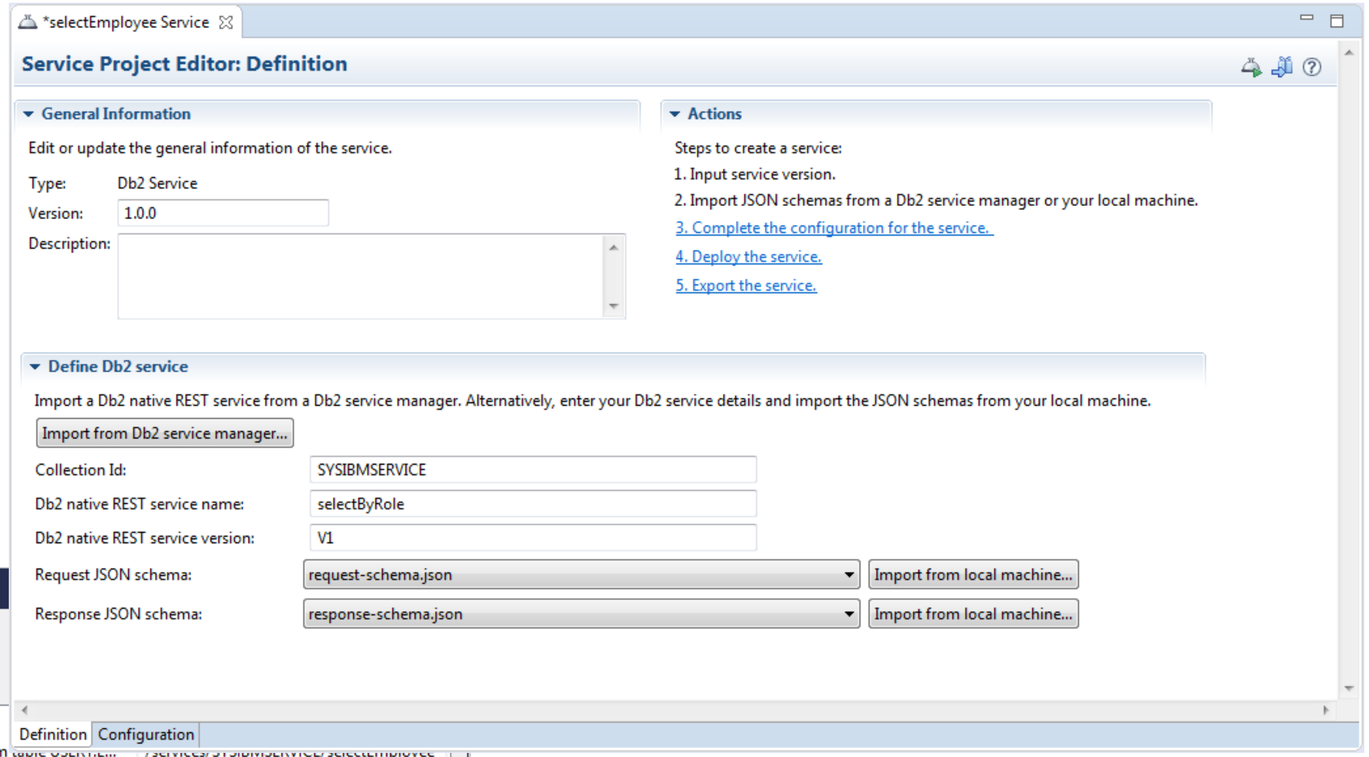
- Trim leading whitespace
- Trim trailing whitespace
- Escape control characters
- Omit fields with uniform byte values
- Omit empty fields
- Enforce minimum array occurrence

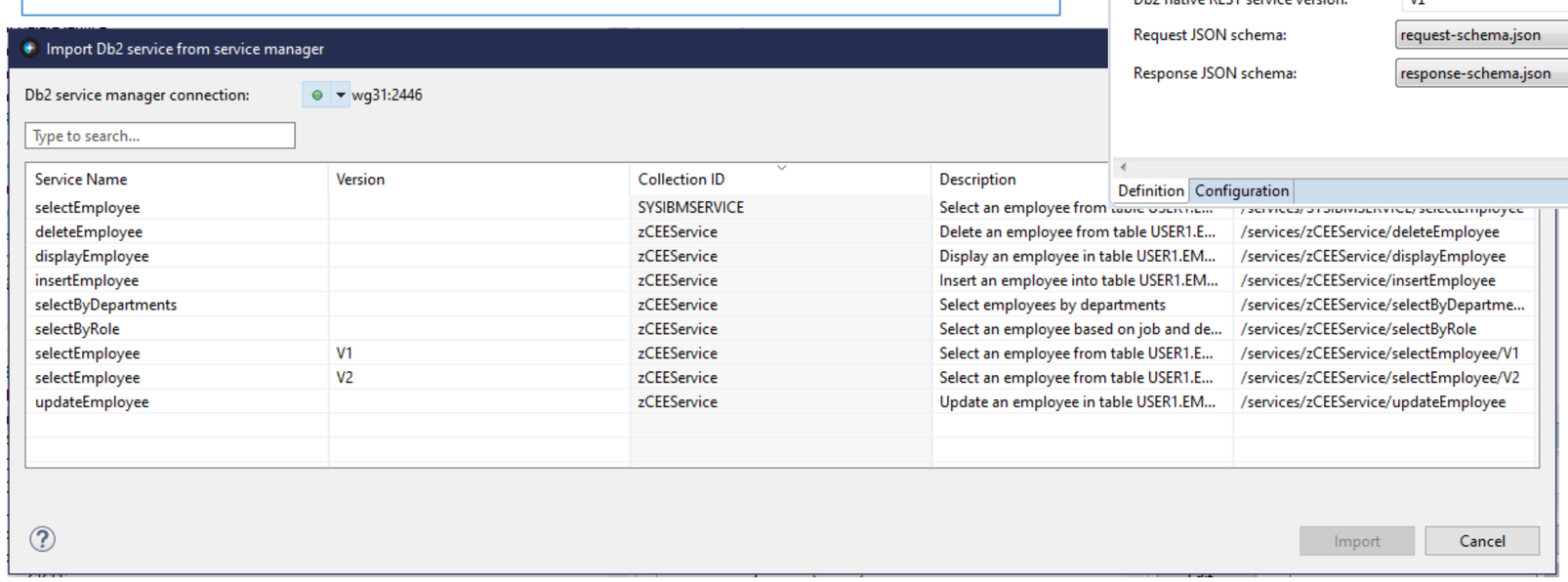
mitchj@us.ibm.com

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')
```



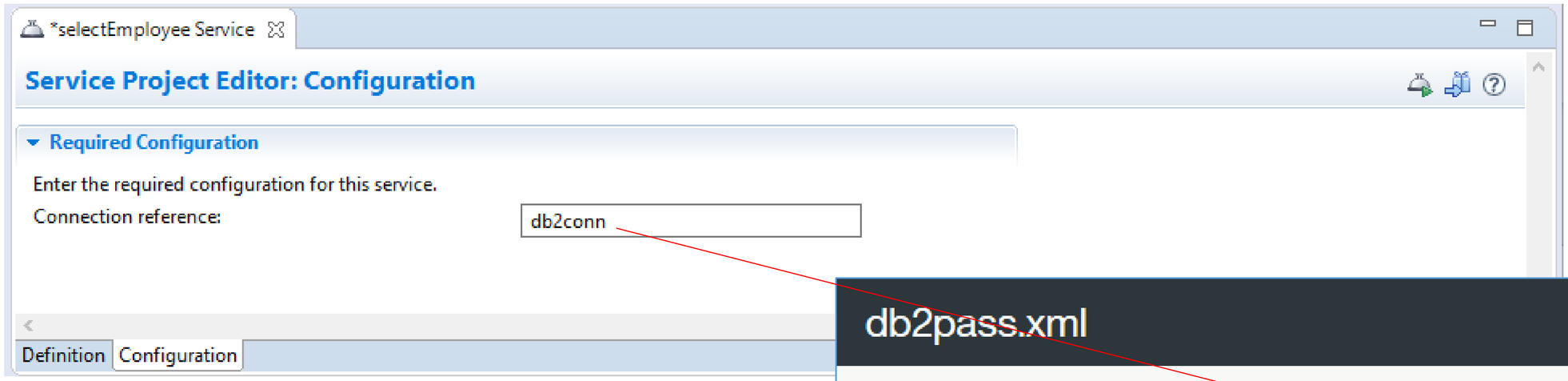


The service developer retrieves details about the Db2 REST services

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

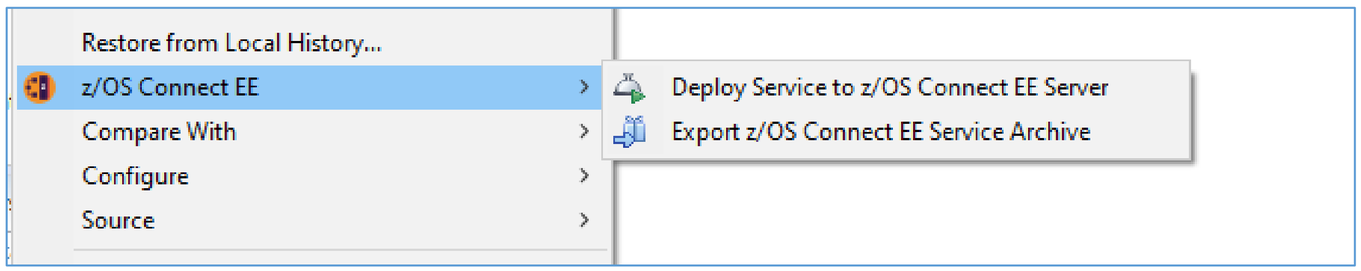
```
1  <server description="DB2 REST">
2
3   <zosconnect_zosConnectServiceRestClientConnection id="db2conn"
4         host="wg31.washington.ibm.com"
5         port="2446"
6         basicAuthRef="dsn2Auth" />
7
8   <zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
9         applName="DSN2APPL"/>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

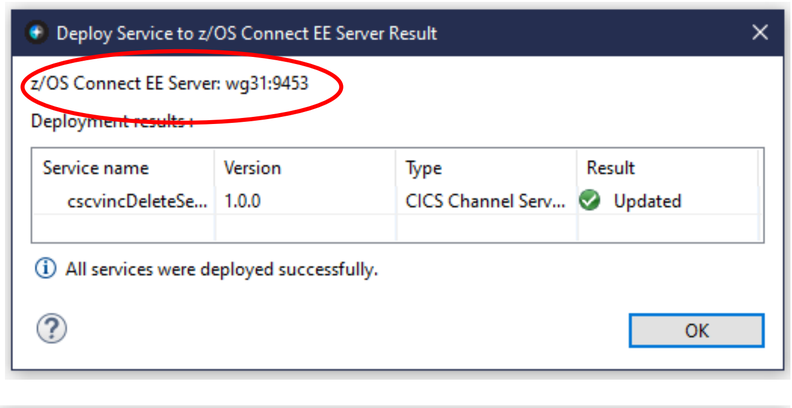
mitchj@us.ibm.com

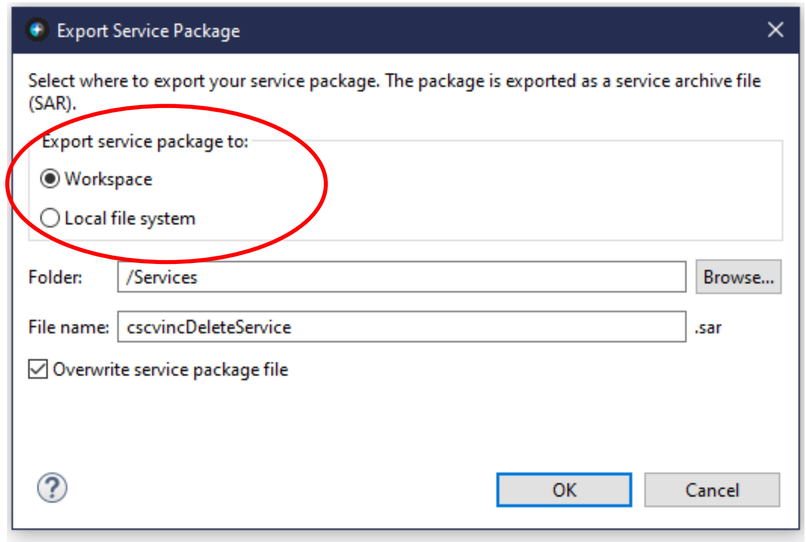
API toolkit – Services Editor

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?

Creating APIs using the Eclipse Tooling

Remember: All service archives files are functionally equivalent regardless of how there are created or which resources they are intended to access

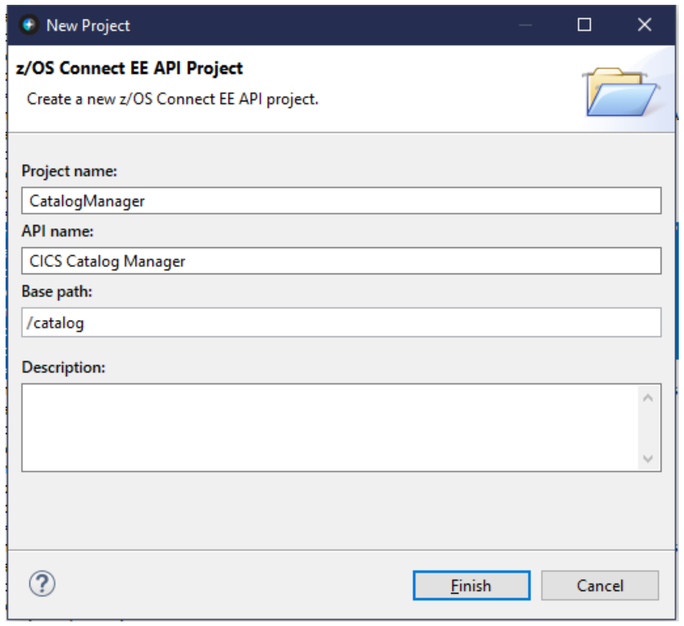
mitchj@us.ibm.com

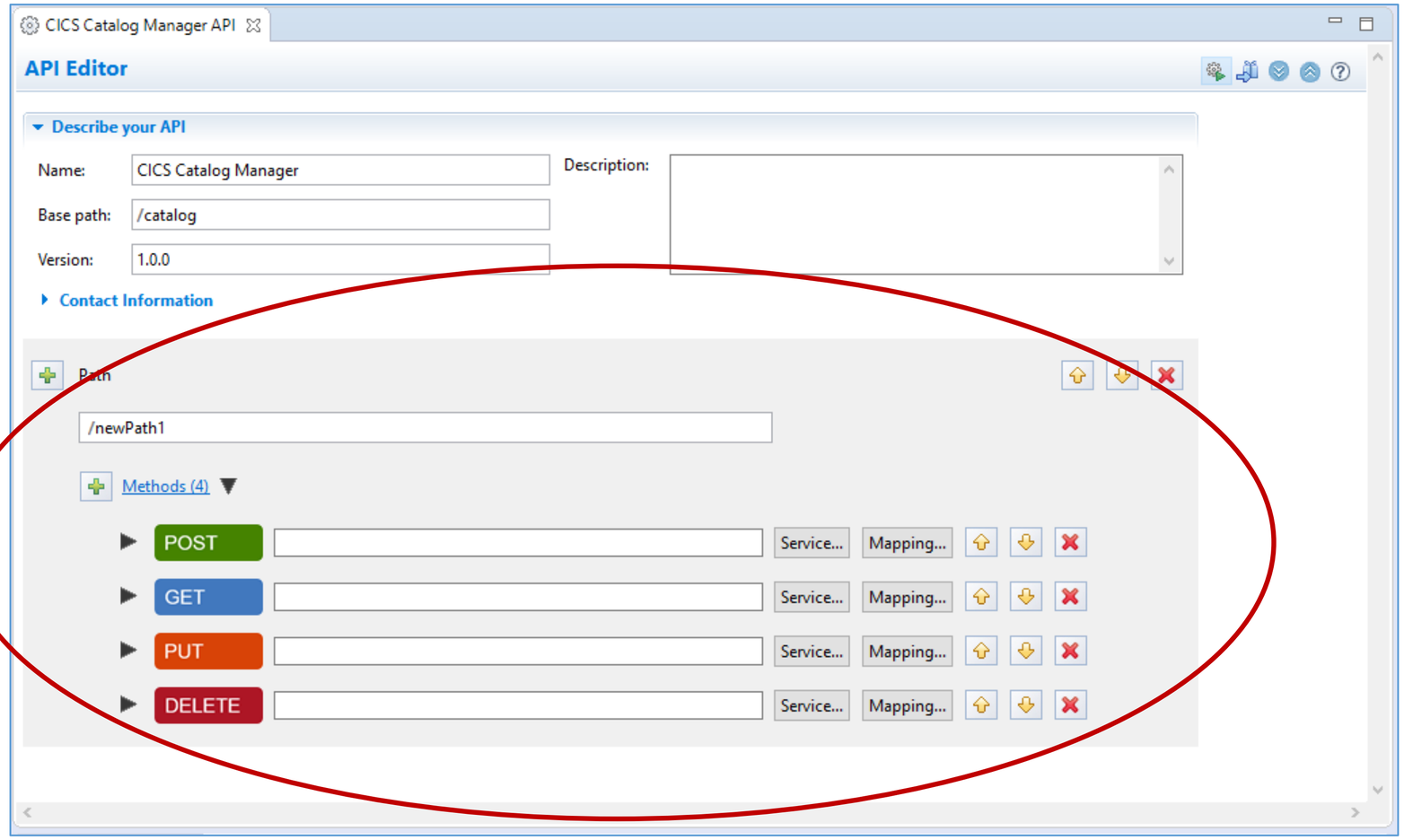
/api_toolkit/api_editor

Quick and easy **API mapping.**

mitchj@us.ibm.com

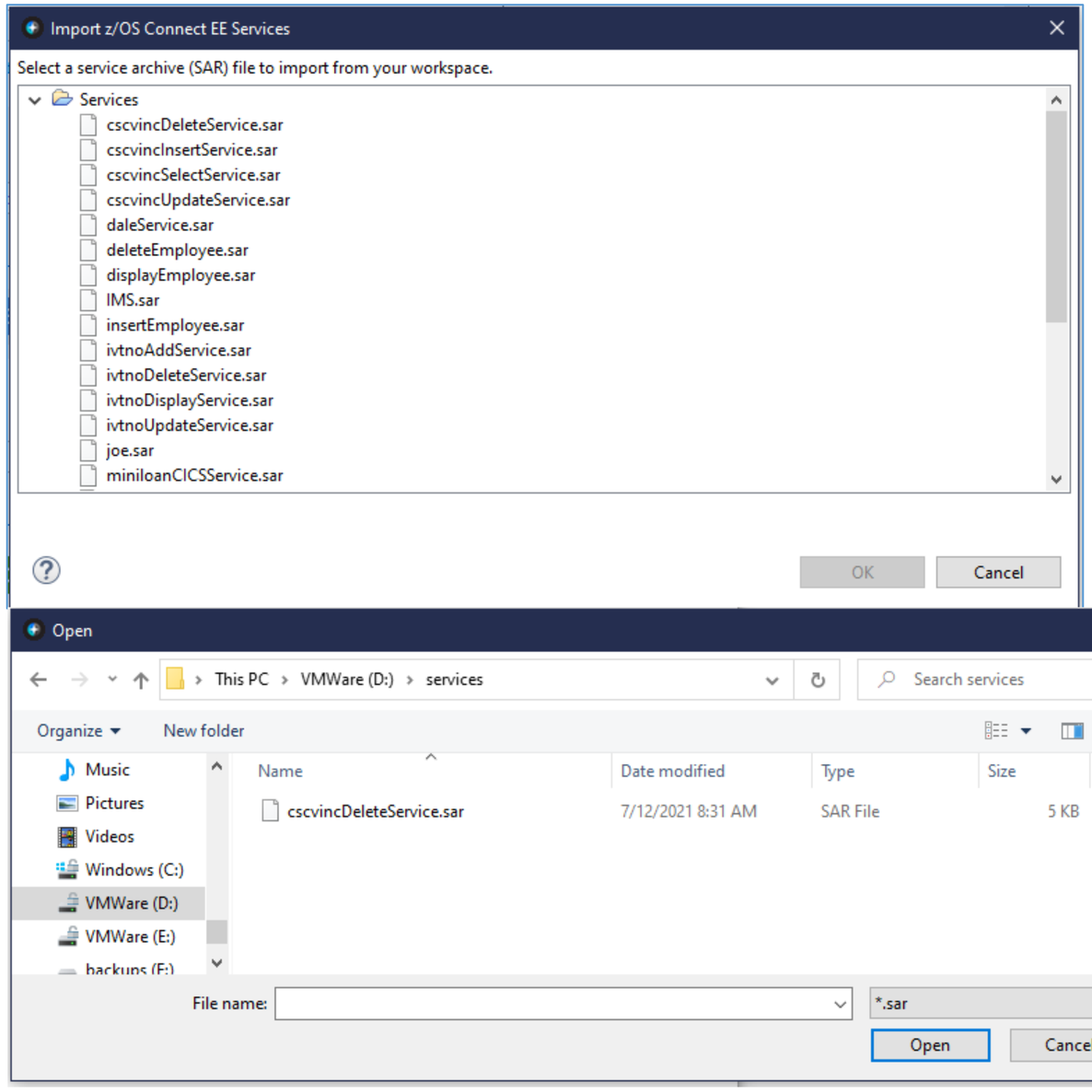
API toolkit – API Editor

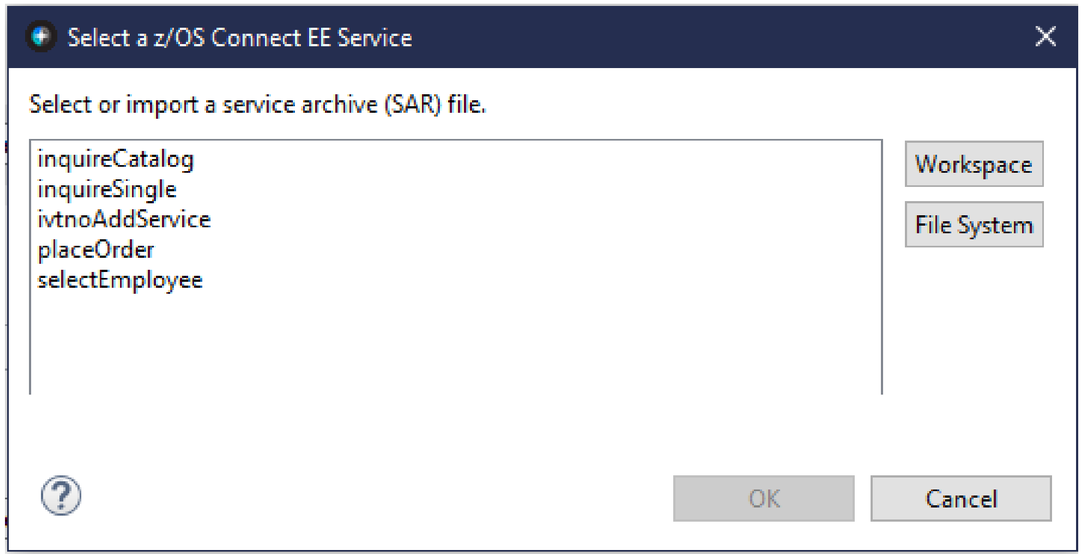




mitchj@us.ibm.com

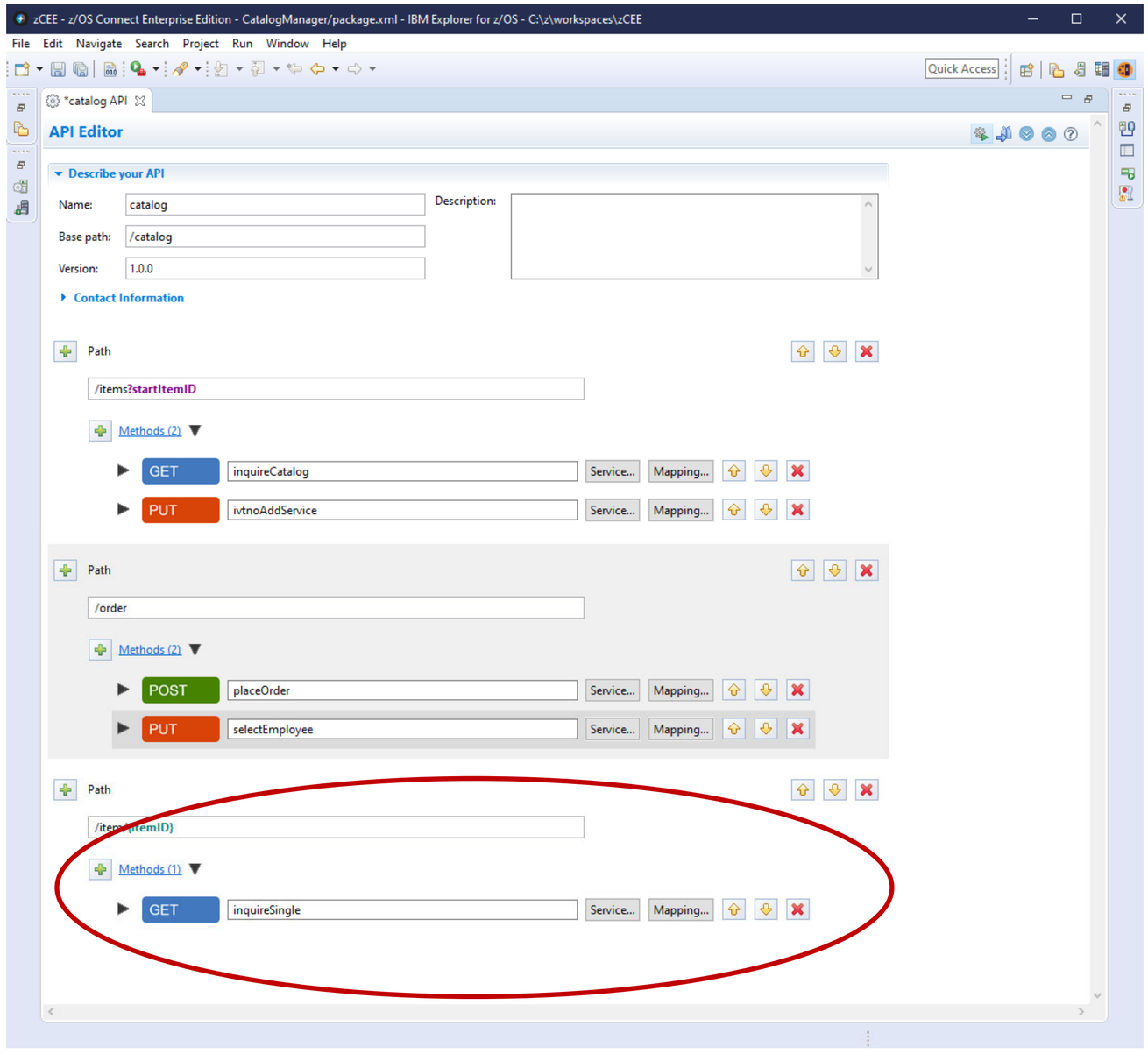
Importing the service archive file from a filesystem or Eclipse project directory





mitchj@us.ibm.com

API toolkit – API Editor



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

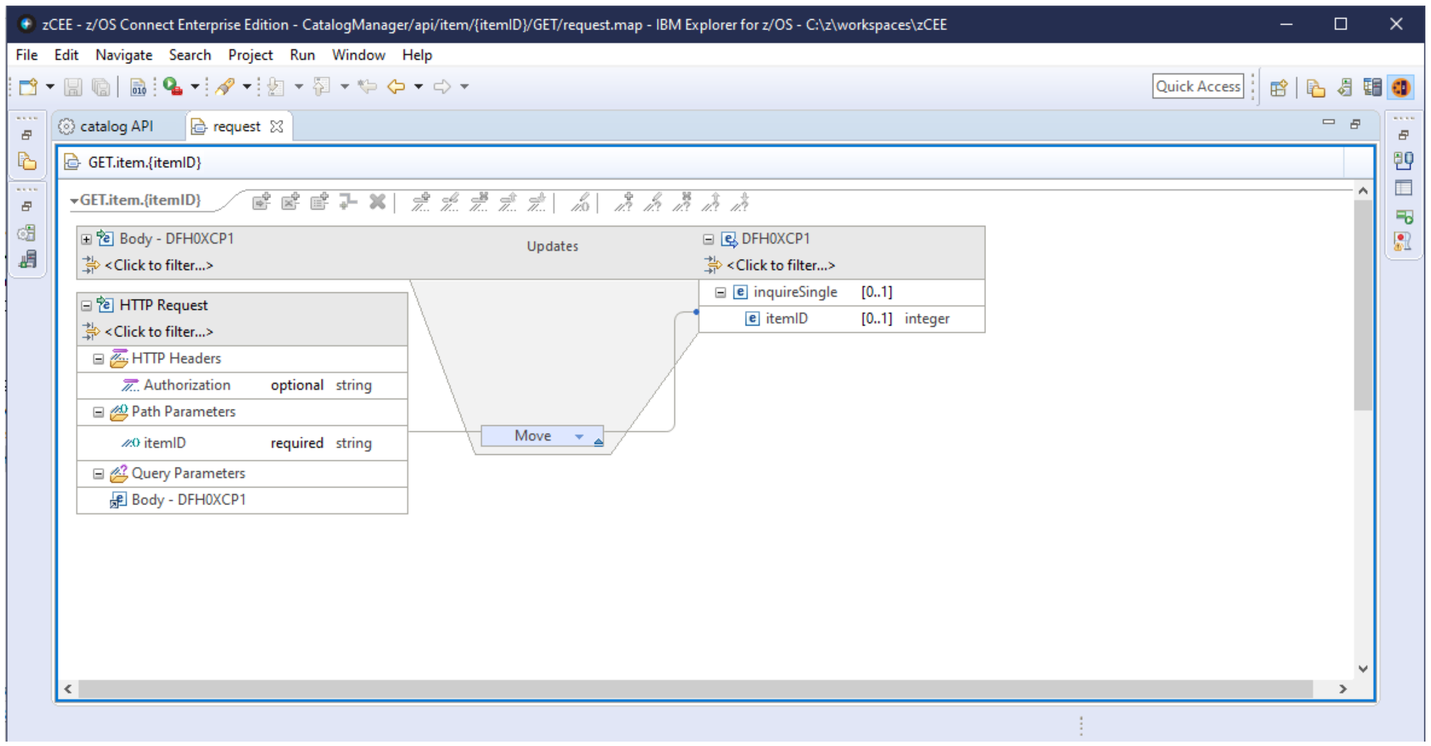
Once you define your API, you can map backend services to each request.

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.

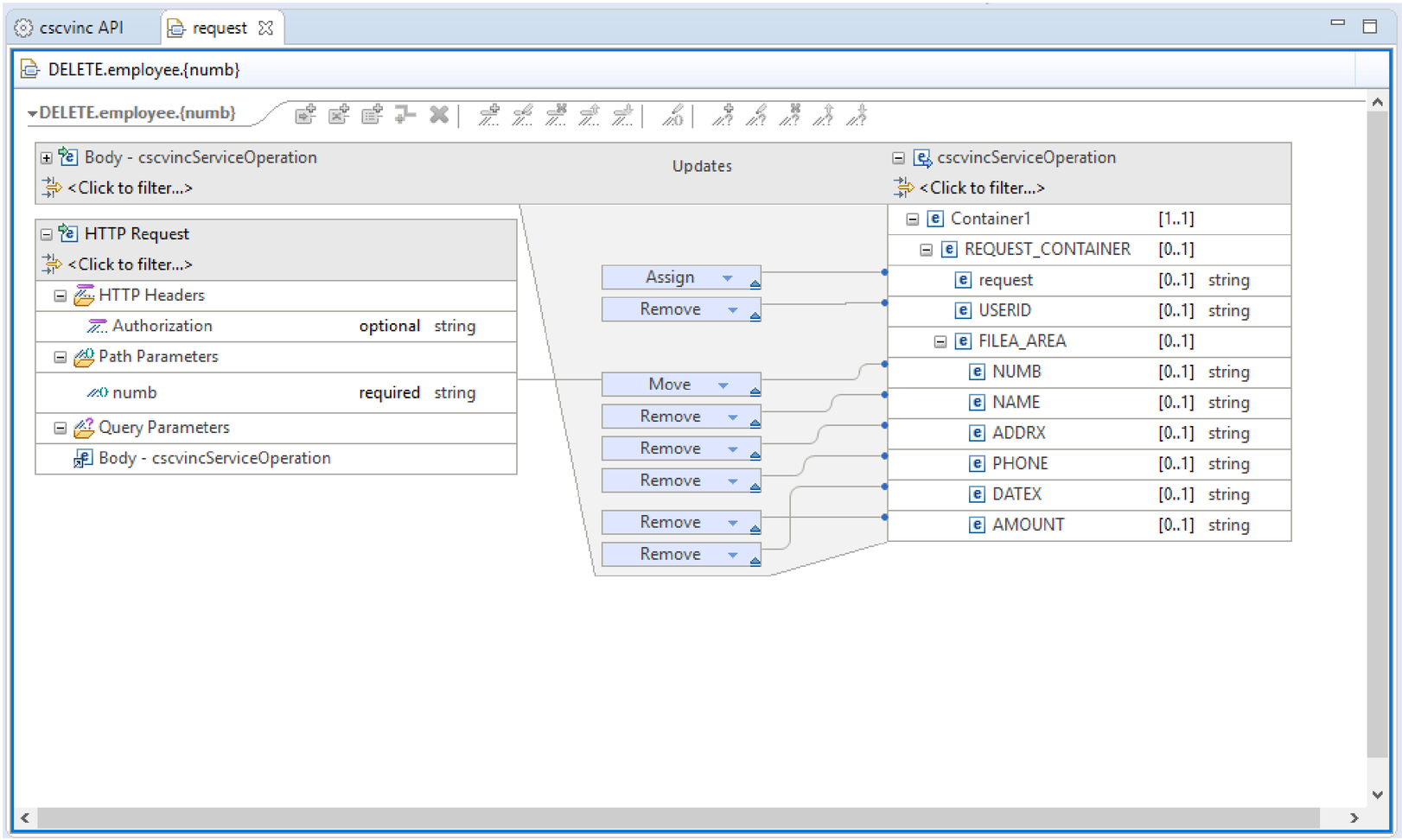
Assign static values to fields, useful for Op codes.

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

mitchj@us.ibm.com

API toolkit – API Editor

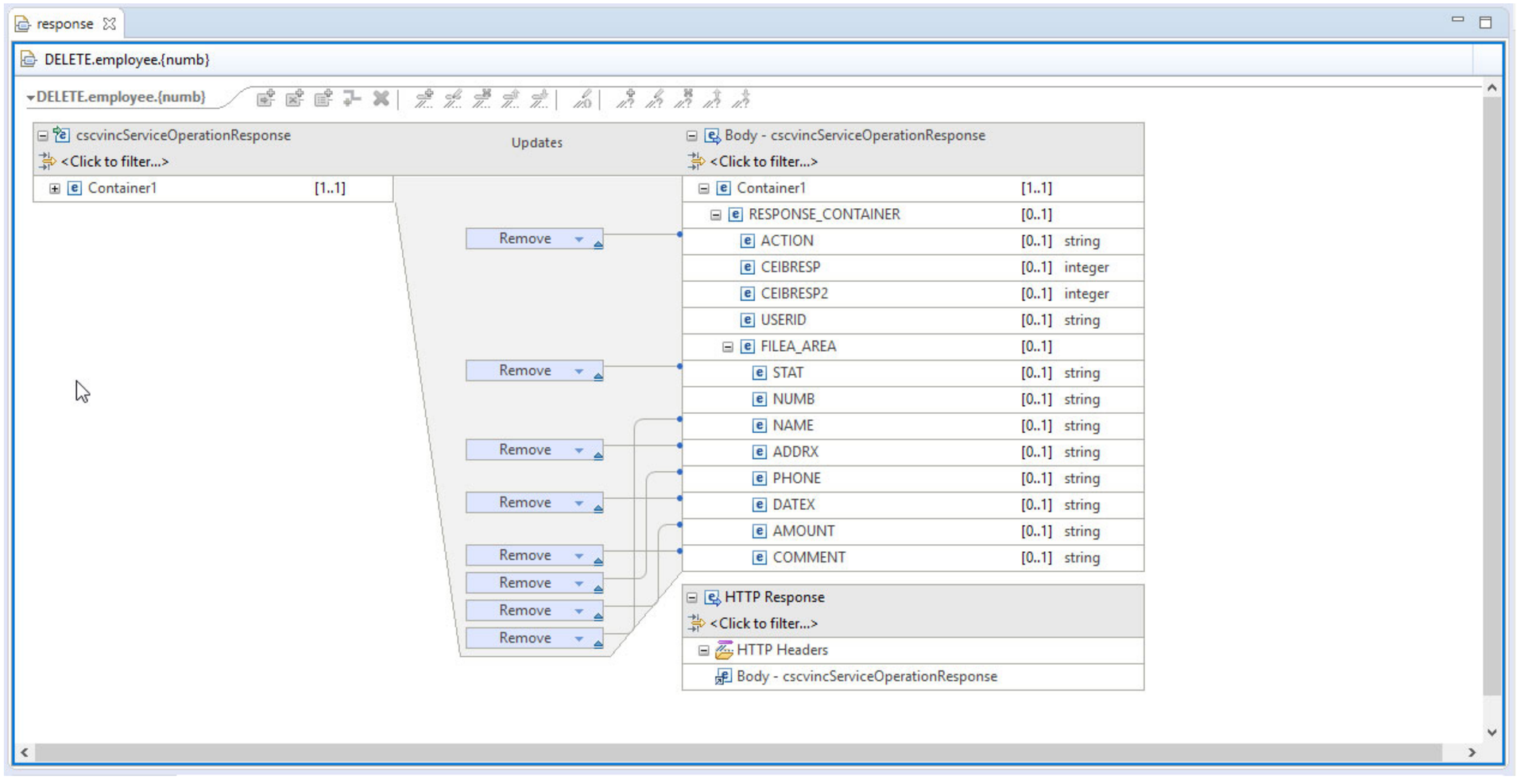
API mapping: Remove or assign values to the fields exposed by service developer



mitchj@us.ibm.com

API toolkit – API Editor

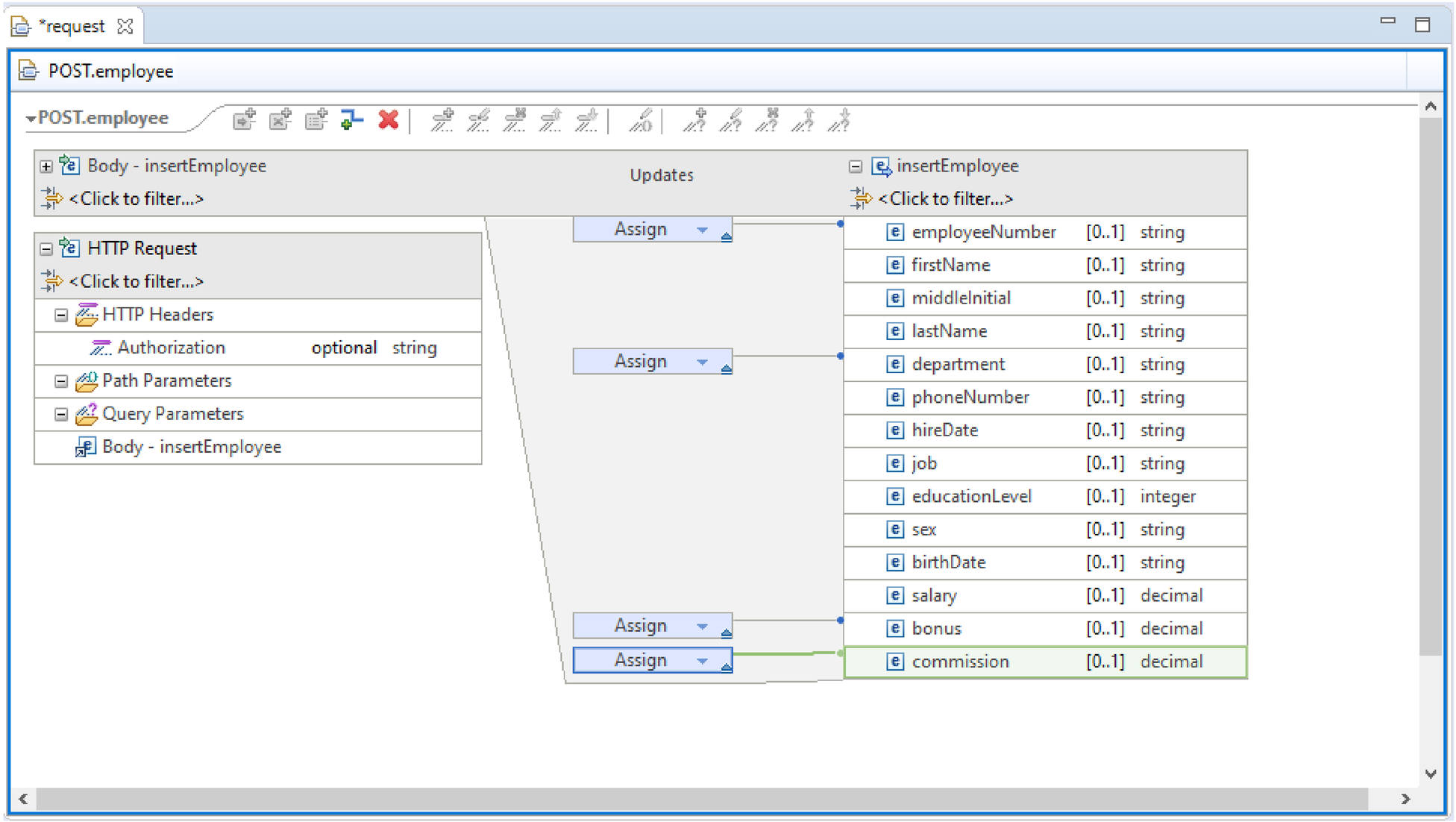
API mapping: Allows the API Developer to remove fields from the response to tailor the API



mitchj@us.ibm.com

API toolkit – API Editor and Db2 REST service

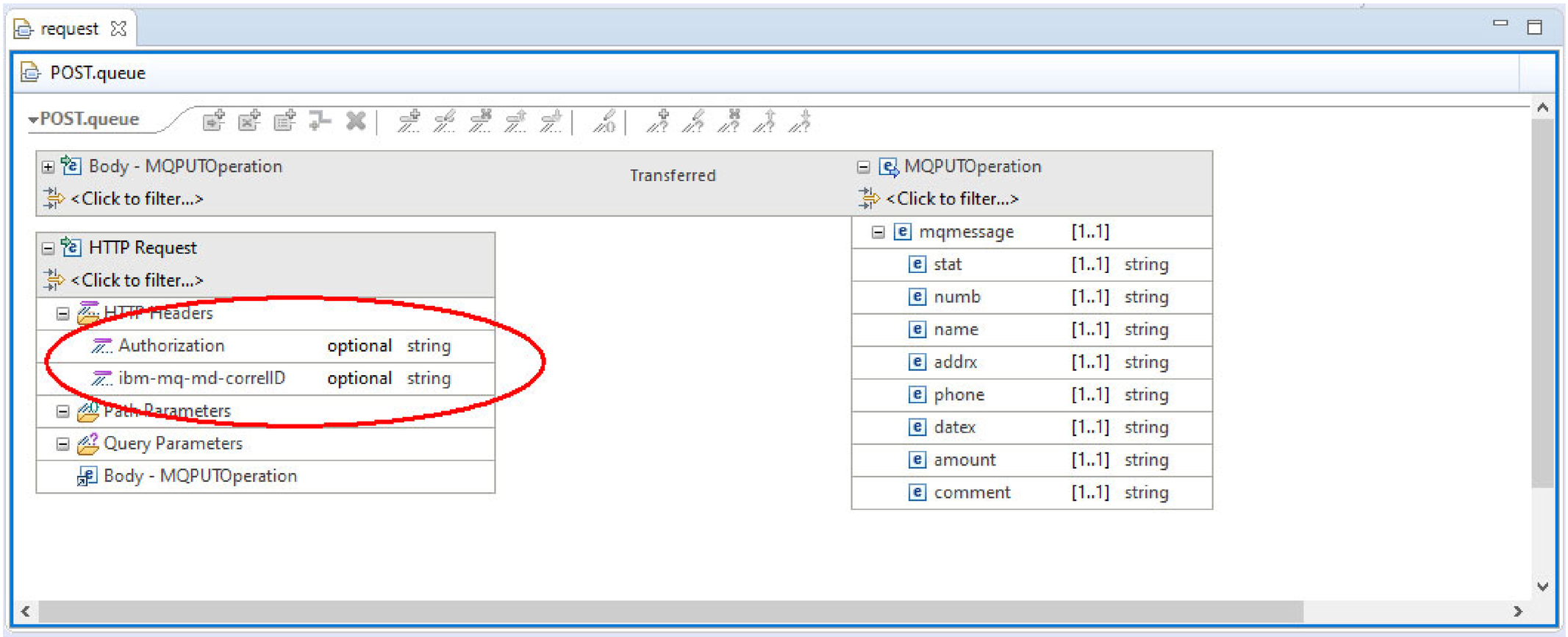
API mapping: Remove/Assign values columns exposed in Db2 REST service



mitchj@us.ibm.com

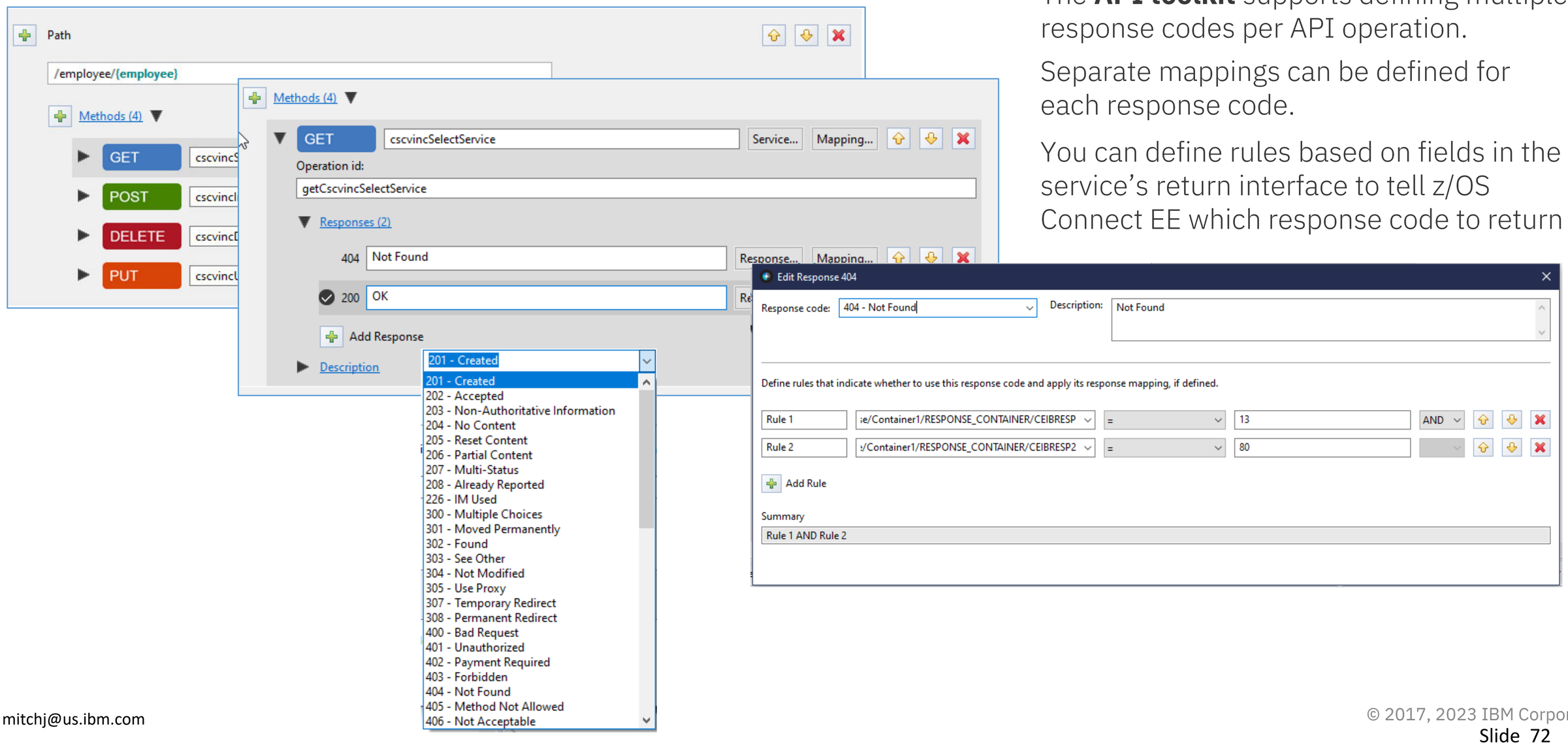
API toolkit – Header properties

API mapping: Allows adding HTTP header properties



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.

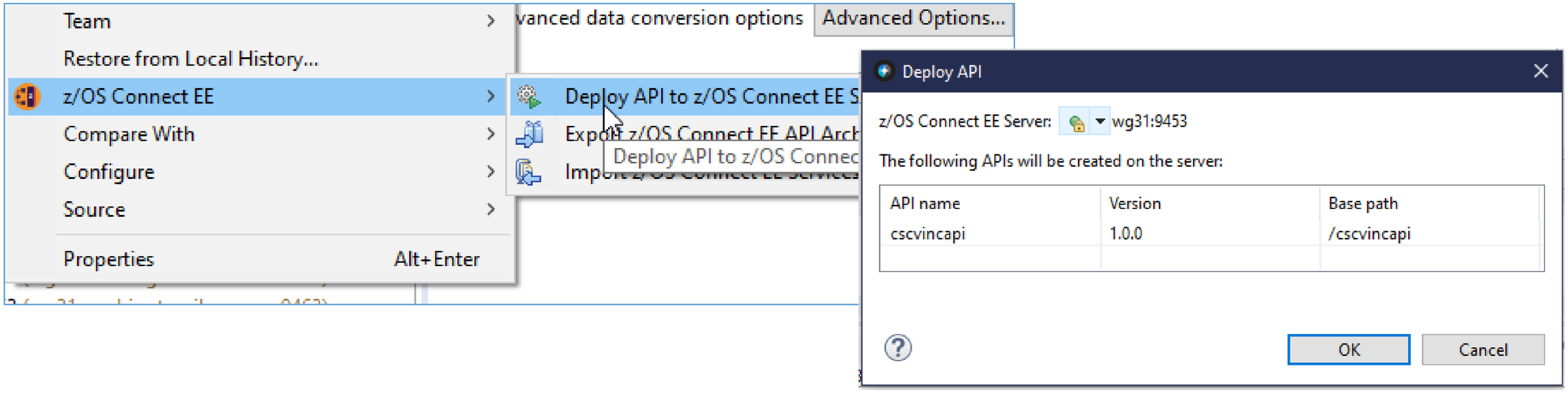
You can define rules based on fields in the service's return interface to tell z/OS Connect EE which response code to return

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.

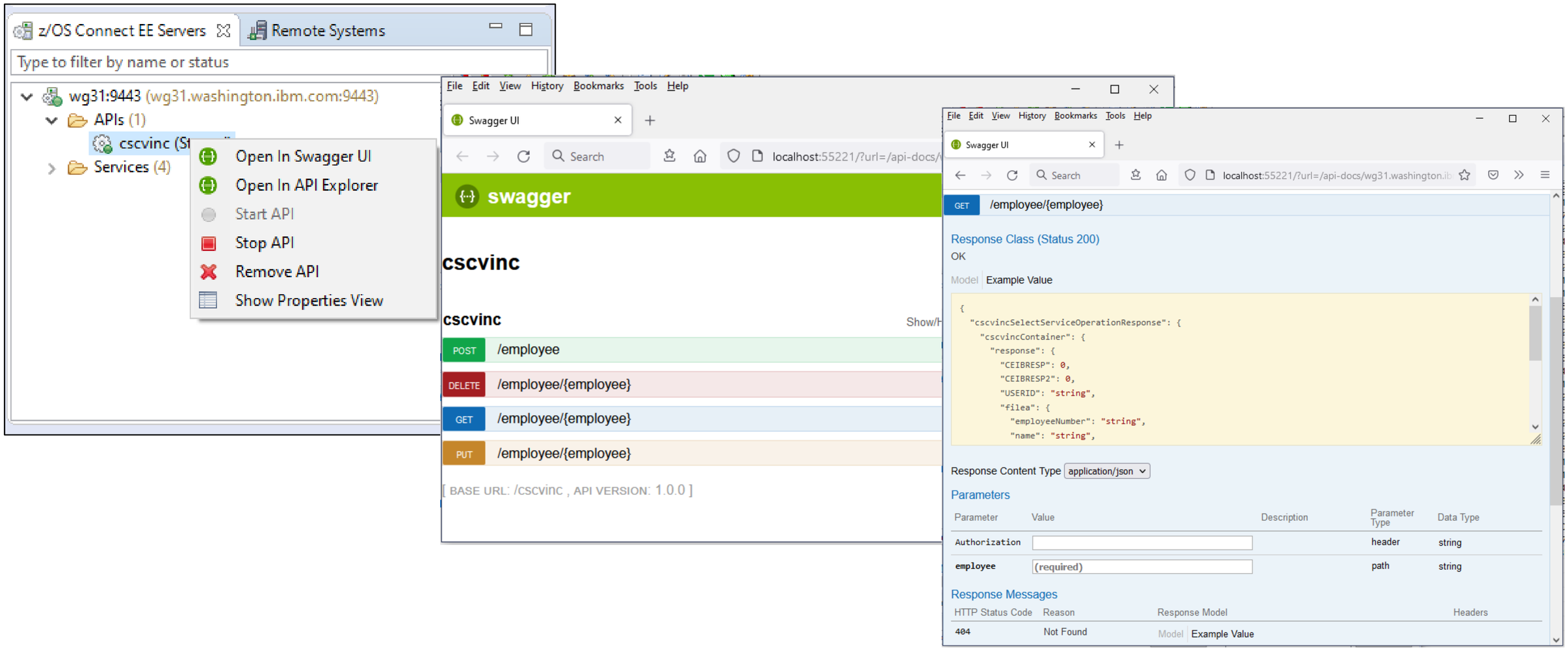
z/OS Connect EE servers view allows you to start, stop, and remove APIs from a running server.

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

/api_toolkit/services

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

mitchj@us.ibm.com

Creating Service and API Archive files outside of the Eclipse tooling

- **Use zconbt to create a service archive file**

```
C:\z\workspaces\zCEE>..\..\software\zconbt\bin\zconbt --
projectDirectory=C:\z\workspaces\zcee\miniloanService --file=miniloan.sar
BAQB0000I: z/OS Connect Enterprise Edition 3.0 Build Toolkit Version 1.10 (20220817-1609).
BAQB0001I: Creating service archive from configuration file
C:\z\workspaces\zcee\miniloanService\service.properties.
BAQB1022I: IBM MQ for z/OS plugin for IBM z/OS Connect EE V3.0 build toolkit, code level is
3.0.60.0(20220817-1609).
BAQB0002I: Successfully created service archive file miniloan.sar
```

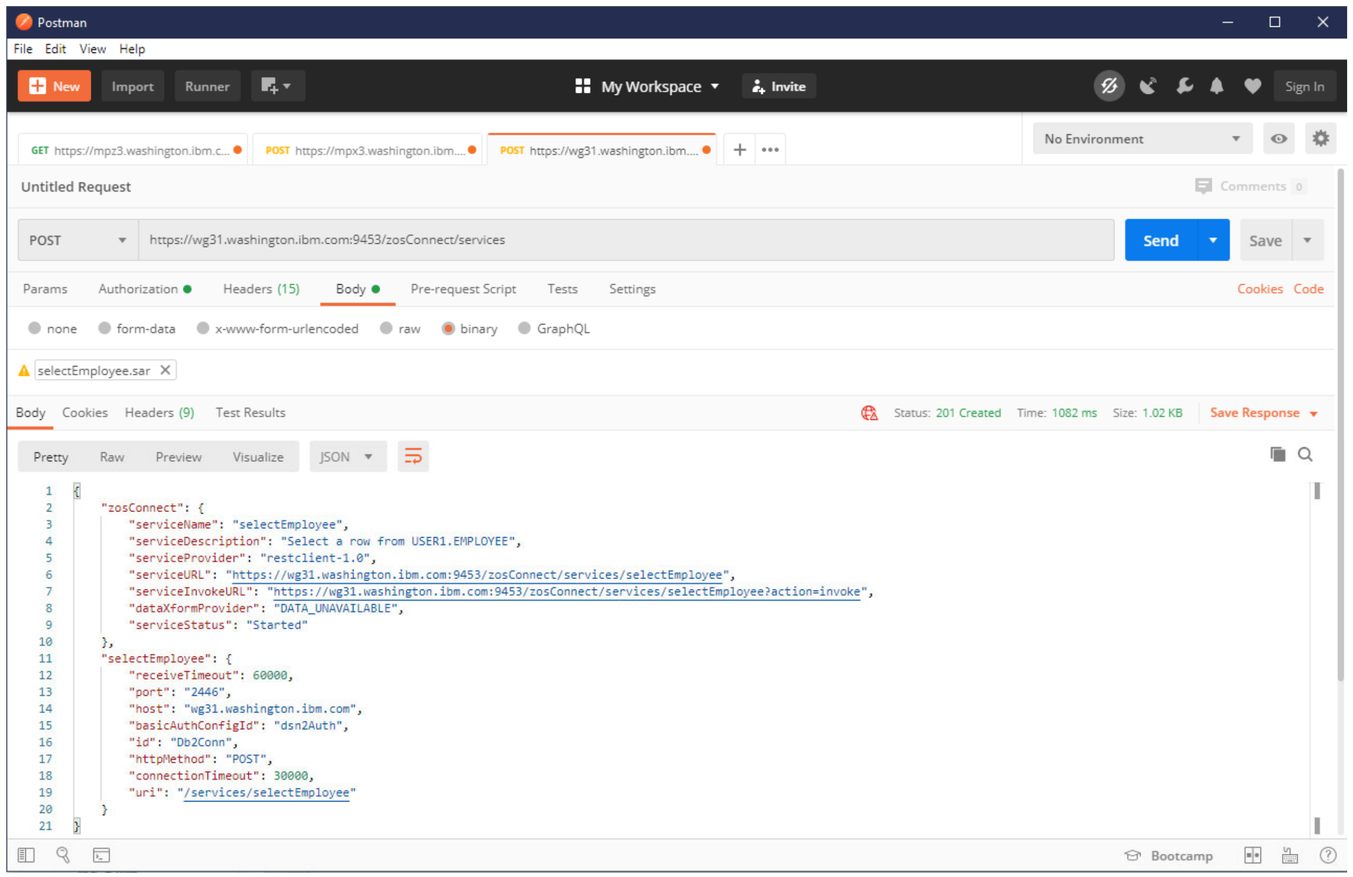
- **Using zconbt to create an API archive file**

```
C:\z\workspaces\zCEE>..\..\software\zconbt\bin\zconbt --
projectDirectory=C:/z/workspaces/zcee/miniloan --file miniloan.aar
BAQB0000I: z/OS Connect Enterprise Edition 3.0 Build Toolkit Version 1.10 (20220817-1609).
BAQB0028I: Creating API archive file from API project directory
C:/z/workspaces/zcee/miniloan.
BAQB0029I: Successfully created API archive file miniloan.aar.
```

mitchj@us.ibm.com

Deploying Service and API Archive files outside of the Eclipse tooling

- Use SAR or AAR files as request message and use HTTP POST
- Use the z/OS Connect administrative RESTful APIs
- Use Postman or cURL (client for URL)



• Deploy a service archive file

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

• Deploy an API archive file

```
curl --data-binary @Catalog.aar
--header "Content-Type: application/zip"
https://mpxm:9453/zosConnect/apis
```

Results:


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


mitchj@us.ibm.com

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

Open API 3
z/OS Connect Designer

Accessing z/OS resources from API using the z/OS Connect Designer

mitchj@us.ibm.com

The specification description provides the details of the API (OpenAPI 3)

cscvinc.yaml

File Edit View

```yaml
  "/employee/{employee}":
    get:
      tags:
        - Employee
      operationId: getEmployeeSelectService
      x-ibm-zcon-roles-allowed:
        - Staff
      parameters:
        - name: Authorization
          in: header
          required: false
          schema:
            type: string
        - name: employee
          in: path
          required: true
          schema:
            type: string
            maxLength: 6
      responses:
        "200":
          description: OK
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/getEmployeeSelectService_response_200"
        "404":
          description: Not Found
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/getEmployeeSelectService_response_404"
        "500":
          description: Severe Error
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/getEmployeeSelectService_response_500"
  put:
    tags:
      - Employee
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

cscvinc.yaml

File Edit View

```yaml
getEmployeeSelectService_response_200:
  type: object
  properties:
    summary:
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_message'
    detail:
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_detail'
getEmployeeSelectService_response_200_message:
  type: object
  properties:
    message:
      type: string
  example:
    message: record retrieved
getEmployeeSelectService_response_200_detail:
  type: object
  properties:
    EmployeeSelectServiceOperationResponse:
      type: object
      properties:
        employeeData:
          type: object
          properties:
            response:
              type: object
              properties:
                employeeDetails:
                  type: object
                  properties:
                    employeeNumber:
                      type: string
                      maxLength: 6
                    name:
                      type: string
                      maxLength: 20
                    address:
                      type: string
                      maxLength: 20
                    phoneNumber:
                      type: string
                      maxLength: 8
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

mitchj@us.ibm.com

Let’s stop for a moment: A lesson learned the hard way. . . .
z/OS Native Server Considerations – API Provider Deployment

Important: In order for multiple API project `.war` files to function when deployed to a single z/OS Connect native server, you must ensure that your OpenAPI specification includes a `contextRoot` defined within the `servers` section. The `contextRoot` attribute specifies the entry point of the deployed application. For example, to use a context root of `/myContextRoot` in the API's OpenAPI definition server's section:

```
openapi: 3.0.0
...
servers:
url: "https://localhost:9443/myContextRoot"
...
```

This definition must match the `contextRoot` attribute value of the `webApplication` element in your `server.xml` file. An example of the configuration might be:

```
<webApplication location="${server.config.dir}/apps/api.war" name="EmployeesApi" contextRoot="/myContextRoot"/>
```

If the server entry in the server section of the OpenAPI definition includes a `contextRoot` value, then this value must be specified in the `contextRoot` attribute of the corresponding `webApplication` element, even when only a single API is deployed to the z/OS Connect server.

Each API deployed to the same IBM z/OS Connect server requires a unique context root.

4. Copy the server configuration.
   Copy the server configuration files from the API project `/scr/main/liberty/config` directory into the `${server.config.dir}/configDropins/overrides` directory of the server or another directory via FTP. For more information, see Overview of IBM z/OS Connect Server configuration
5. Deploy the generated API files to the z/OS Connect native server
   Deploy API `.war` files into a permanent USS directory. An example directory, such as the one provided by the z/OS Connect native server template, is `${server.config.dir}/apps`. API files can also be deployed to other directories, such as in separately mounted zFS file systems.

   For each deployed API define a webApplication element in the configuration file. An example element is included in the supplied openApi3 server template. An example element might be the following:

```
<webApplication id="My API" location="${server.config.dir}/apps/api.war" name="MyAPI"/>
```

https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=overview-api-provider-devops

mitchj@us.ibm.com

z/OS Server Considerations – API Context Root

https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=devops-zos-connect-server-overview

The drop-ins directory

The *drop-ins* directory, `/config/dropins` is a special directory that is supported by WebSphere® Application Server for Liberty. It allows `.war` files to be deployed and dynamically loaded into the running IBM z/OS Connect with no additional definitions that are required in the configuration file.

By default, z/OS Connect Designer deploys the API `.war` file to this directory. Using the same directory in your API container image simplifies the creation of that image because the configuration remains the same.

A directory other than drop-ins

This is required in any of the following situations:

- The API's OpenAPI definition server's section contains server entry that includes a context root value, which is not just /.
- Multiple APIs are to be deployed to the same IBM z/OS Connect container. Because the API `.war` file will be generated with a context root of /, and multiple API `.war` files in the same server must have unique context root values.

  You need to include a context root value (not /) in the API's OpenAPI definition server's section, for example to use a context root of `/MyCompany`:

```
openapi: 3.0.0
   ...
   servers:
      url: https://localhost:9443/MyCompany
   ...
```

- Requests to start an API require authentication only, without authorization, so the authorization roles need to be mapped to the WebSphere Application Server for Liberty special subject ALL_AUTHENTICATED_USERS. For more information, see How to define authorization roles.

If you choose not to use the drop-ins directory, you must alter the configuration that is used in z/OS Connect Designer during the creation of the API container image.

The samples as provided by z/OS Connect are not suitable as is for deployment to a z/OS Connect Native server.

- **Server XML configuration required to define applications and add context root**

```
<webApplication id="cics" contextRoot="/cics"  name="cicsAPI"
      location="${server.config.dir}apps/cscvinc.war"/>
<webApplication id="db2" contextRoot="/db2"  name="db2API"
      location="${server.config.dir}apps/employees.war"/>
```

mitchj@us.ibm.com

The API’s specification required updates

cscvinc.json - Notepad

File Edit Format View Help

```
{
    "swagger": "2.0",
    "info": {
        "description": "",
        "version": "1.0.0",
        "title": "cscvincapi"
    },
    "basePath": "/cscvincapi",
    "schemes": [
        "https",
        "http"
    ],
    "consumes": [
        "application/json"
    ],
    "produces": [
        "application/json"
    ],
    "paths": {
        "/employee/{employee}": {
            "get": {
                "tags": [
                    "cscvincapi"
                ],
                "operationId": "getCscvincSelectService",
                "parameters": [
                    {
                        "name": "Authorization",
                        "in": "header",
                        "required": false,
                        "type": "string"
                    },
                    {
                        "name": "employee",
                        "in": "path",
                        "required": true,
                        "type": "string",
                        "maxLength": 6
                    }
                ],
                "responses": {
                    "200": {
                        "description": "OK",
                        "schema": {
                            "$ref": "#/definitions/getCscvincSelectService_response_200"
                        }
                    },
                    "404": {
                        "description": "Not Found",
```

Ln 18, Col 7 | 100% | Windows (CRLF) | UTF-8

cscvinc.yaml - Notepad

File Edit Format View Help

```
openapi: 3.0.1
info:
  title: cscvinc
  description: ""
  version: 1.0.0
servers:
- url: /cscvinc
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
      - cscvinc
      operationId: postCscvincInsertService
      x-ibm-zcon-roles-allowed:
        - Staff
      parameters:
      - name: Authorization
        in: header
        schema:
          type: string
      requestBody:
        description: request body
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/postCscvincInsertService_request'
        required: true
      responses:
        200:
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/postCscvincInsertService_response_200'
    x-codegen-request-body-name: postCscvincInsertService_request
/employee/{employee}:
  get:
    tags:
    - cscvinc
    operationId: getCscvincSelectService
    x-ibm-zcon-roles-allowed:
      - Staff
    parameters:
    - name: Authorization
      in: header
      schema:
        type: string
```

Ln 44, Col 16 | 100% | Unix (LF) | UTF-8

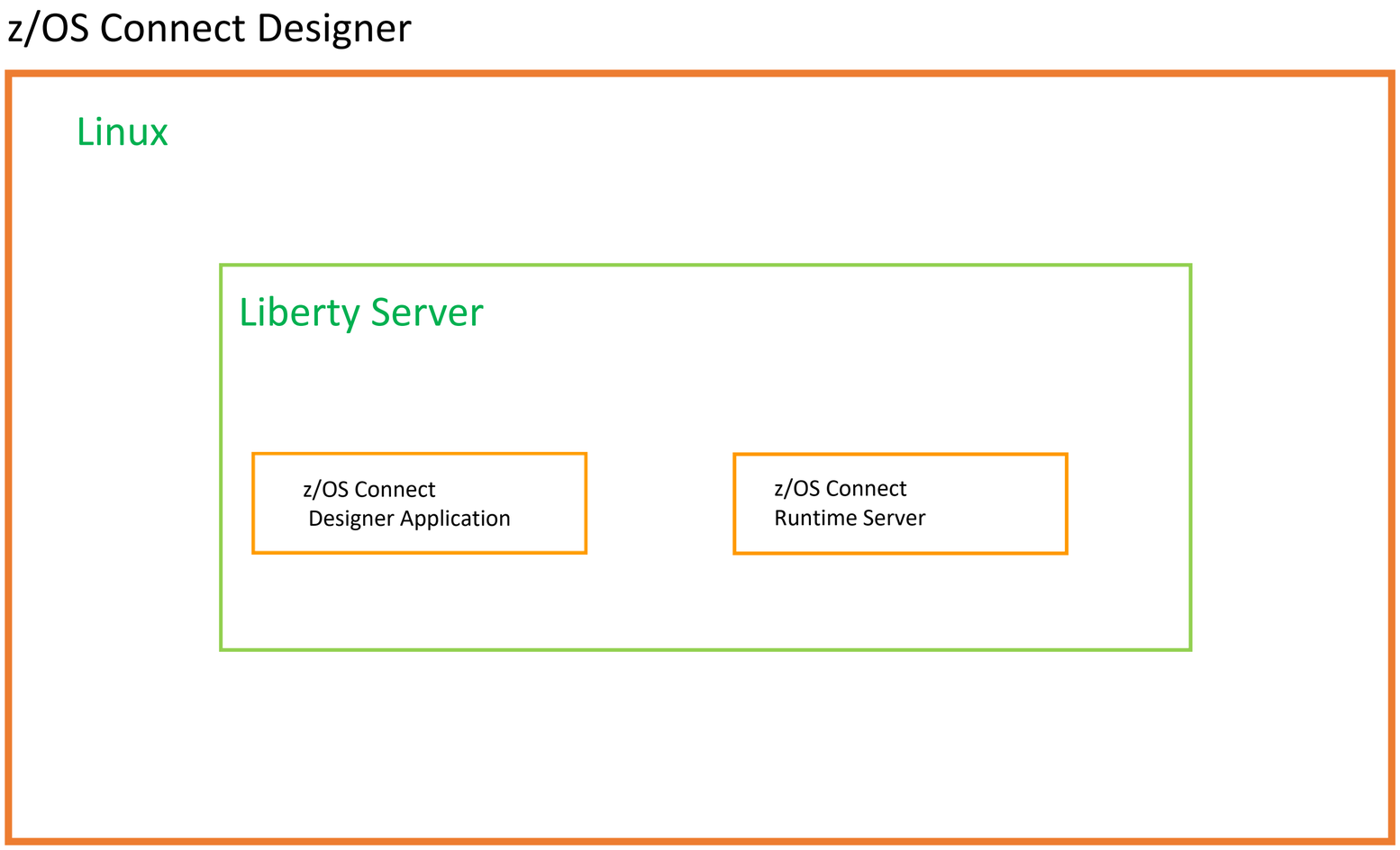
Where to make these Is the question.

We recommend making the changes in the specification file before it is imported into the Designer.

mitchj@us.ibm.com

Let’s explore the z/OS Connect Designer Topology

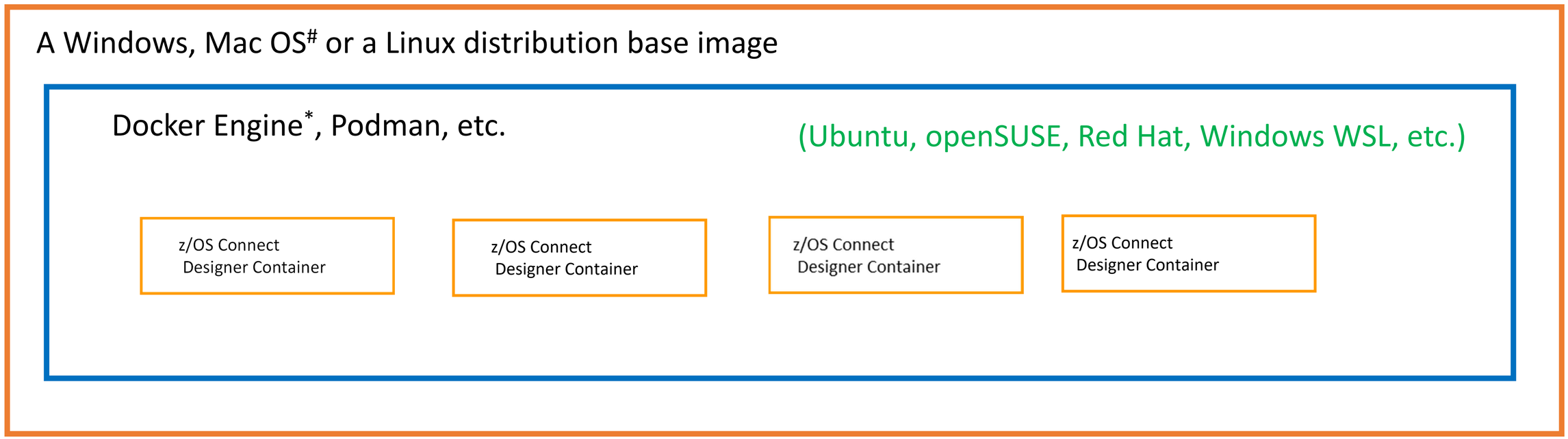
- A z/OS Connect Designer is composed of a self-contained Linux environment configured with a Liberty server running a z/OS Connect Designer application ***and*** a z/OS Connect runtime server, in total, known as a “container”.
- Running a Designer container requires a container runtime environment.



mitchj@us.ibm.com

z/OS Connect Designer runtime options

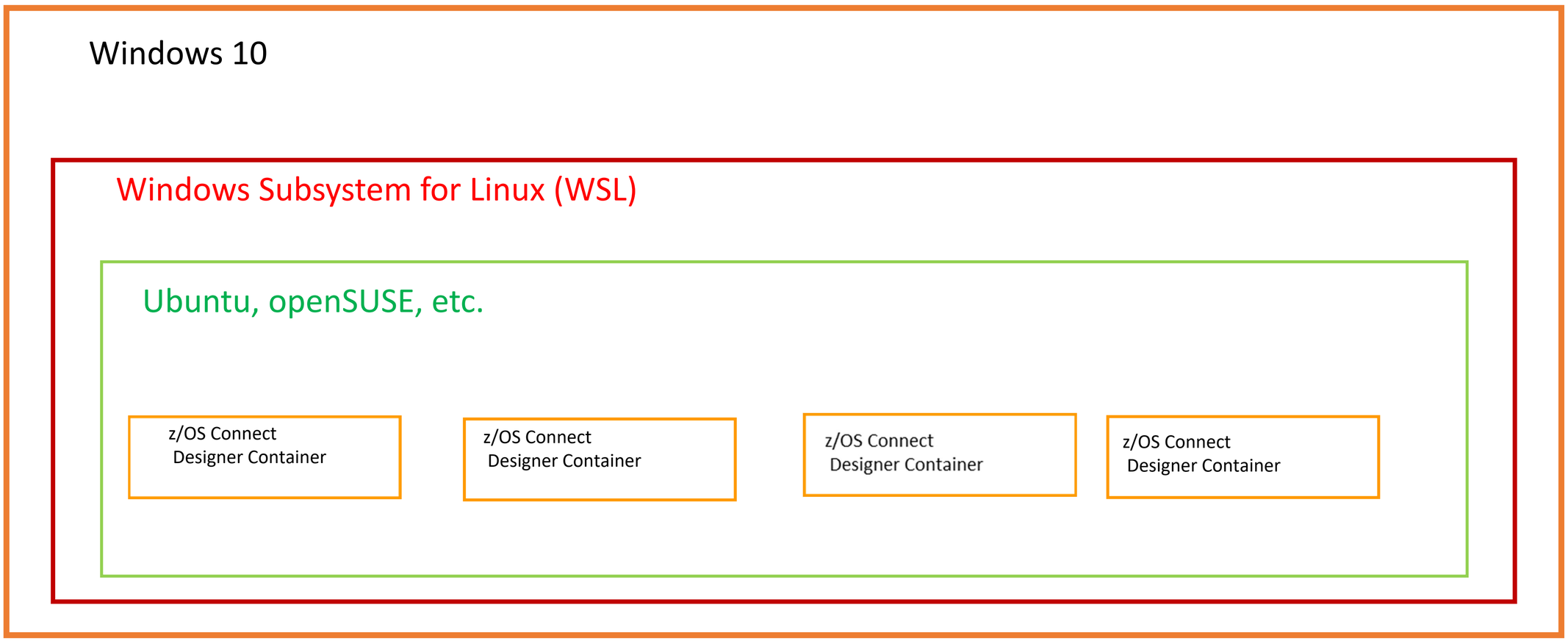
one API per container, multiple Designer containers may be required



#Warning, Machines with Mac M1 Pro processors may be problematic.
*Docker Engine is not an option on Macs, use Podman

mitchj@us.ibm.com

Windows considerations

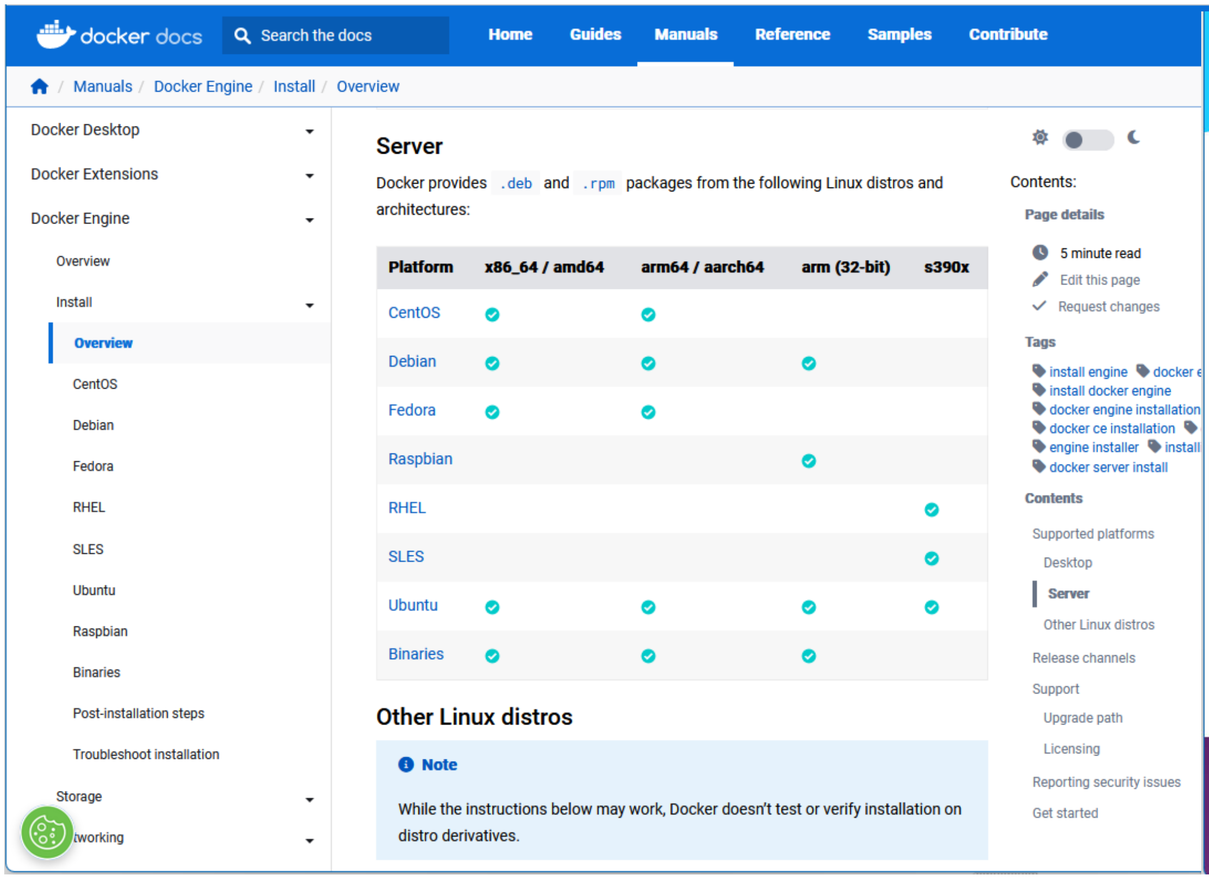


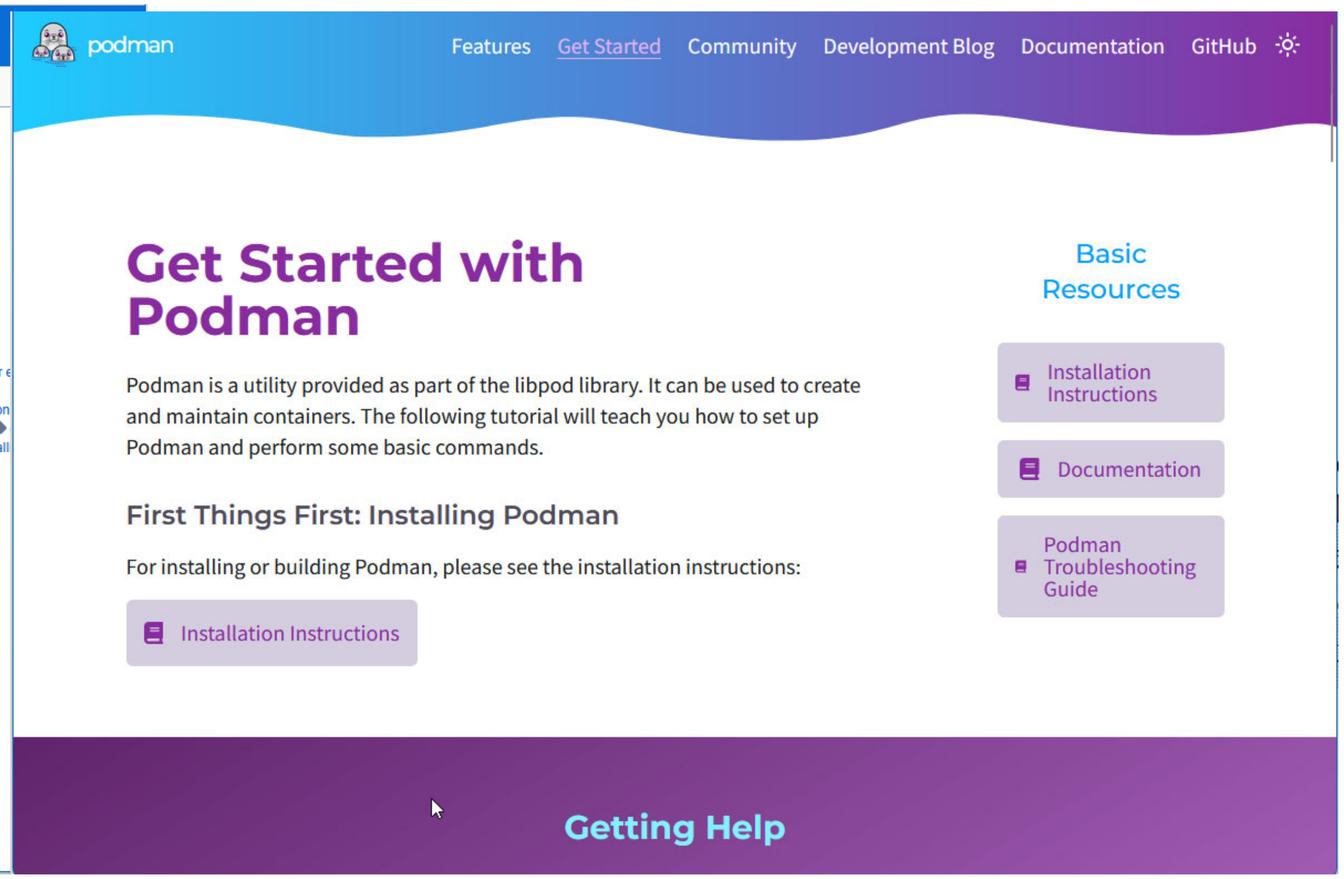
A z/OS Connect Designer is an application running in Liberty running in Linux (the container) which is running Linux (the container runtime) running in WSL running in Windows.

mitchj@us.ibm.com

Tech-Tip: Some suggested runtime environments that required no license

one goal to show you why you don't need the GUI tools (or why I am considered a neanderthal)





https://docs.docker.com/engine/install/#server

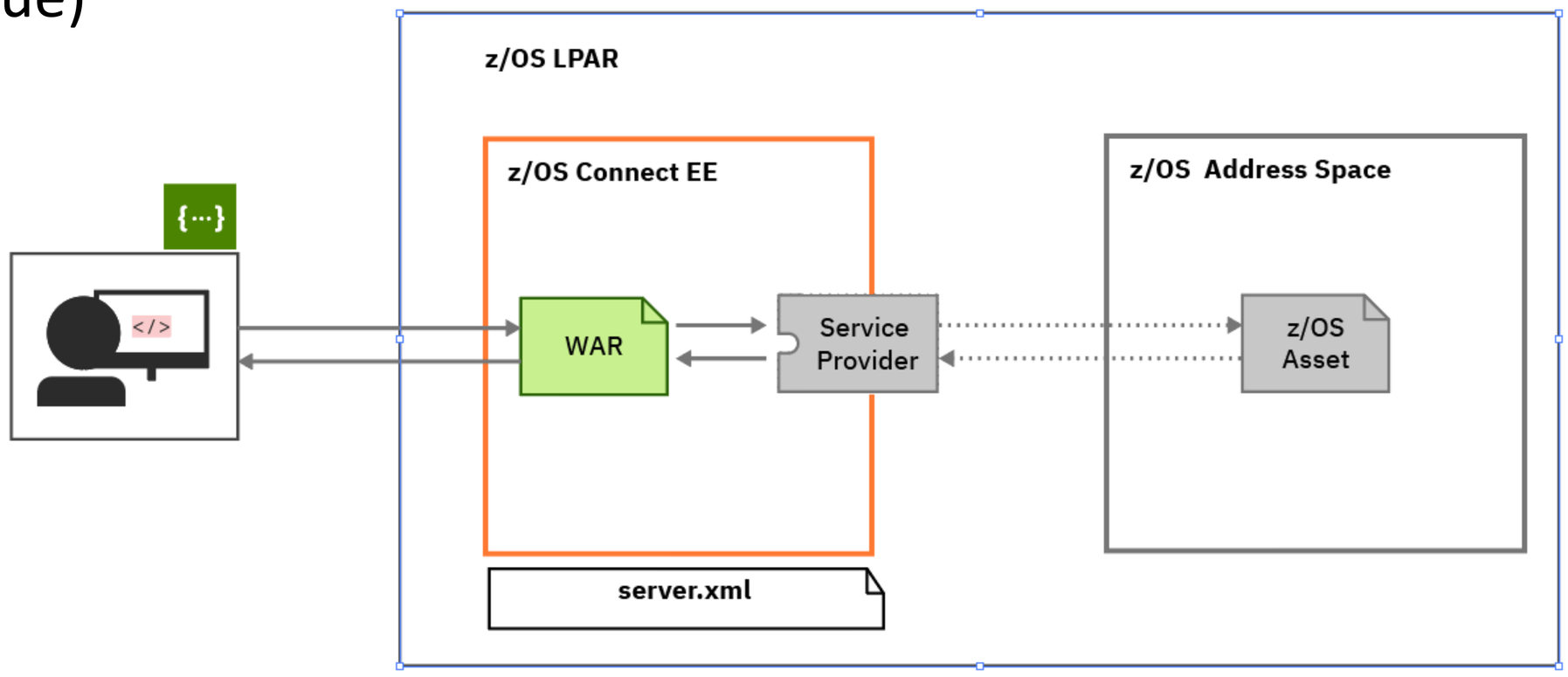
https://podman.io/get-started

Important: The command line interface (CLI) syntax is the same between the Docker Engine and Podman. Just change a Docker command from using the ***docker*** command to the ***podman*** command, e.g., ***docker ps -a*** when using Docker Engine becomes ***podman ps –a*** when using Podman.

mitchj@us.ibm.com

Accessing the CICS, Db2 or IMS asset (Open API 3)

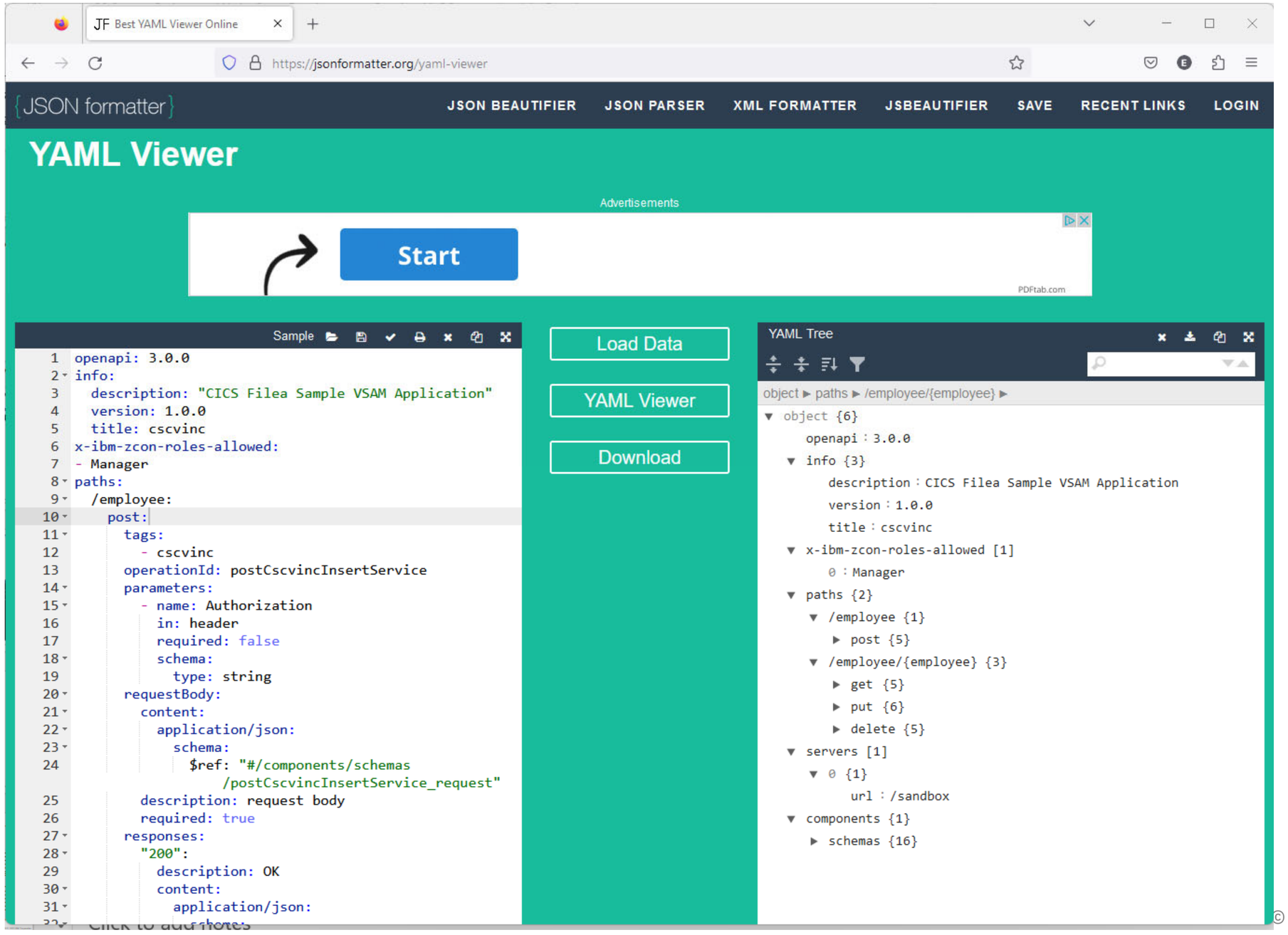
- z/OS Connect OpenAPI 3 APIs are developed using a z/OS Connect Designer web browser tool to developer Web ARchive (WAR) files (a traditional Java packaging technique)



- 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)

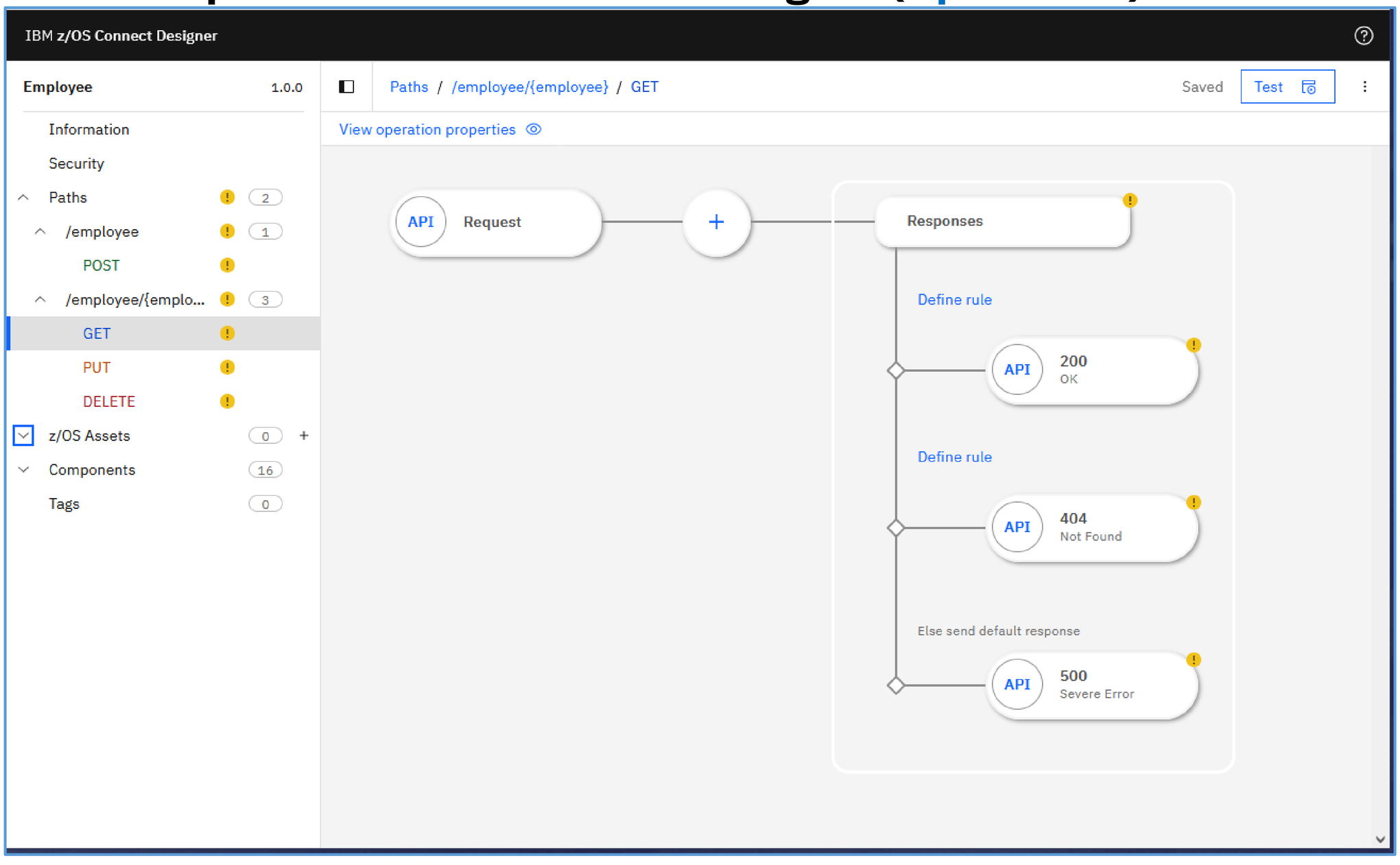
mitchj@us.ibm.com

With OpenAPI 3 you start with an YAML file description of the API



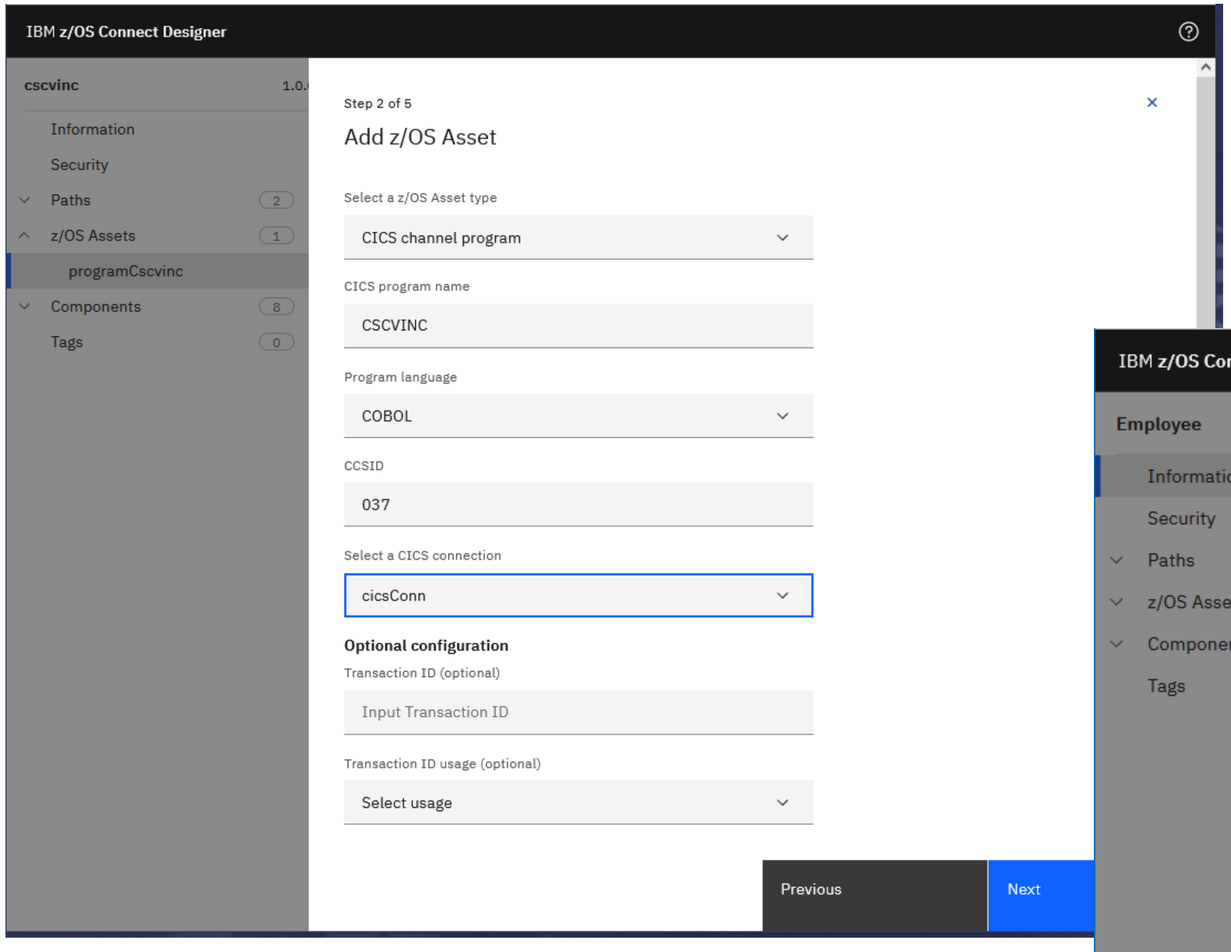
mitchj@us.ibm.com

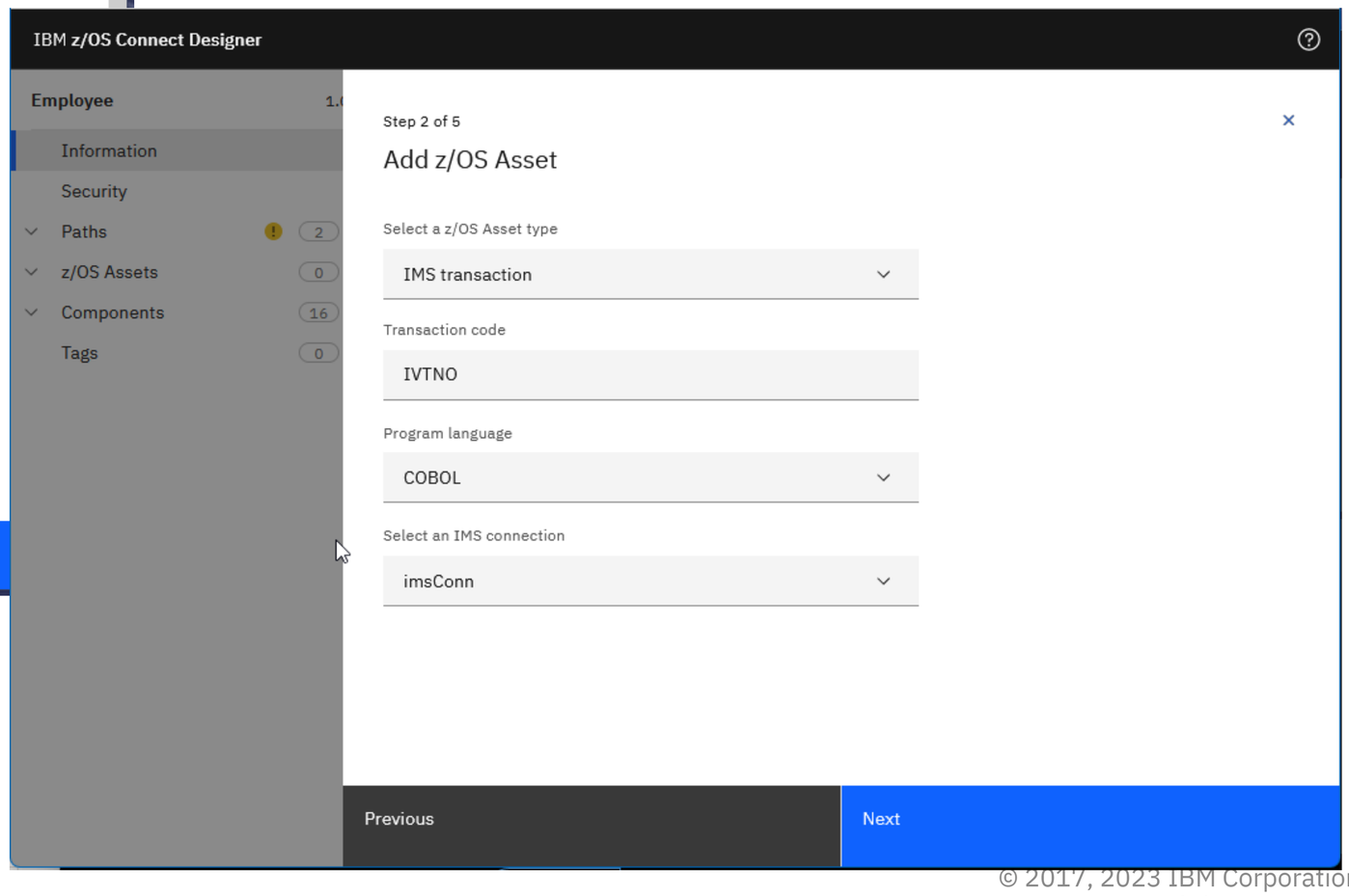
Import the description of an API into the Designer (OpenAPI 3)



mitchj@us.ibm.com

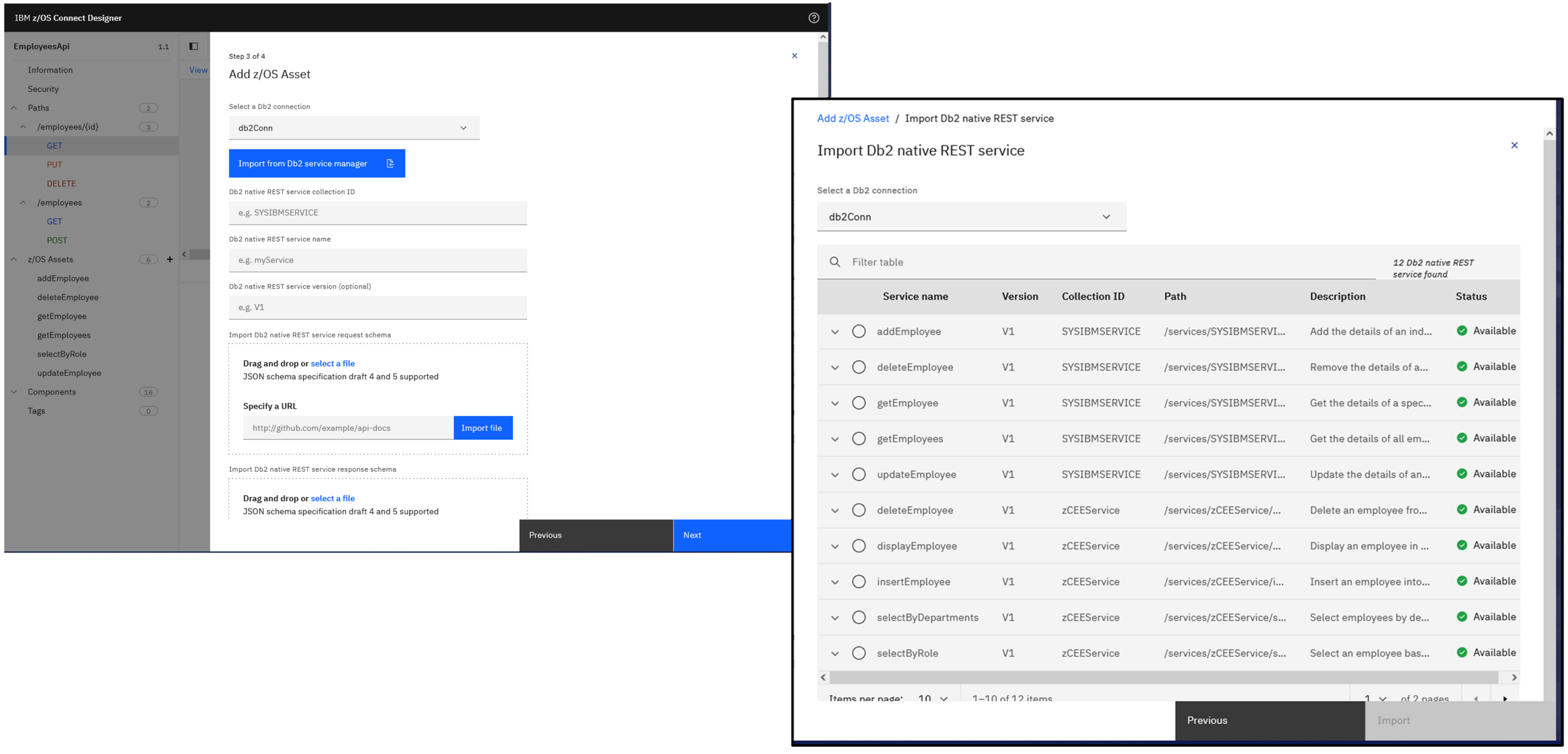
Describe the asset, a CICS program or IMS transaction (OpenAPI 3)





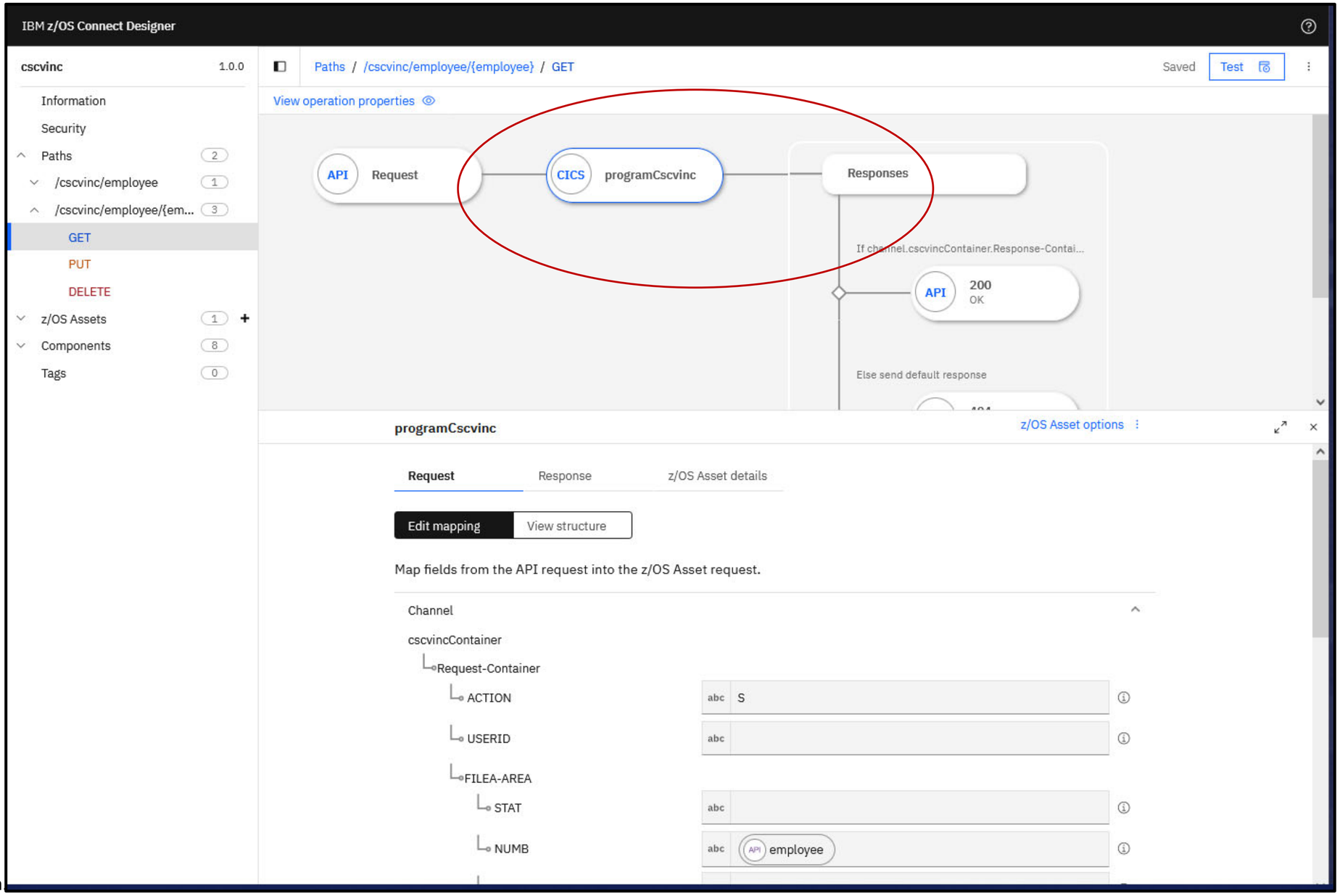
mitchj@us.ibm.com

Or a Db2 REST service (OpenAPI 3)



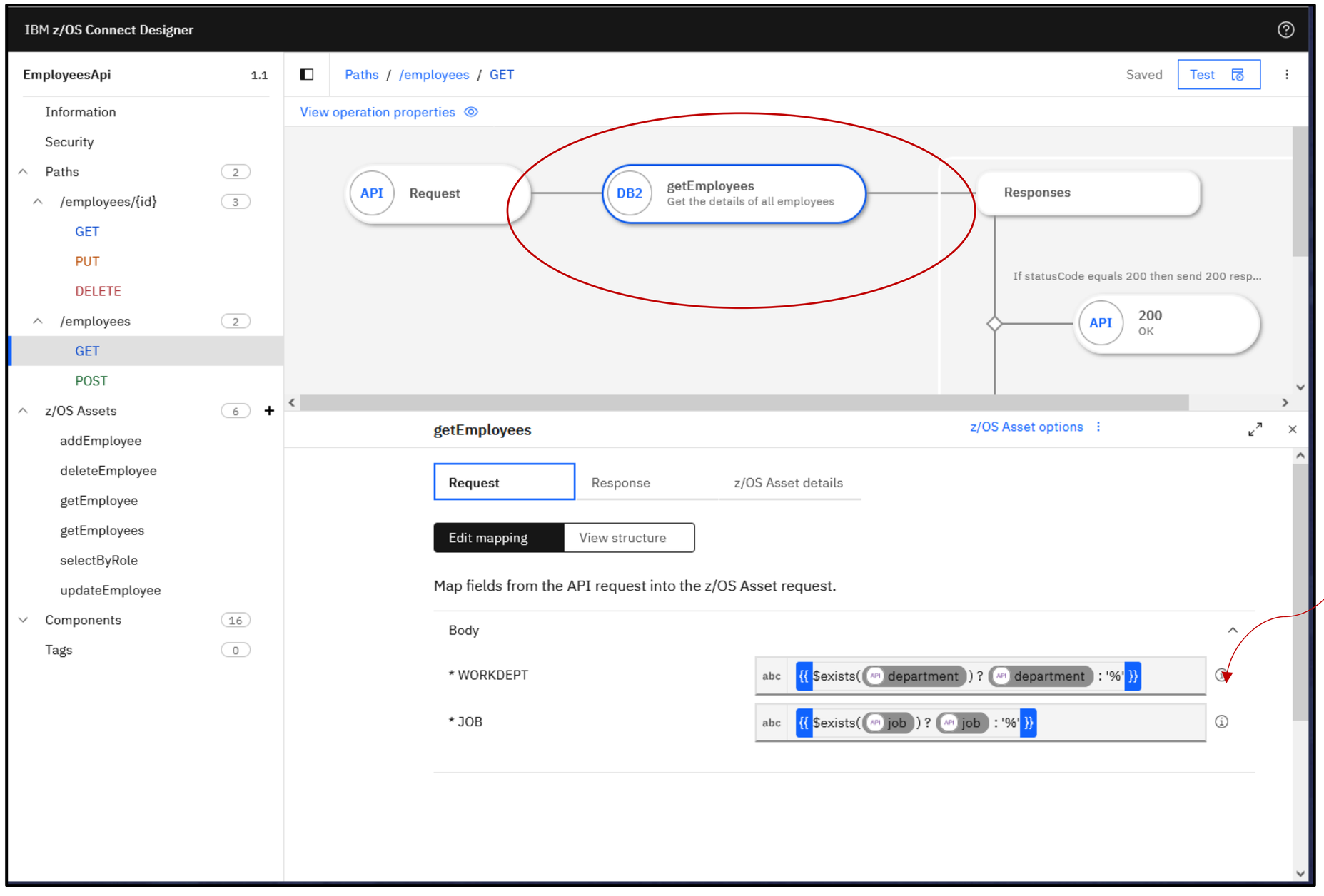
mitchj@us.ibm.com

Map the API's method and request with the resource's input (OpenAPI 3)



mitchj@us.ibm

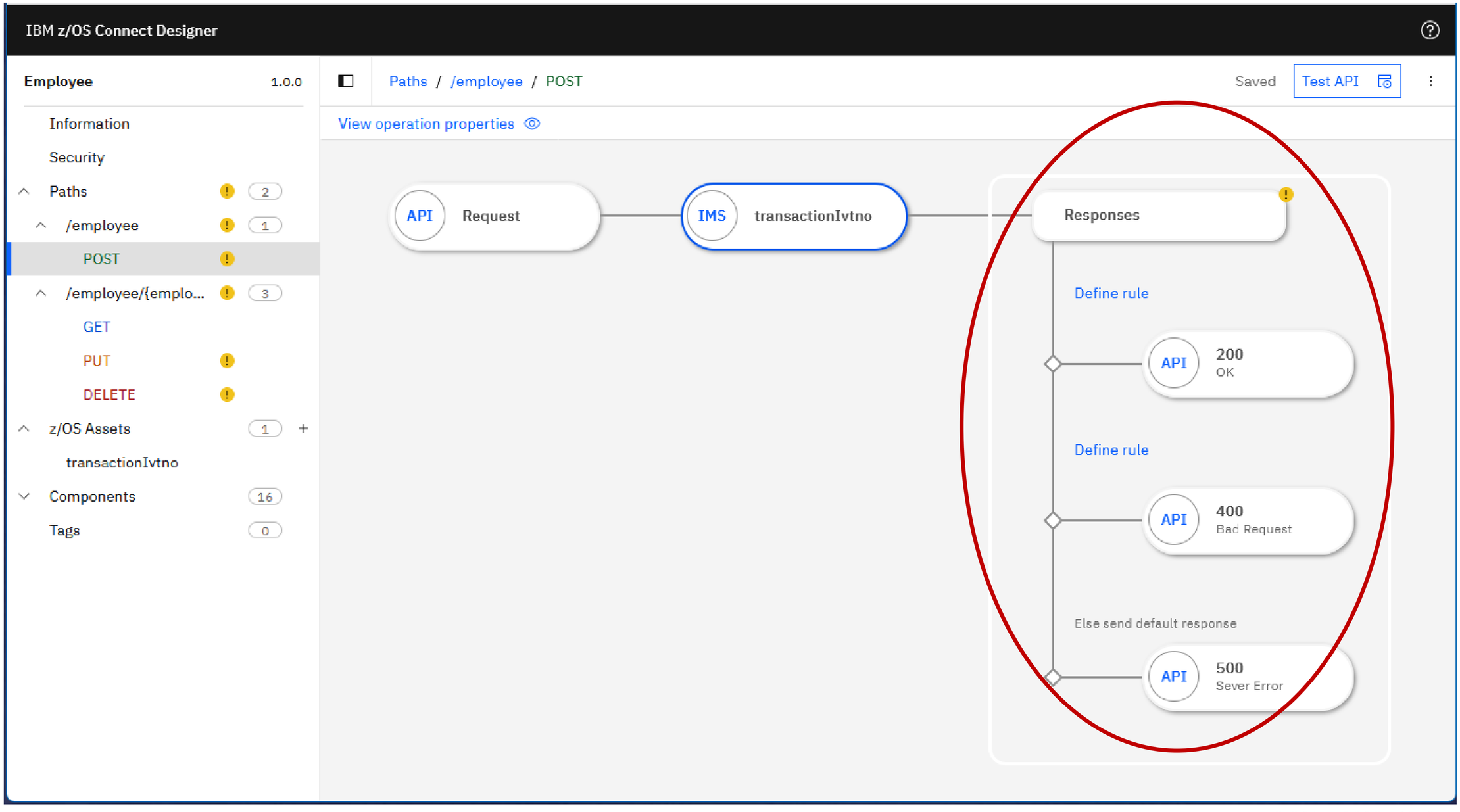
Map the API's request with a Db2 REST service's request message (OpenAPI 3)



JSONata example

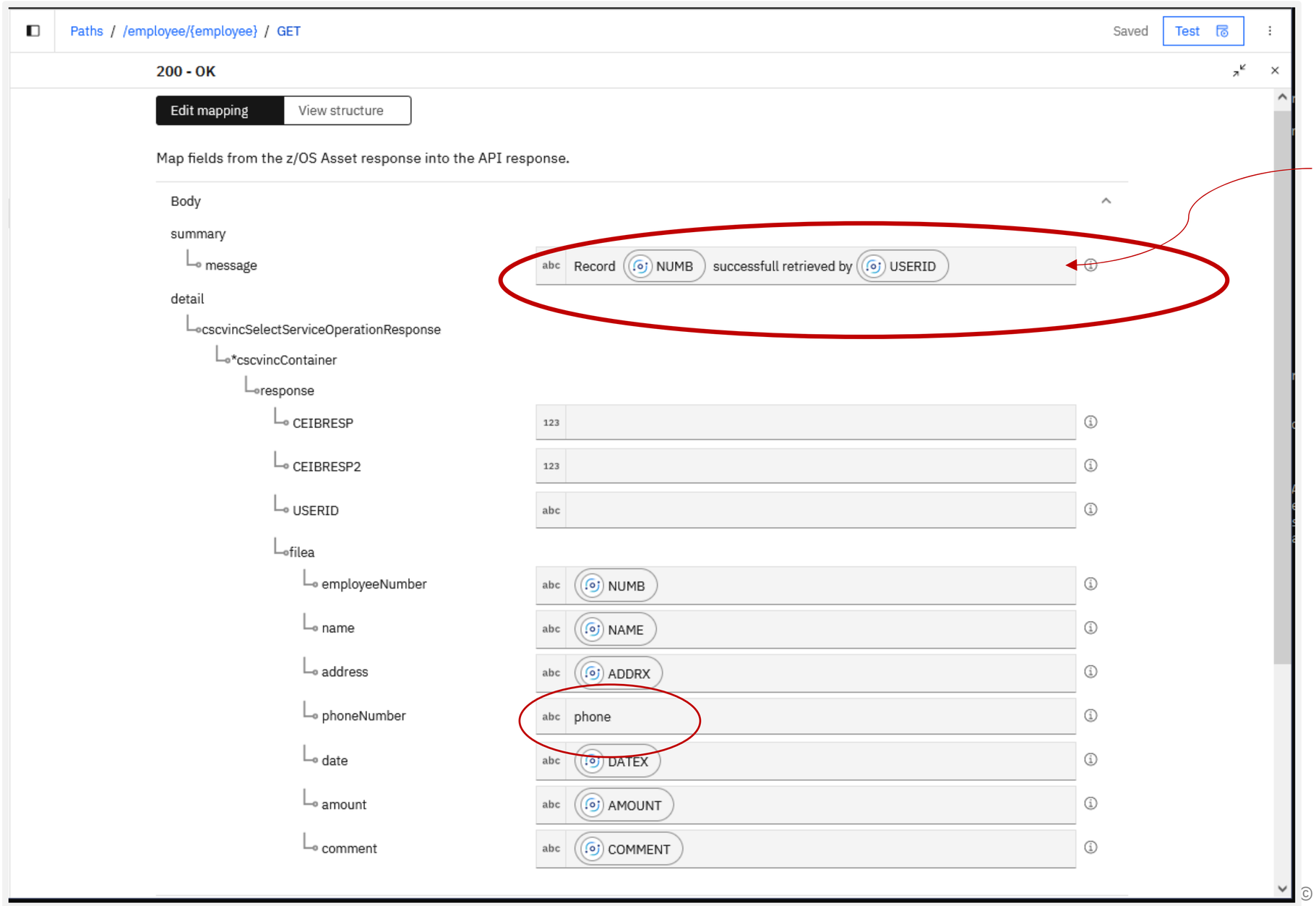
mitchj@us.ibm.com

Map the resource's results to the API's response message (OpenAPI 3)



mitchj@us.ibm.com

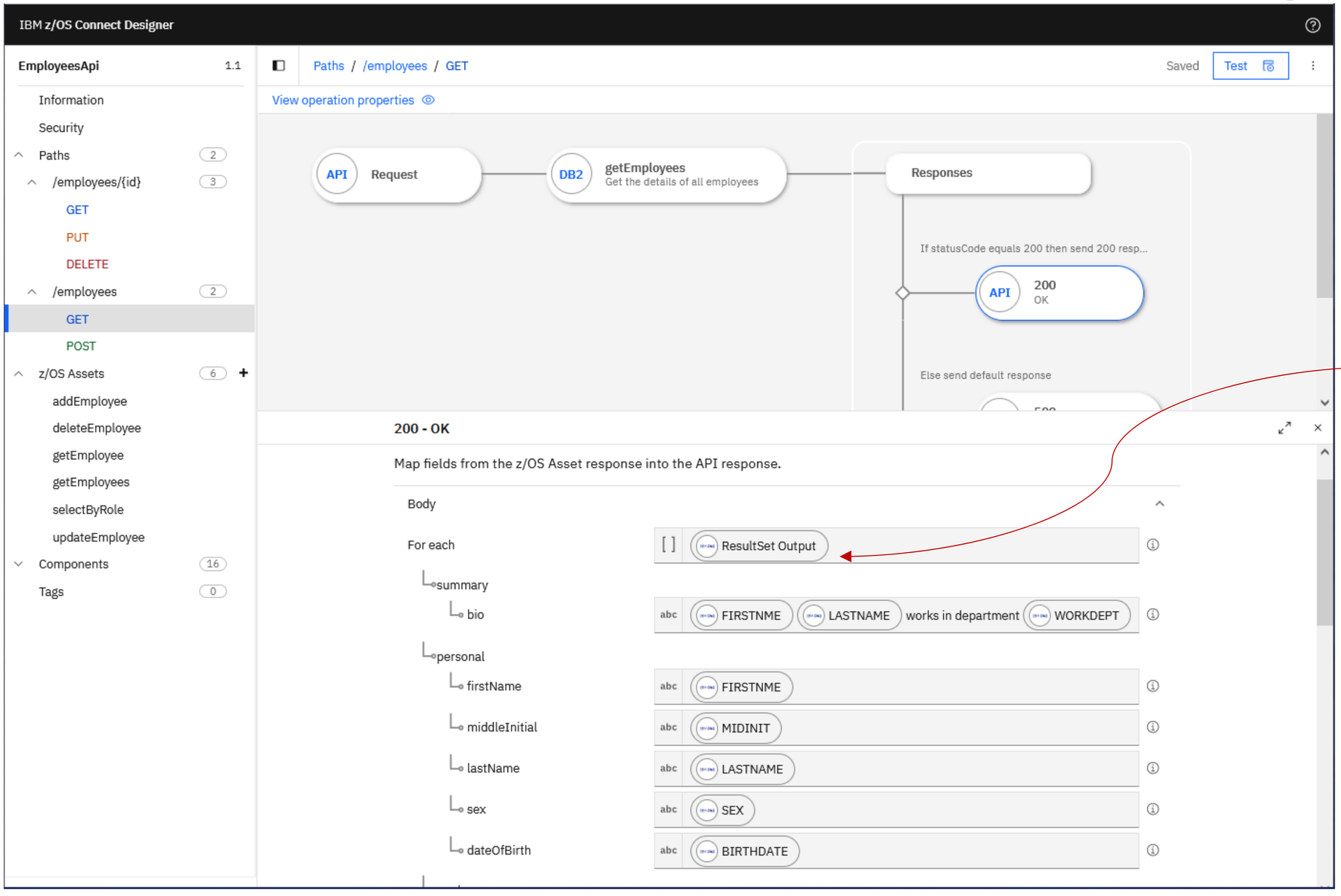
Map the resource’s response to the API’s response (200)(OpenAPI 3)



JSONata example showing the building of a complex message from the response.

mitchj@us.ibm.com

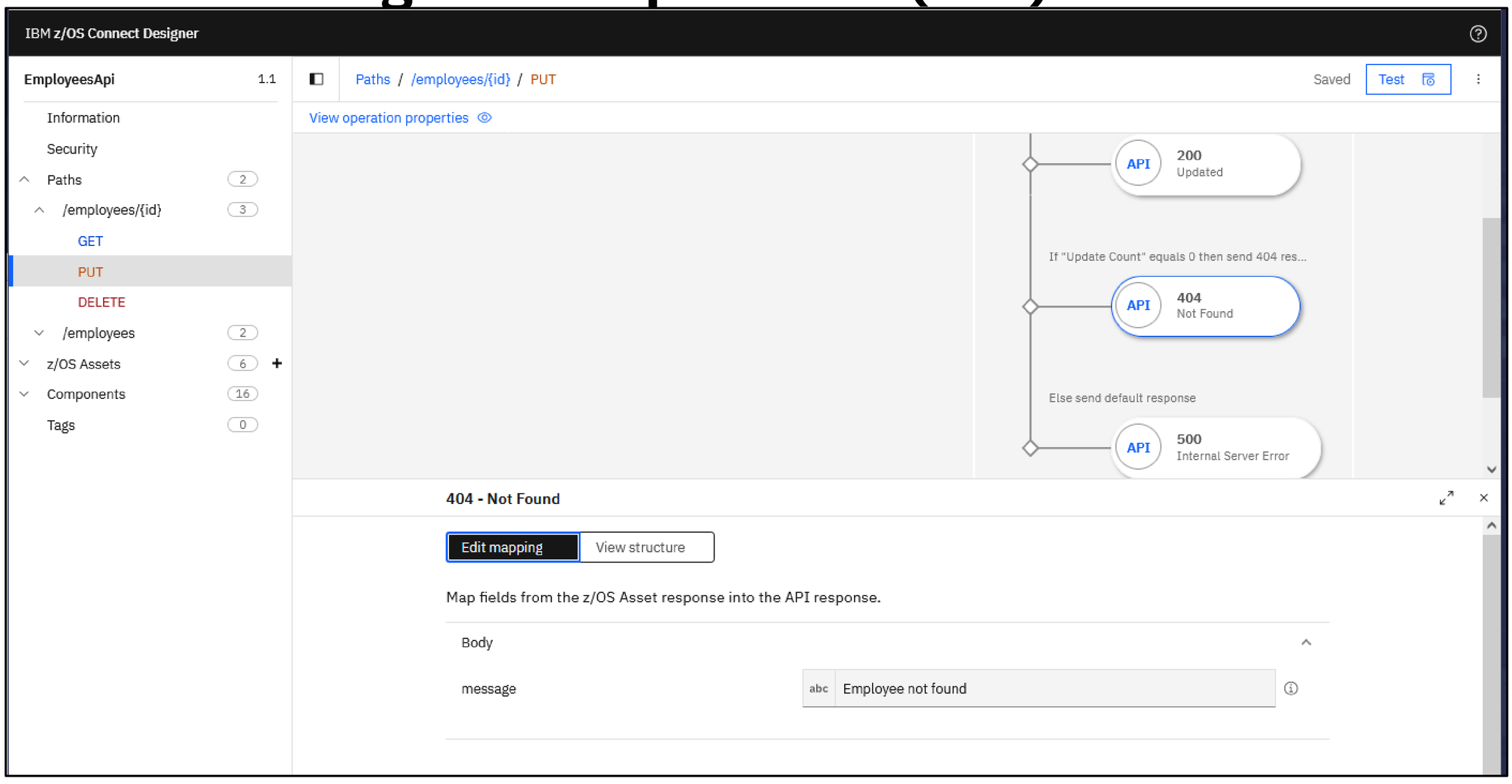
Map the Db2's response to the API's response message (OpenAPI 3)



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

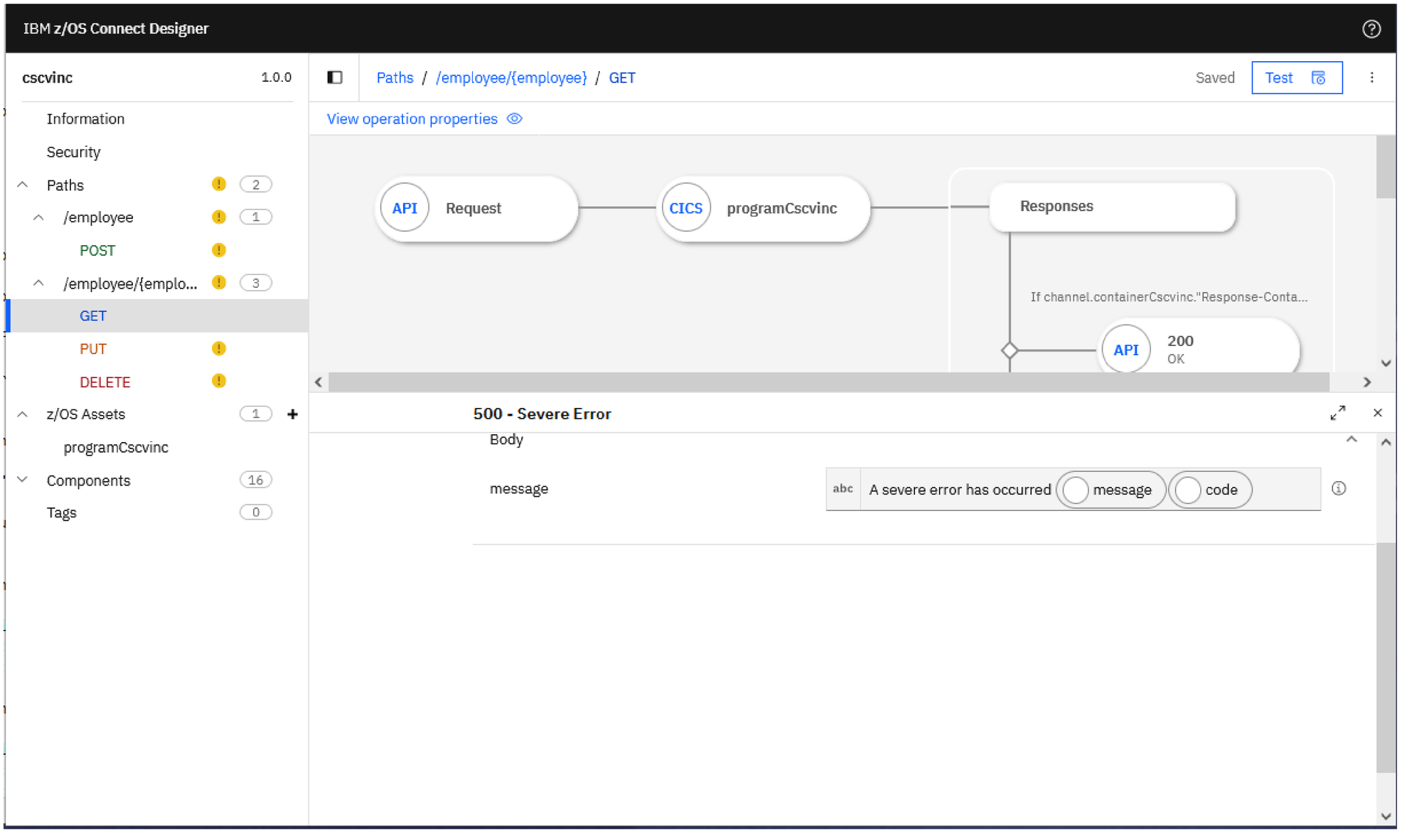
mitchj@us.ibm.com

z/OS Connect Designer for OpenAPI 3 (404)



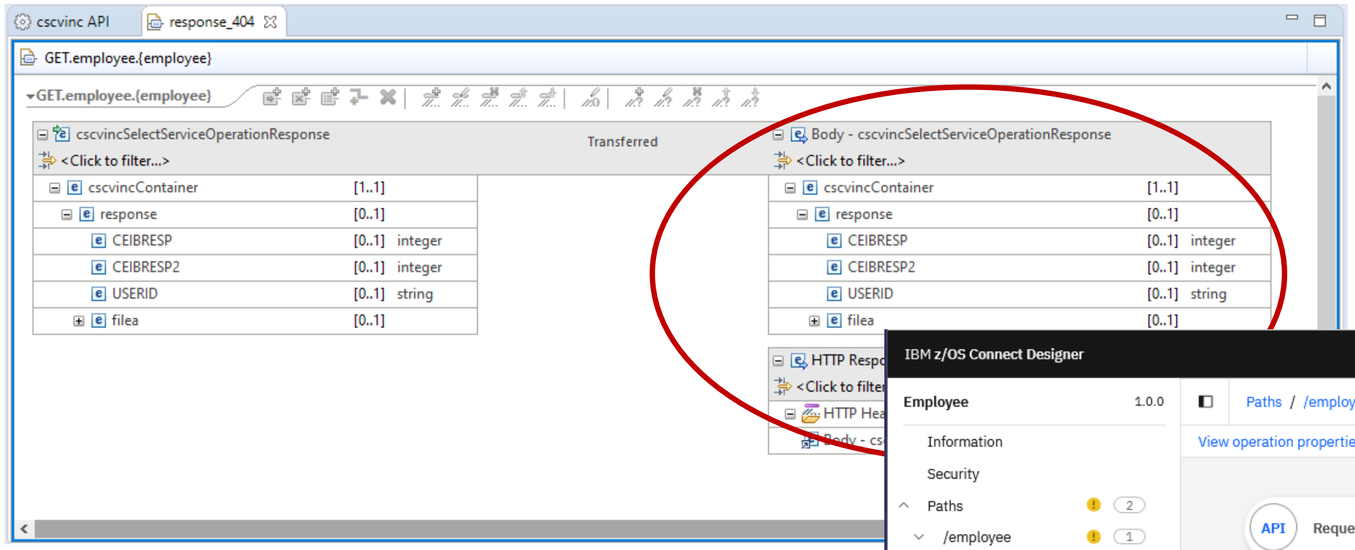
mitchj@us.ibm.com

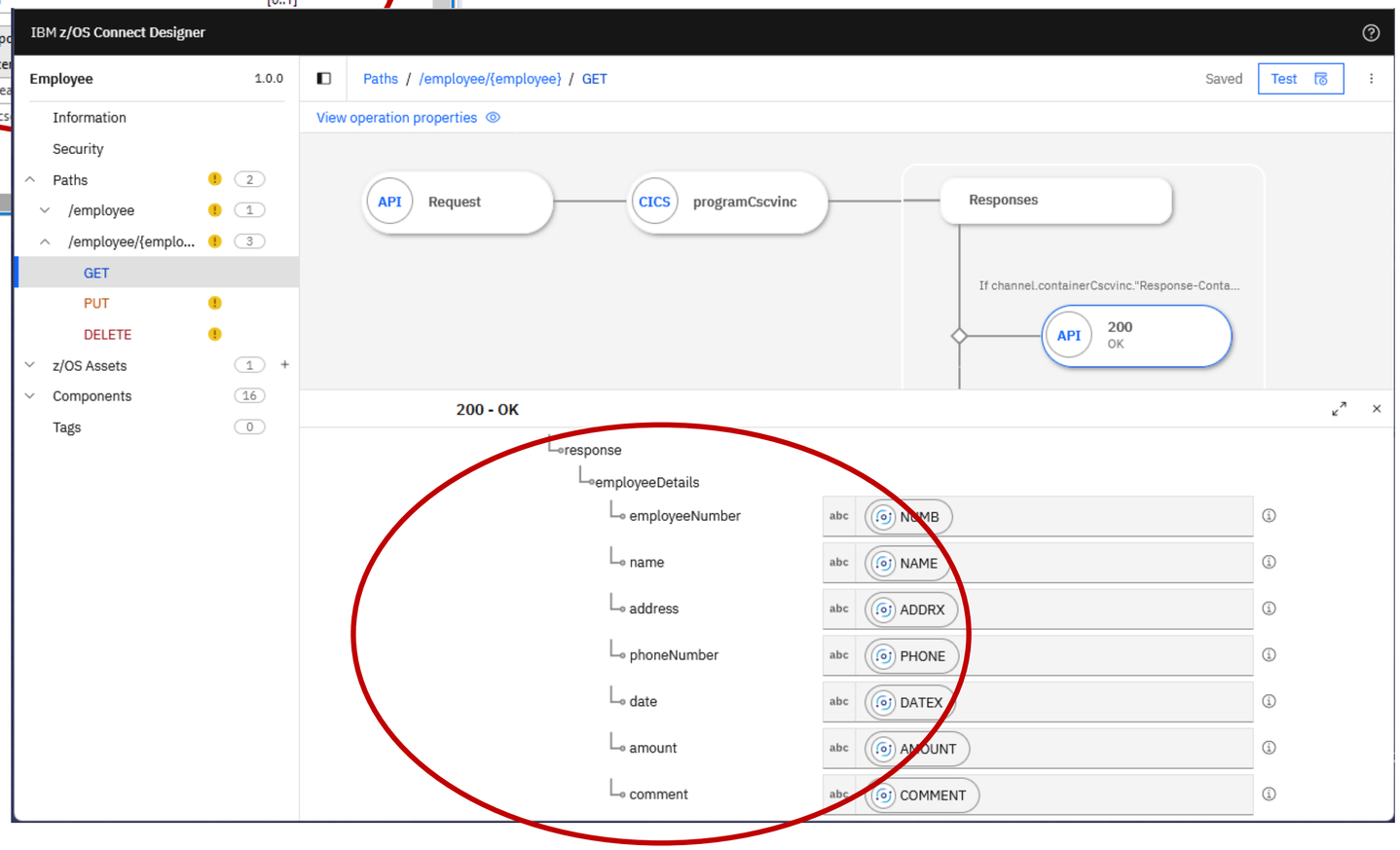
z/OS Connect Designer for OpenAPI 3 (500)



mitchj@us.ibm.com

Contrast the OpenAPI 2 versus an OpenAPI 3 Response Message



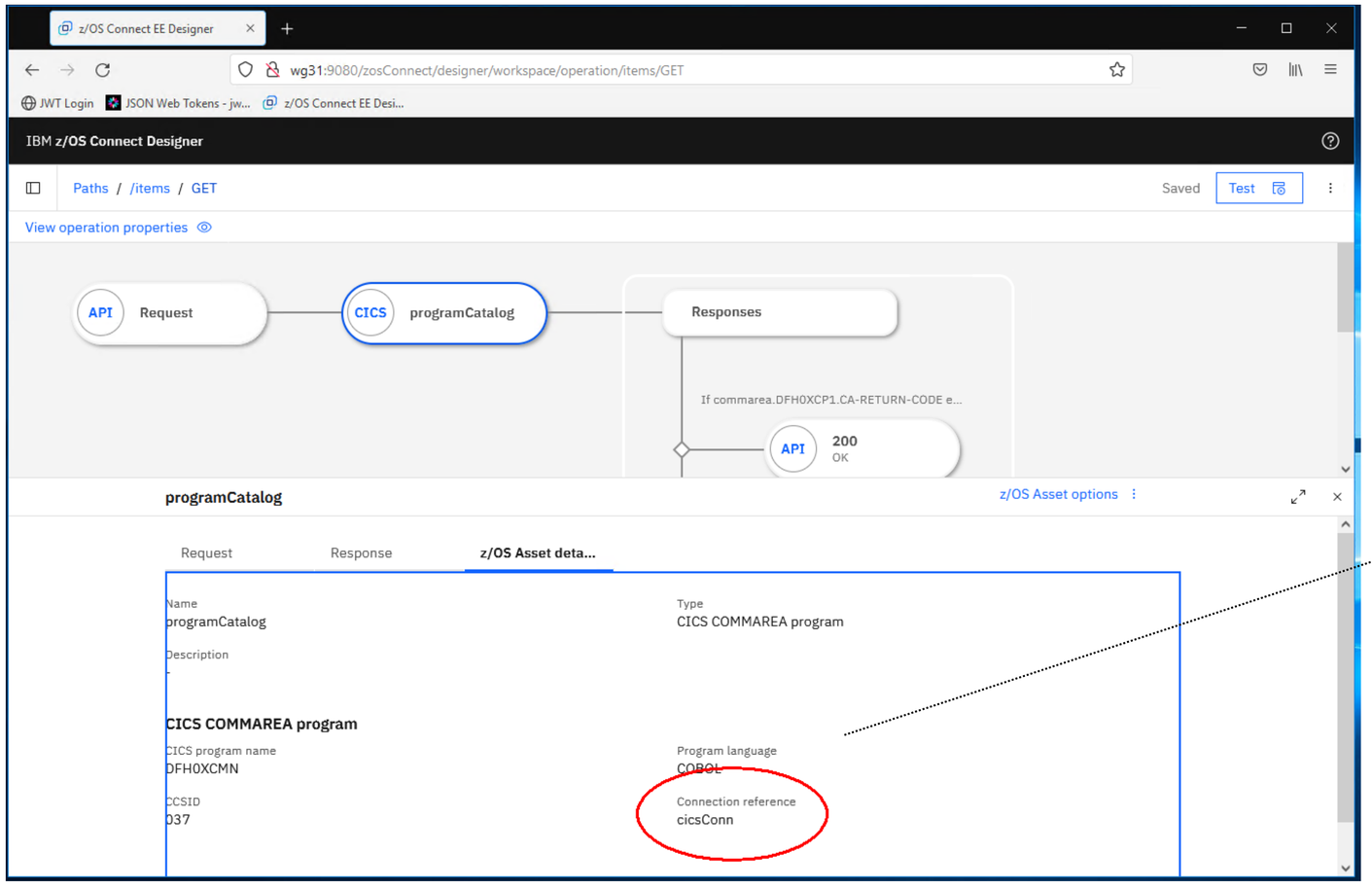


Eclipse based - z/OS Connect API Toolkit -
The contents of the API's response message are directly derived from the z/OS asset's response and usually a subset of the asset's response contents.

Web browser - z/OS Connect Designer – The contents of the API's response message are must be mapped to 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

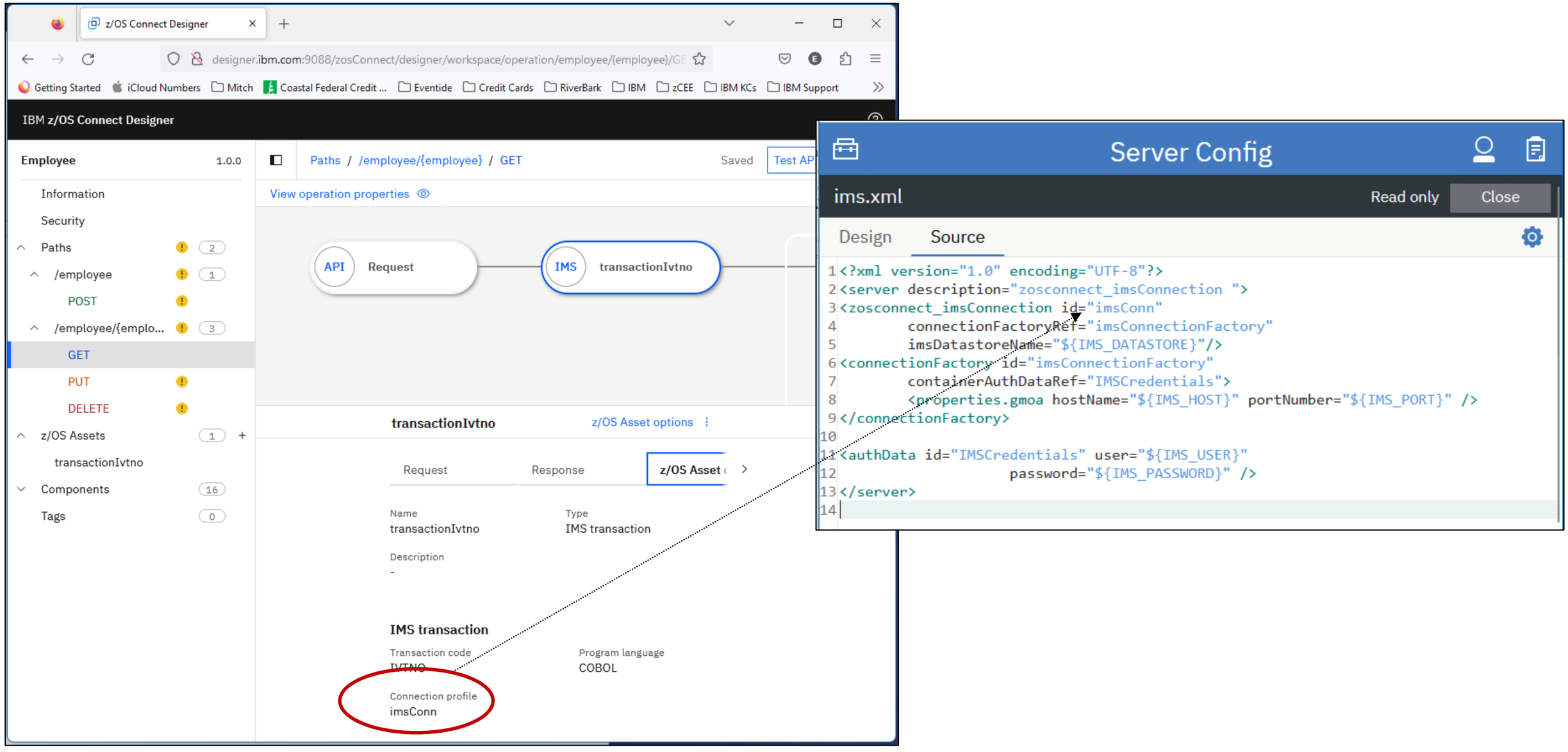
Server XML - Accessing a CICS program using IPIC (OpenAPI 3)



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

mitchj@us.ibm.com

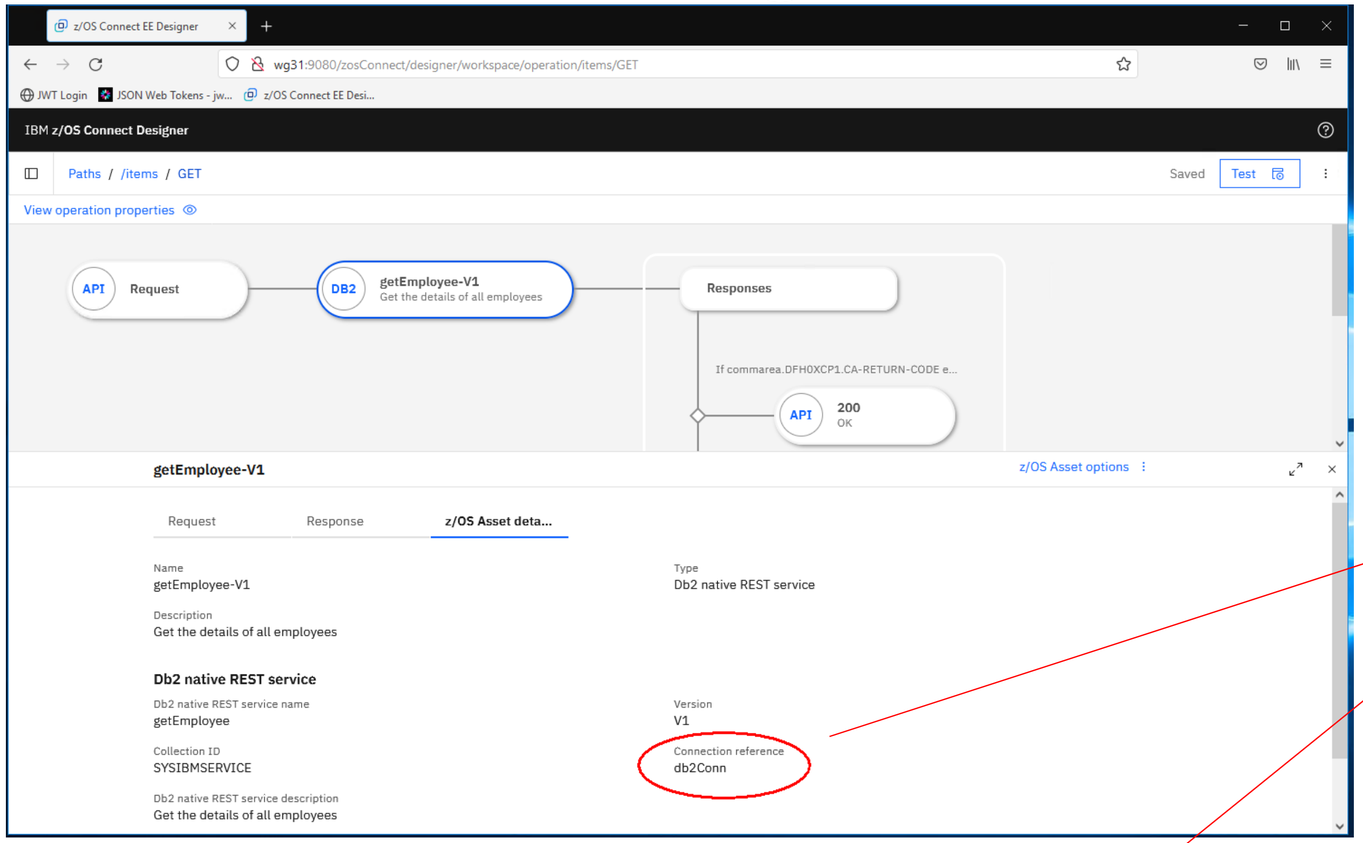
Server XML - Accessing an IMS transaction(OpenAPI 3)

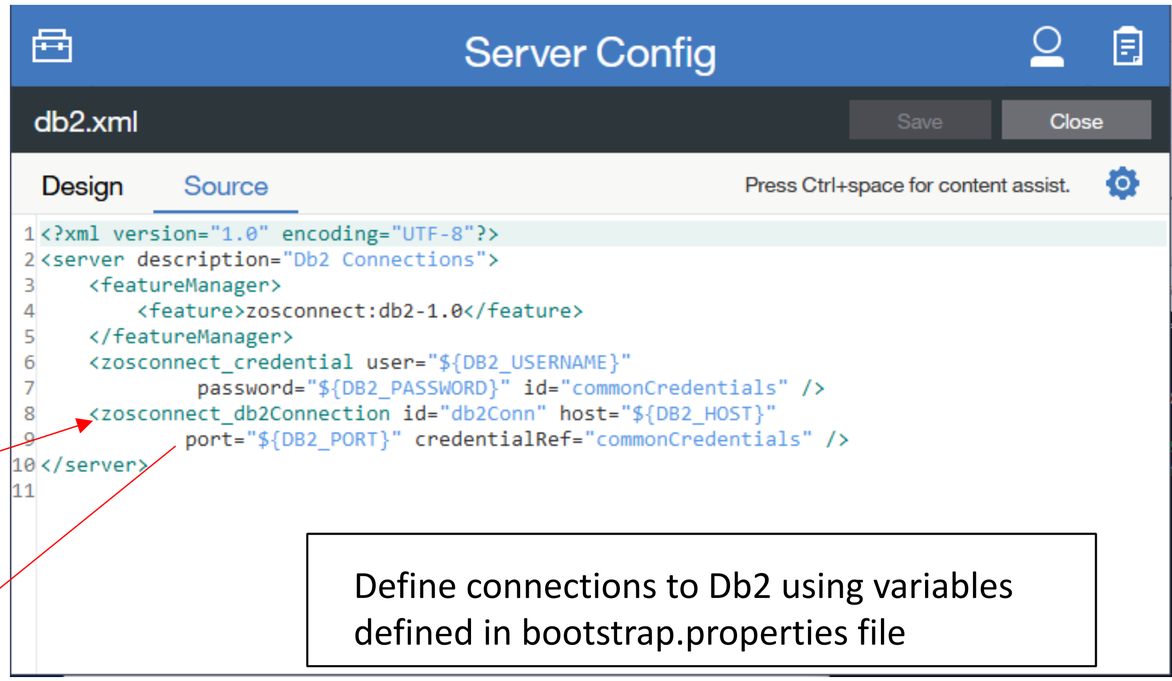


The connection references identifies a zosconnect_imsConnection element. Which provides the connection details to IMS.

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

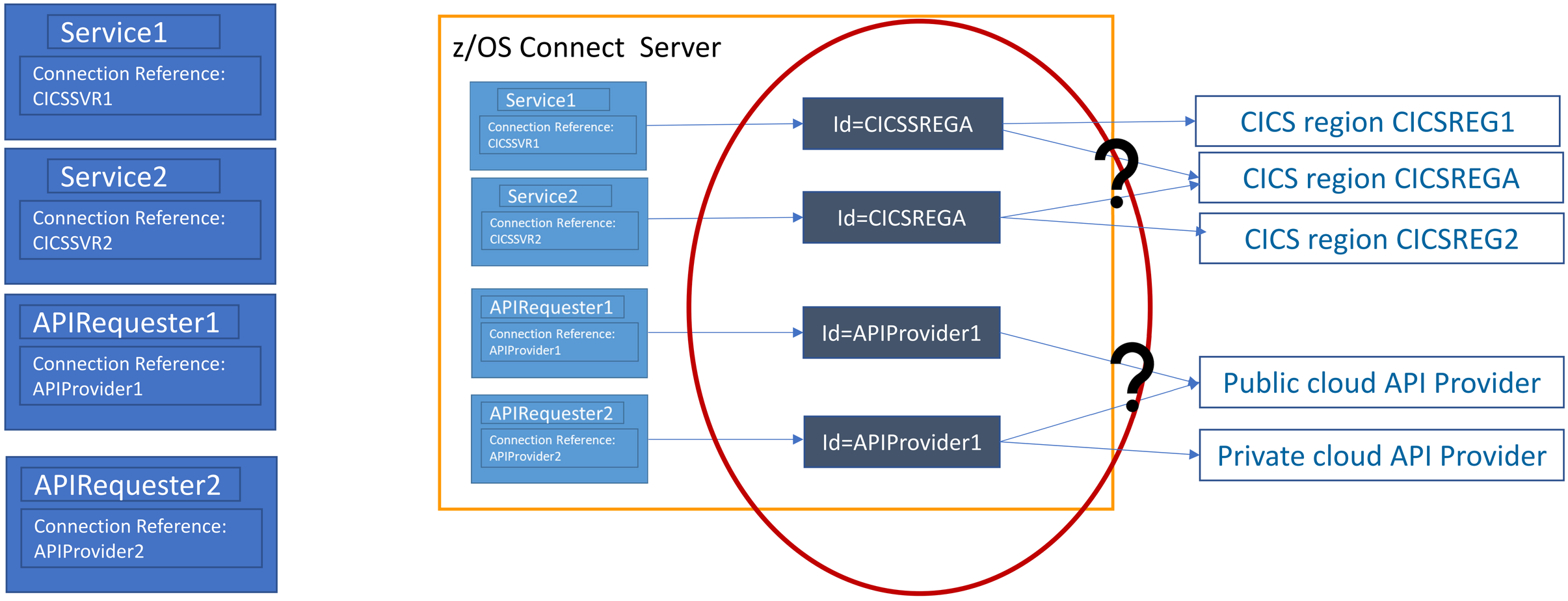
Connection Reference Consideration

Carefully consider the names used for connections

mitchj@us.ibm.com

Use naming conventions for connection references

Use application meaningful names or an extendable convention for connection reference names

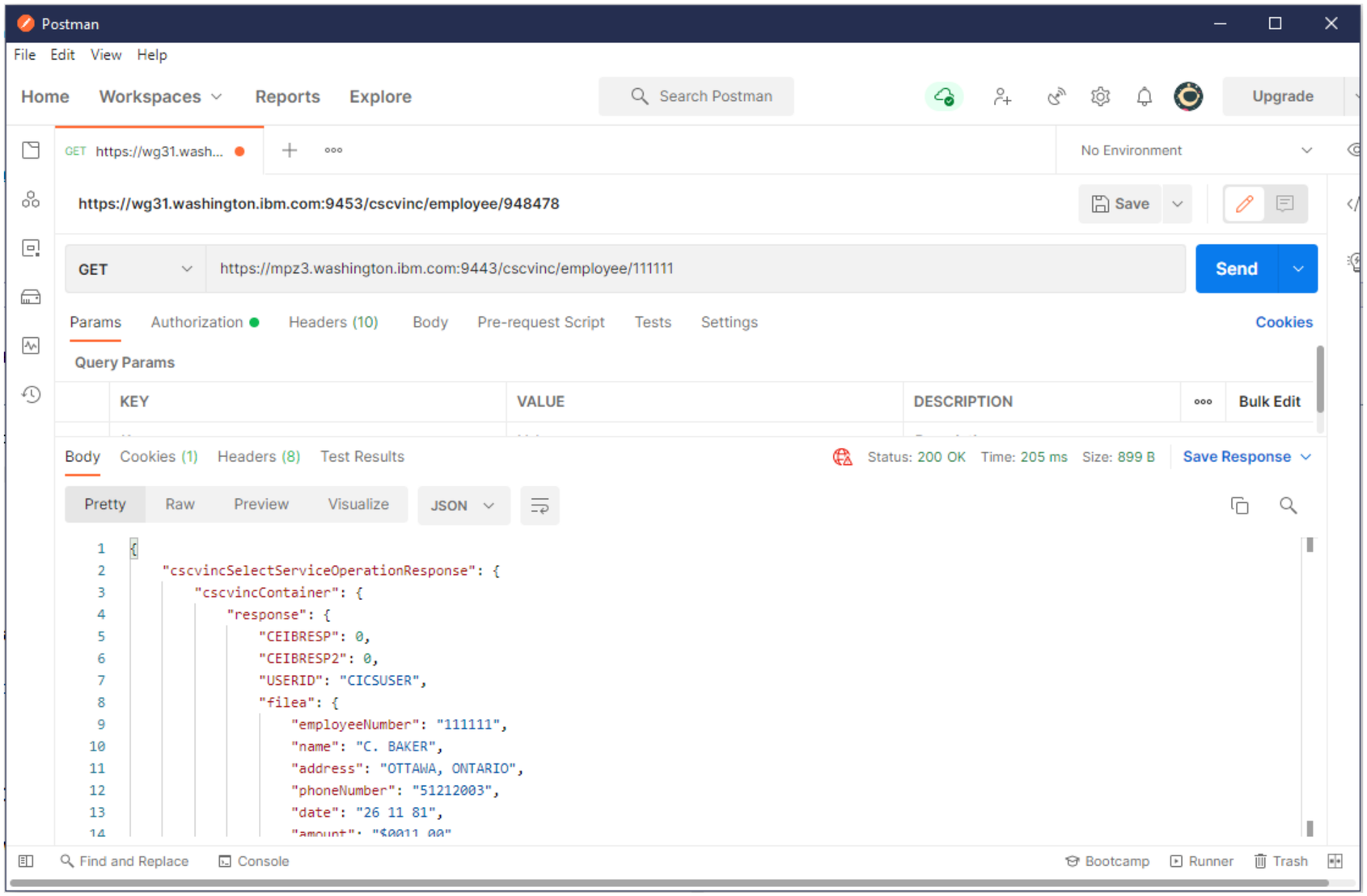


mitchj@us.ibm.com

What REST test tooling is available?

mitchj@us.ibm.com

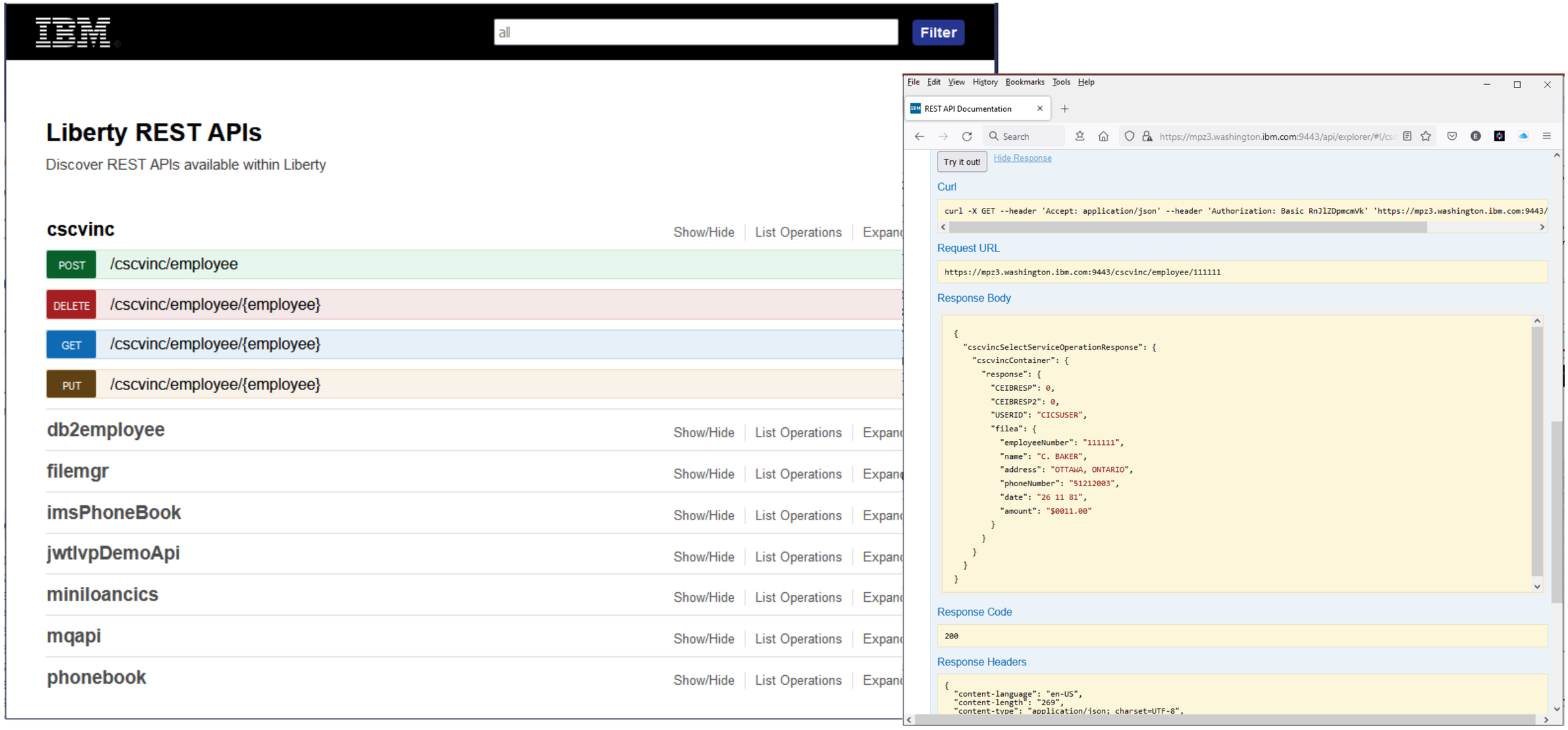
Testing an API with Postman



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

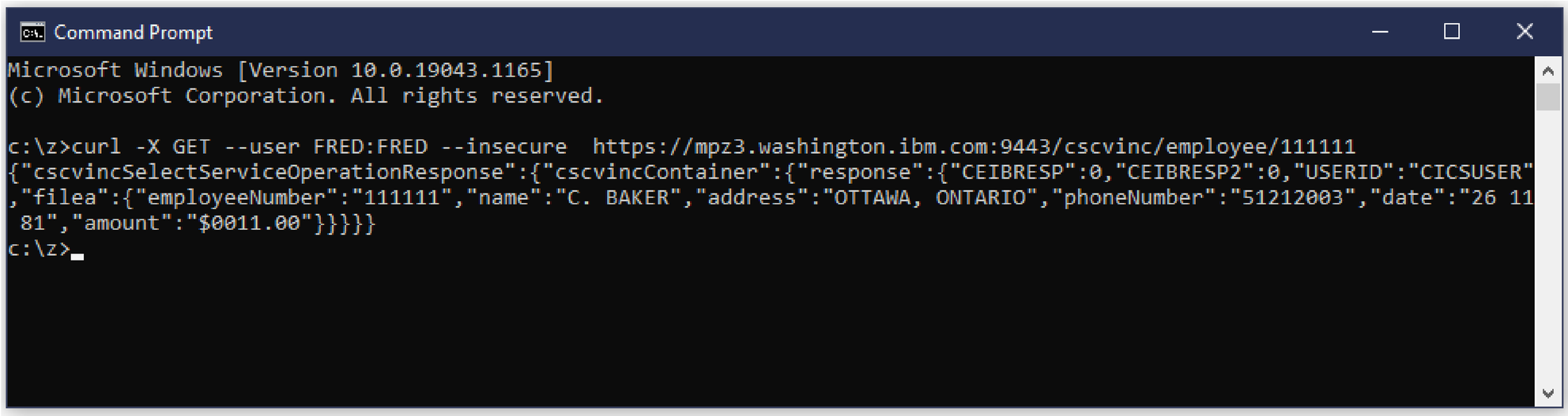
mitchj@us.ibm.com

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

Examples of curl in JCL and in scripts, bat or 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
curl -X GET -user user1:user1 --insecure
https://%TARGET%:9483/cscvinc/employee/000100
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

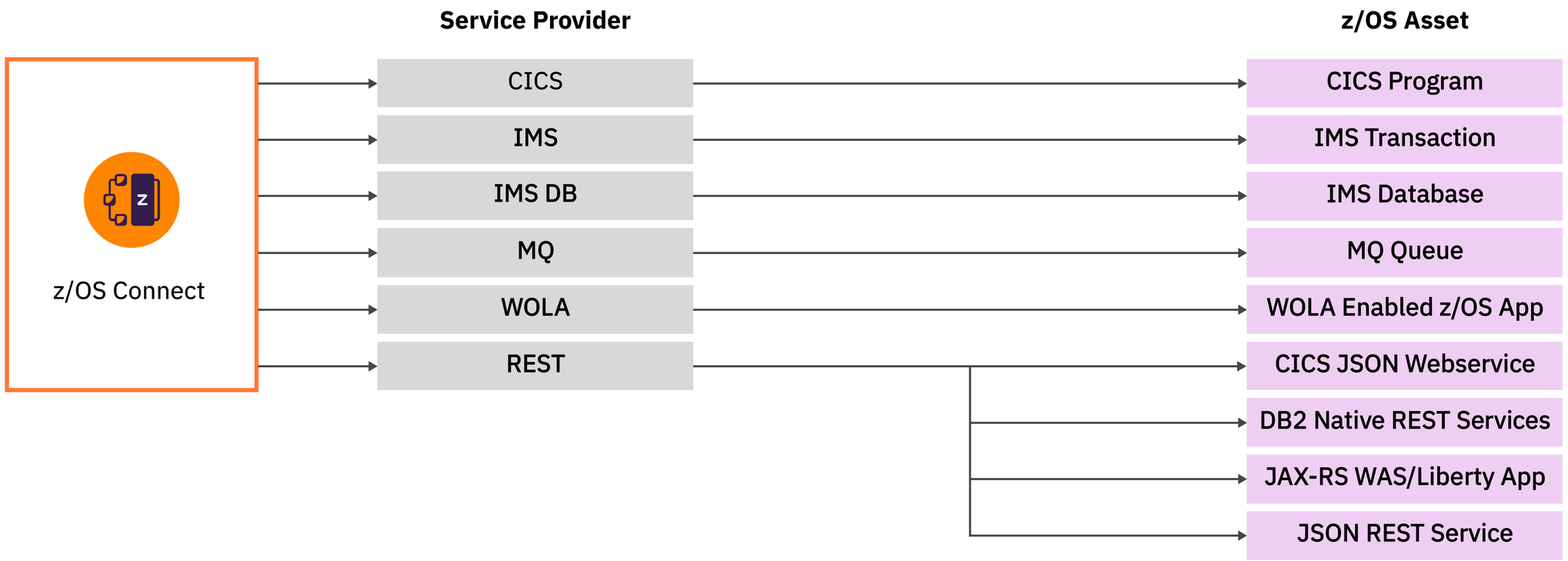
/miscellaneousTopics

performance, high availability, Liberty

mitchj@us.ibm.com

What assets can z/OS Connect EE map to?

And which service provider could I use?

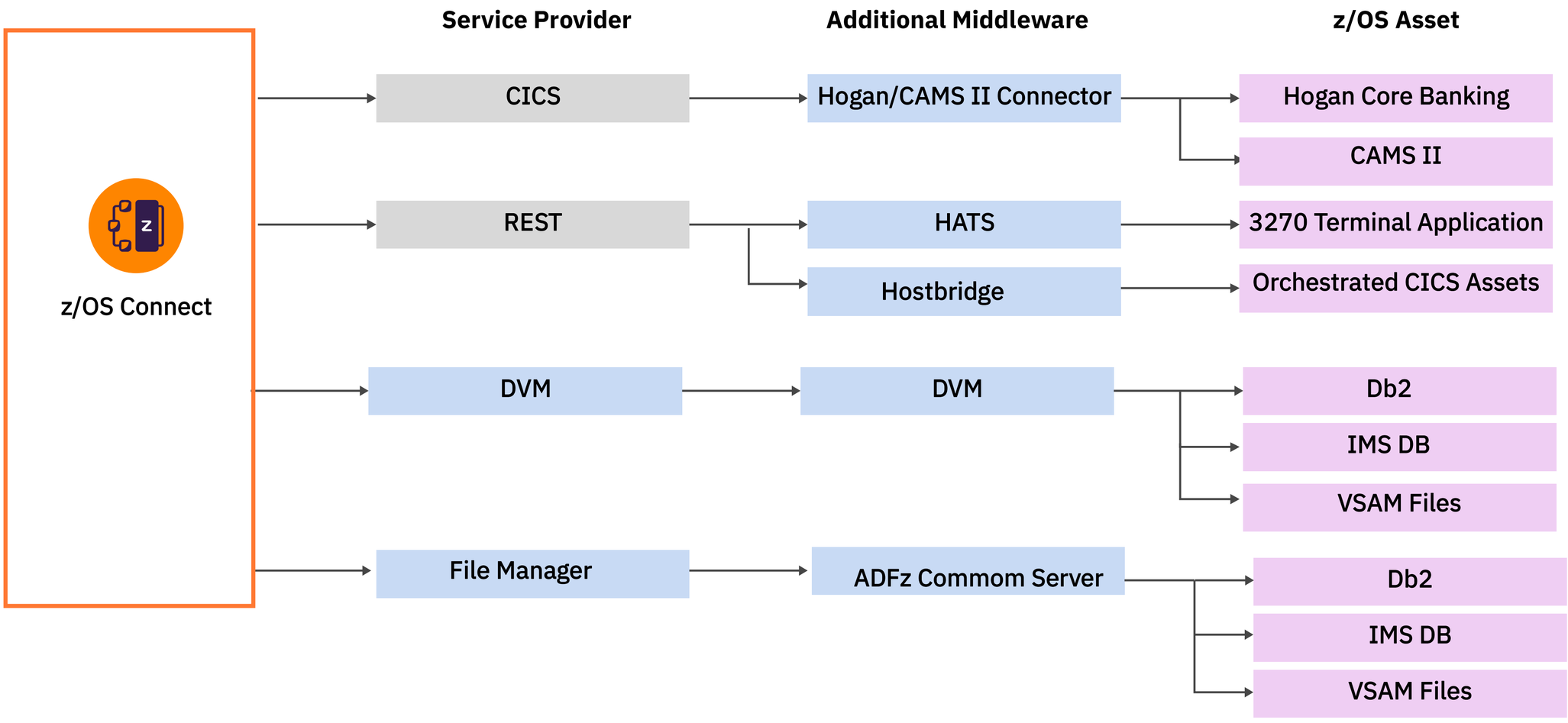


The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.

mitchj@us.ibm.com

Additional Middleware

Additional value from the ecosystem



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

mitchj@us.ibm.com

API Policies

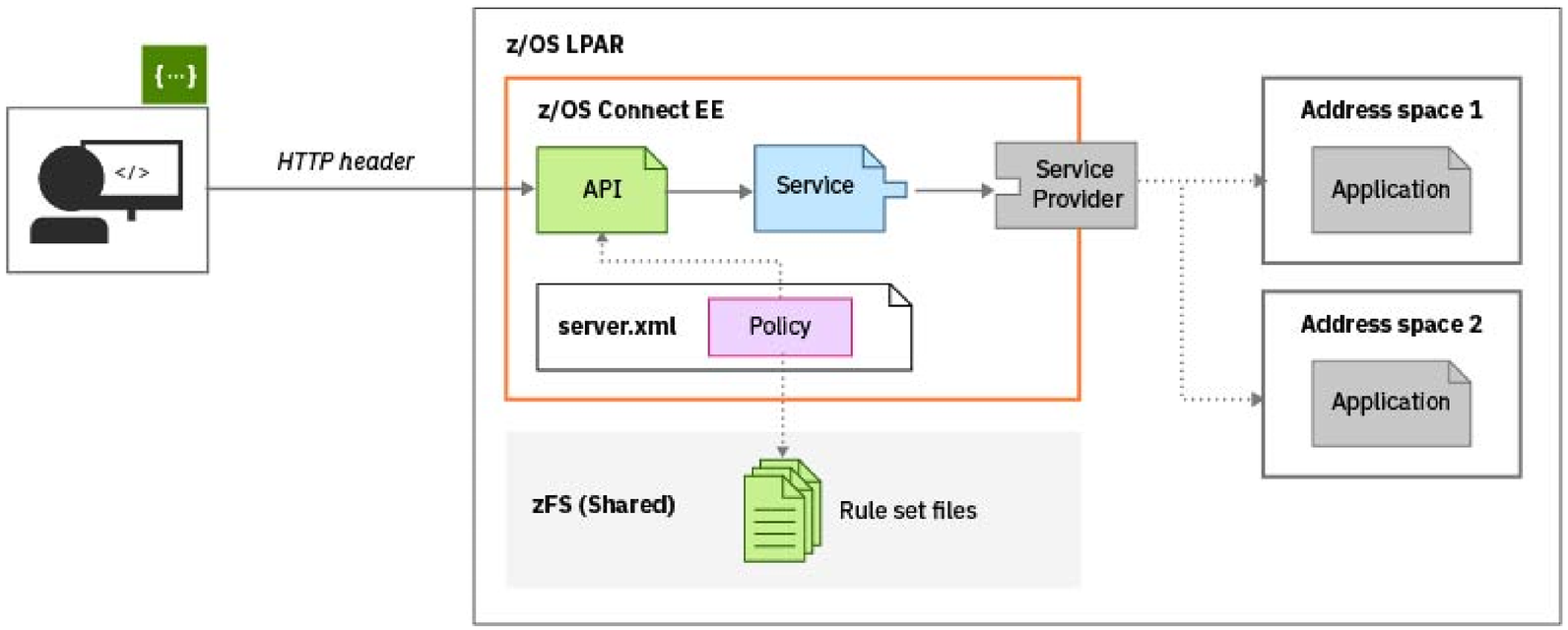
- HTTP header properties can be used to select alternative for IMS (V3.0.4) , CICS (V3.0.10), Db2 (V3.0.36) or MQ (V3.0.39)
- Policies can be configured globally for every API in the server or for individual APIs (V3.0.11)

CICS attributes
- cicsCcsid
- cicsConnectionRef
- cicsTransId

IMS attributes
- imsConnectionRef
- imsInteractionRef
- imsInteractionTimeout
- imsLtermOverrideName
- imsTranCode
- imsTranExpiration

Db2 attributes
- db2ConnectionRef
- db2CollectionID

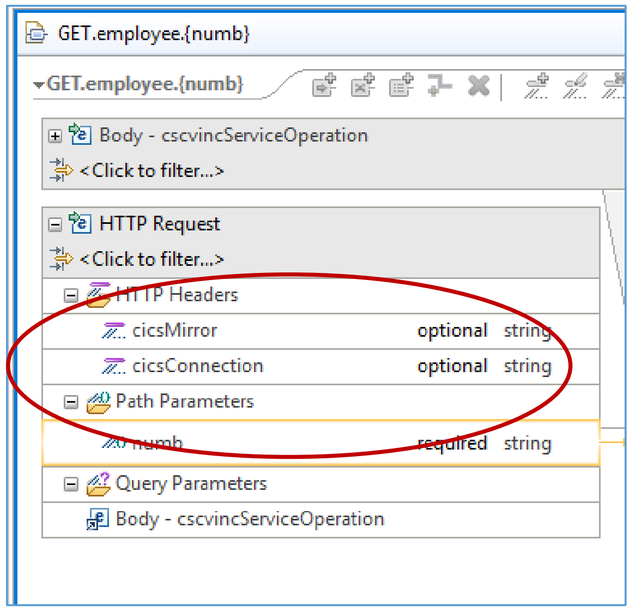
MQ attributes
- mqConnectionFactory
- mqDestination
- mqReplyDestination



mitchj@us.ibm.com

A sample API Policies for CICS

```
<ruleset name="CICS rules">
     <rule name="csmi-rule">
         <conditions>
             <header name="cicsMirror" match="ANY_VALUE"/> *
         </conditions>
         <actions>
             <set property="cicsTransId" value="${cicsMirror}"/>
         </actions>
     </rule>
     <rule name="connection-rule">
         <conditions>
             <header name="cicsConnection" value=" match="ANY_VALUE"/>
         </conditions>
         <actions>
             <set property="cicsConnectionRef" value="${cicsConnection}"
         </actions>
     </rule>
</ruleset>
```



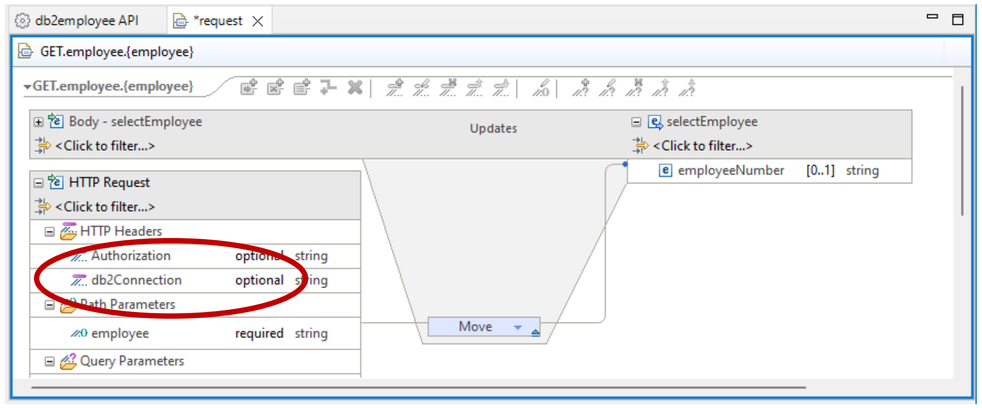
Curl

```
curl -X GET --header 'Accept: application/json' --header 'cicsMirror: MIJO' --header 'cicsConnection: cscvinc' 'https://m
```

*Transaction MIJO needs to be a clone of CSMI (e.g., invoke program DFHMIRS)

A sample API Policies for Db2

```
<ruleset name="Db2 rules">
     <rule name="connection-rule">
       <conditions>
           <header name="db2Connection" value="" match="ANY_VALUE"/>
       </conditions>
       <actions>
           <set property="db2ConnectionRef" value="${db2Connection}"/>
       </actions>
     </rule>
 </ruleset>
```

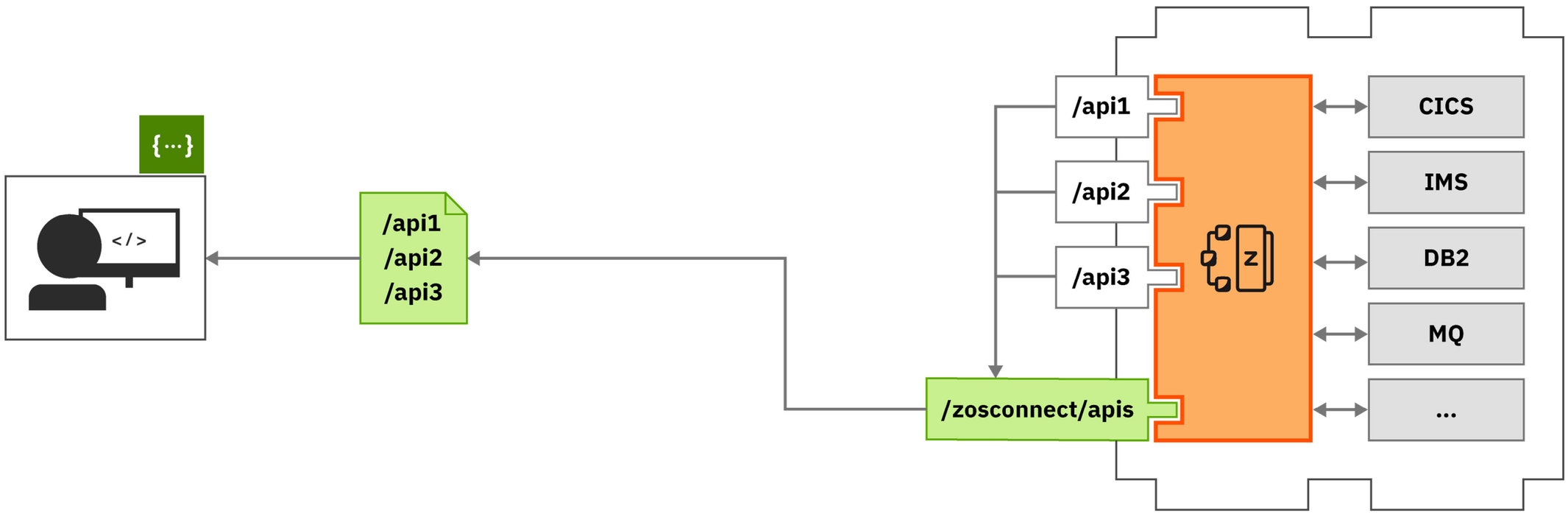



```
<zosconnect_zosConnectServiceRestClientConnection id="Db2Conn"
       sslCertsRef="db2SSLSettings" host="wg31.washington.ibm.com" port="2445" basicAuthRef="dsn2Auth" />
<zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth" applName="DSN2APPL"/>
<ssl id="db2SSLSettings"  keyStoreRef="Db2KeyStore"  trustStoreRef="Db2KeyStore"/>

<zosconnect_zosConnectServiceRestClientConnection id="fred"
       sslCertsRef="fredSSLSettings"  host="wg31.washington.ibm.com" port="2445" />
<ssl id="fredSSLSettings" keyStoreRef="Db2KeyStore"  trustStoreRef="Db2KeyStore" clientKeyAlias="FRED"/>

<zosconnect_zosConnectServiceRestClientConnection id="user1"
       sslCertsRef="user1SSLSettings"  host="wg31.washington.ibm.com"  port="2445" />
<ssl id="user1SSLSettings" keyStoreRef="Db2KeyStore"  trustStoreRef="Db2KeyStore" clientKeyAlias="USER1"/>

<keyStore id="Db2KeyStore" location="safkeyring:///zCEE.Db2.KeyRing"
       password="password" type="JCERACFKS" fileBased="false" readOnly="true" />
```

Requires APAR PH53162

mitchj@us.ibm.com

API Documentation



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

mitchj@us.ibm.com

/security

How is security implement?

mitchj@us.ibm.com

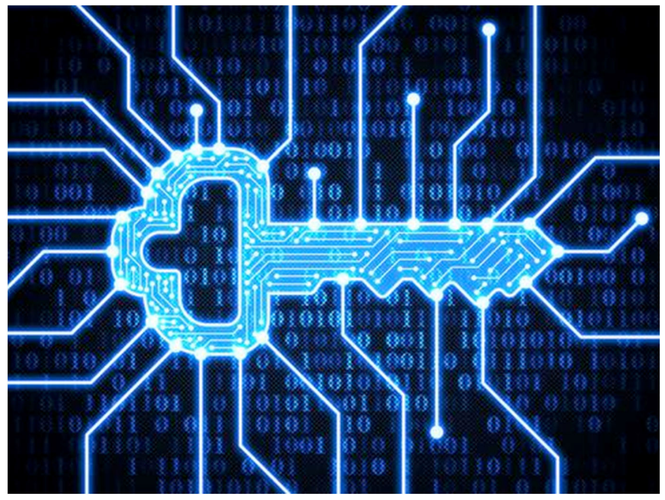
General security terms or considerations

Security involves

- Identifying who or what is requesting access (**Authentication)**
  - Basic Authentication
  - Mutual Authentication using Transport Layer Security (TLS), formerly known as SSL
  - Third Party Tokens

- Ensuring that the message has not been altered in transit (**Data Integrity)** and ensuring the confidentiality of the message in transit (**Encryption)**
  - TLS (encrypting messages and using a digital signature)

- Controlling access (**Authorization)**
  - Is the authenticated identity authorized to access to z/OS Connect
  - Is the authenticated identity authorized to access a specific API, Services, etc.



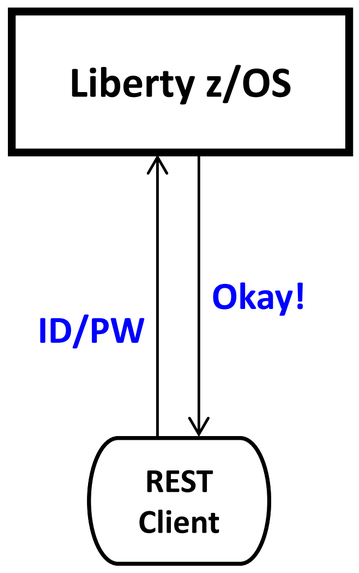
mitchj@us.ibm.com

Slide 126

API Provider Authentication

Several different ways this can be accomplished:

Basic Authentication

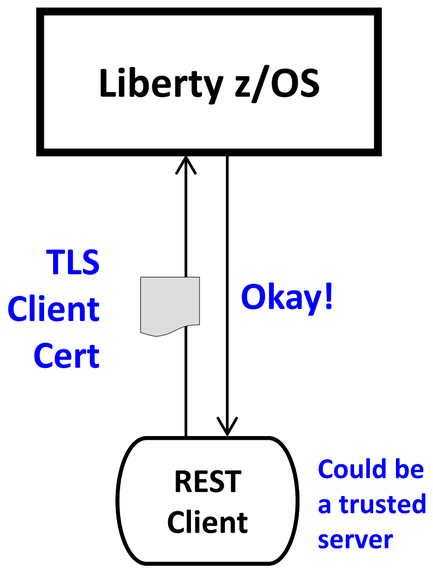


Server prompts for ID/PW

Client supplies ID/PW or ID/Passticket

Server checks registry:
- **Basic (server.xml)**
- **LDAP**
- **SAF**

Client Certificate



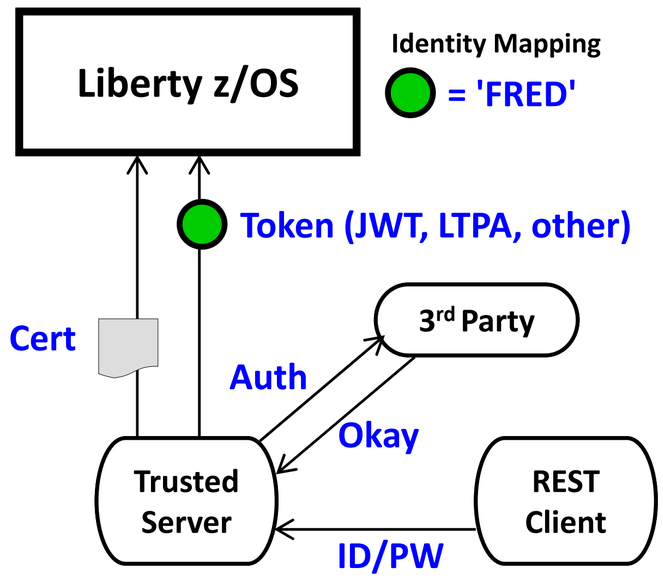
Server prompts for cert.

Client supplies certificate

Server validates cert and maps to an identity

Registry options:
- **LDAP**
- **SAF**

Third Party Authentication



Client authenticates to 3rd party sever

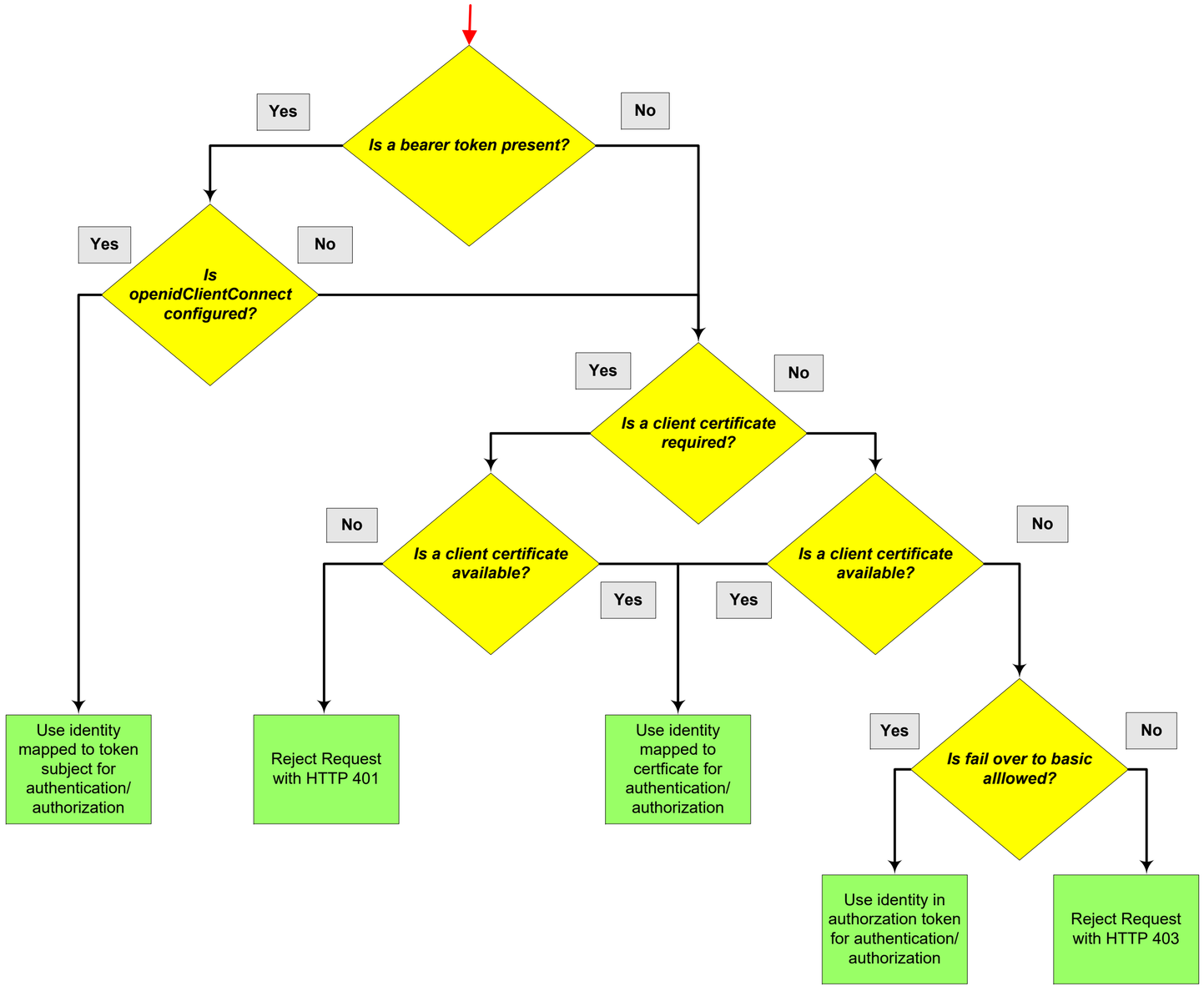
Client receives a trusted 3rd party token

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

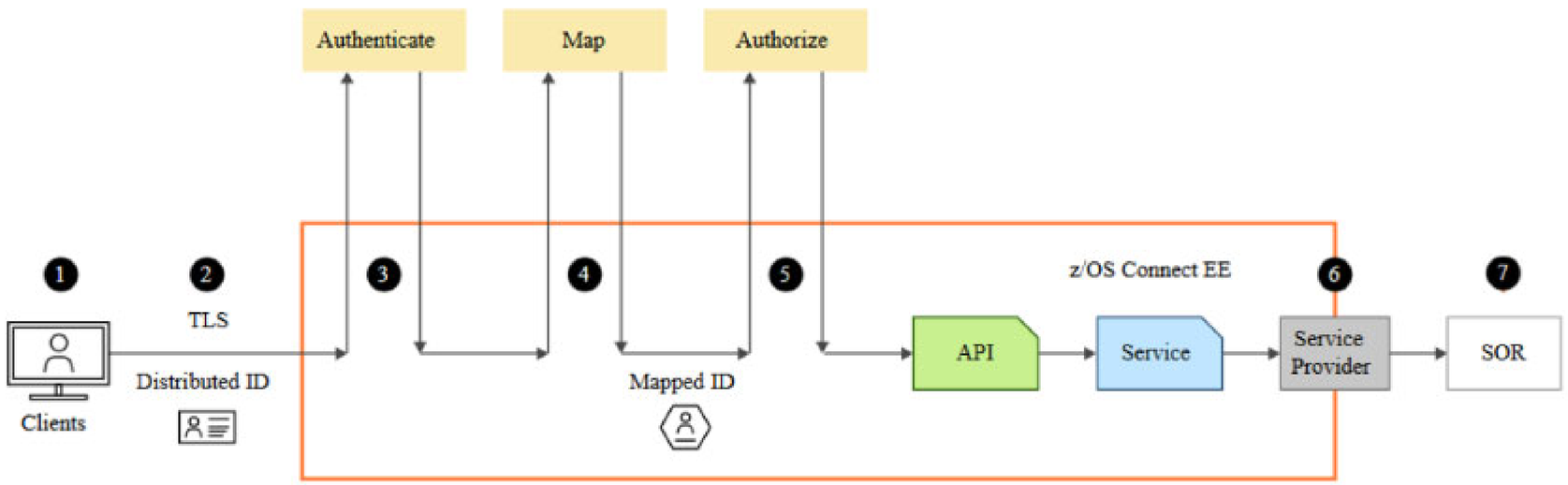
Registry options:
- **LDAP**
- **SAF**

mitchj@us.ibm.com

Authentication credential precedence order for determining authorization identity



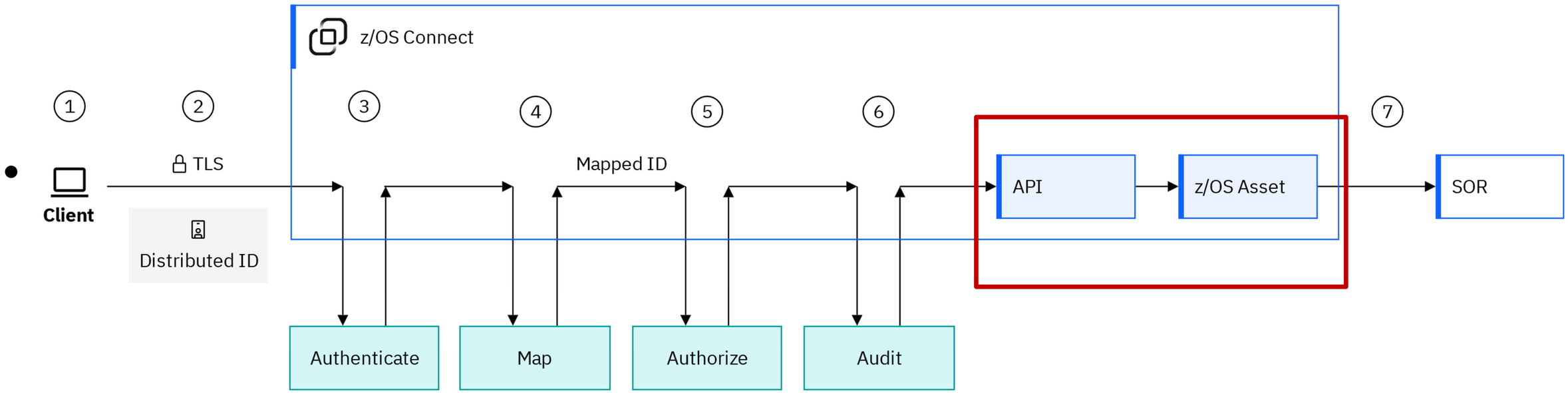
Typical z/OS Connect EE API Provider security flow



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

Details of a typical z/OS Connect EE API Provider security flow - inbound (OpenAPI 3)



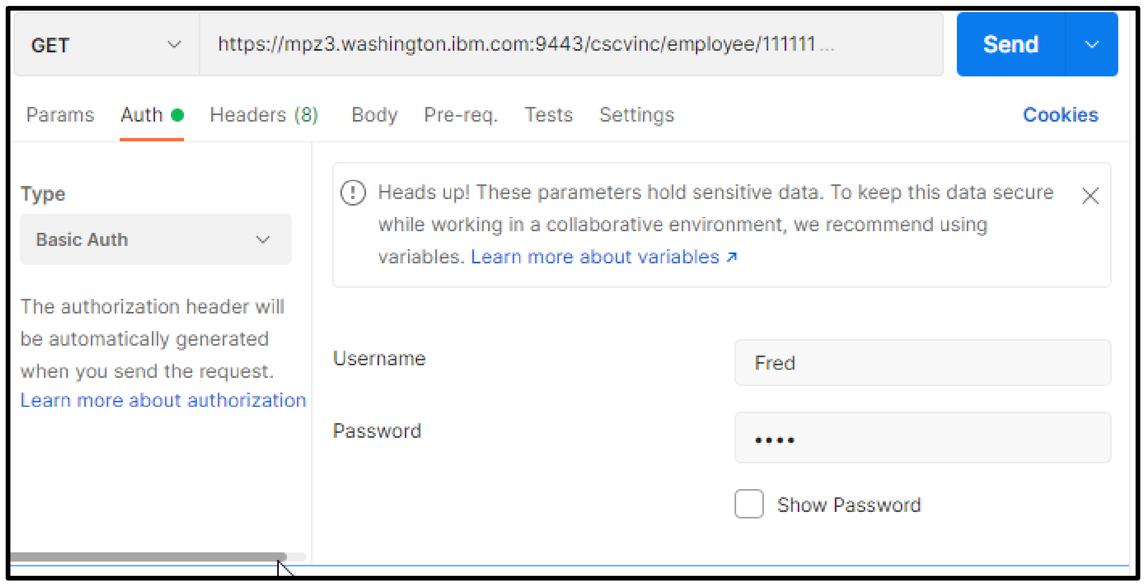
The flow includes the following security steps that can be performed by IBM z/OS Connect. The credentials are provided by the client. These can be a user ID and password, a JWT, or a TLS certificate.

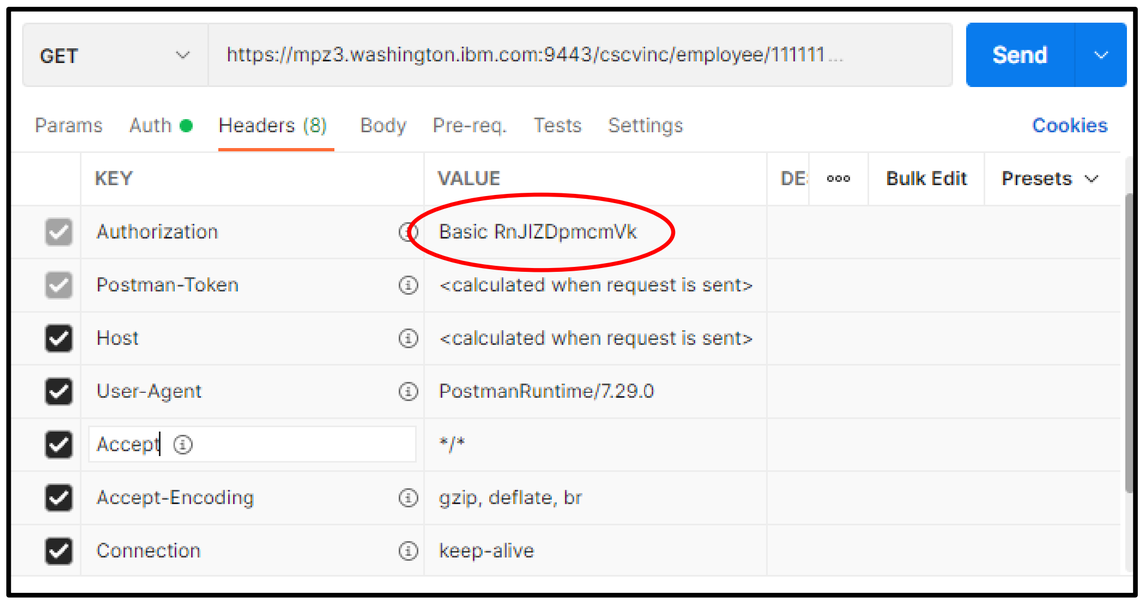
1. The credentials, including the identity, are passed on the connection between the client and IBM z/OS Connect. The identity is typically a distributed ID, such as an X.509 distinguished name and associated LDAP realm that originates from a remote system. Alternatively, the identity might be a SAF user ID. The data that is sent on the connection can be encrypted using TLS.
2. The client is authenticated. This can be within IBM z/OS Connect or by requesting verification from a third-party server.
3. The authenticated identity can be mapped to a user ID in the IBM z/OS Connect user registry.
4. The user is authorized to invoke the API operation if they have the required role.
5. The API request is audited by using the Liberty Audit feature.
6. The authenticated user identity can be propagated to the System of Record (SoR) when the SoR supports this capability. Alternatively, the SoR connection can be configured to use a functional identity.

mitchj@us.ibm.com

Basic authentication – Where the client provides an identity and password

- When sending a request to a Liberty server, basic authentication information (identity and password) is provided in the HTTP header in a Basic *Authorization t*oken with the identity and password encoded or formatted using Base64.
  - An example with Postman:

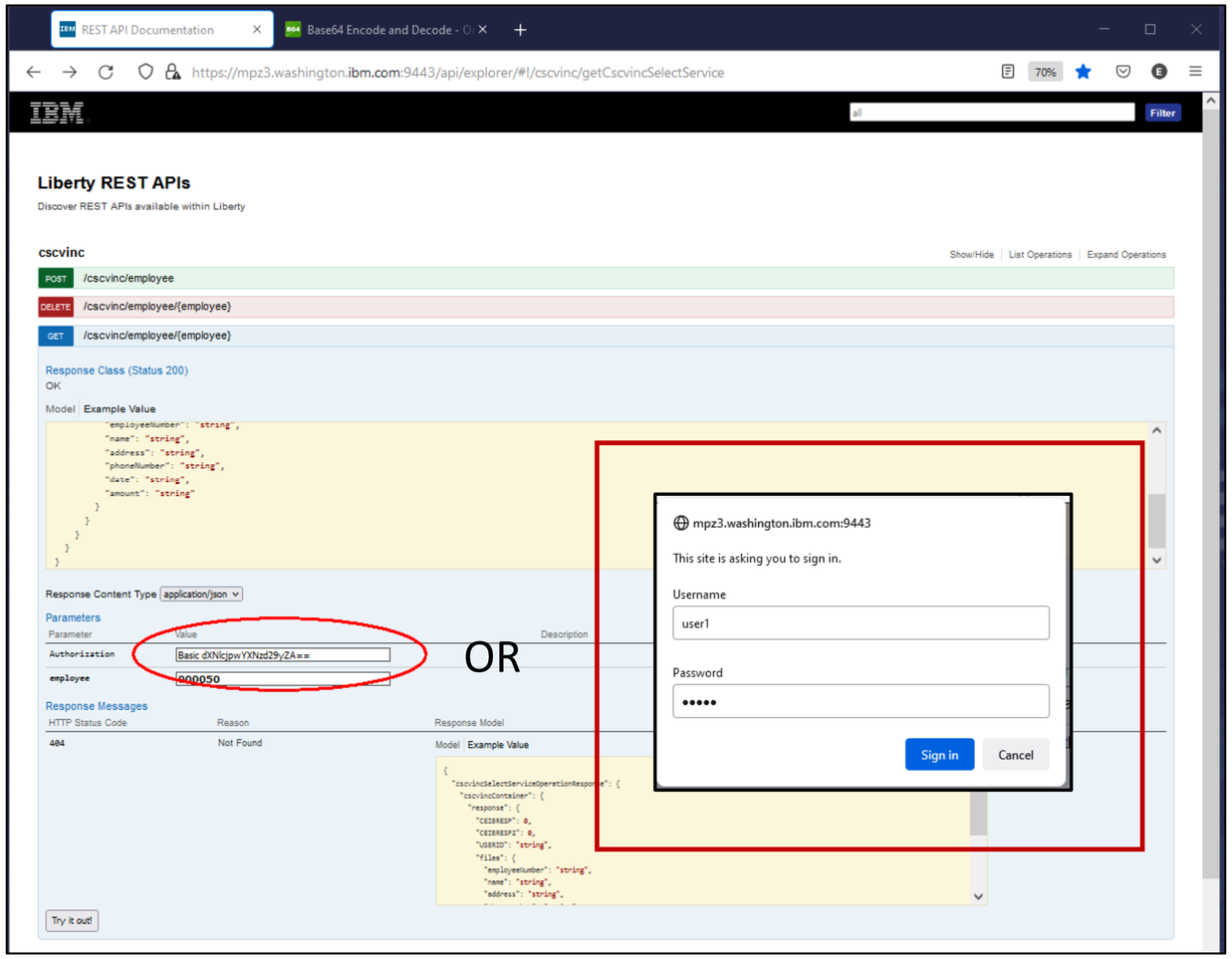


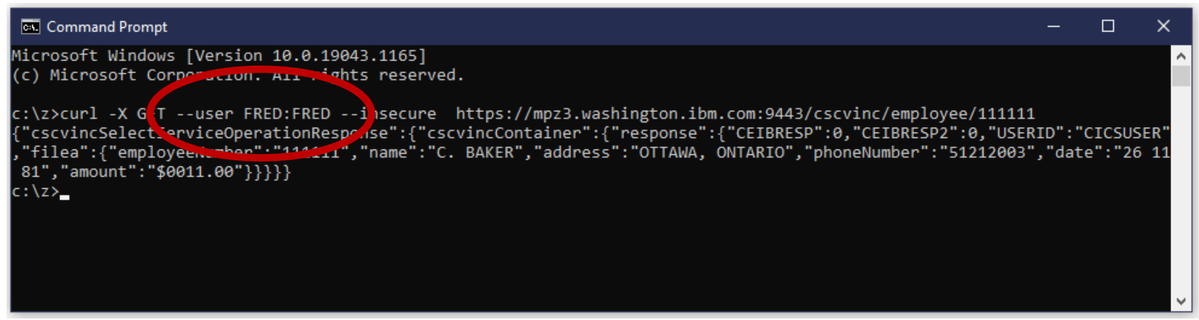


mitchj@us.ibm.com

There are multiple ways to provide an identity and password

- When sending a request to a Liberty server running z/OS Connect, basic authentication information (identity and password) is provided in the HTTP header in a Basic *Authorization t*oken with the identity and password encoded or formatted using Base64.
  - Examples using the API Explorer feature , cURL, and a Java client.





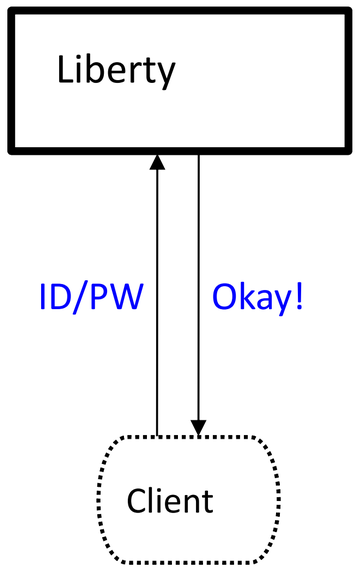
https://www.base64encode.org/

mitchj@us.ibm.com

Authentication - TLS Mutual Authentication

Several different ways this can be accomplished:

Basic Authentication

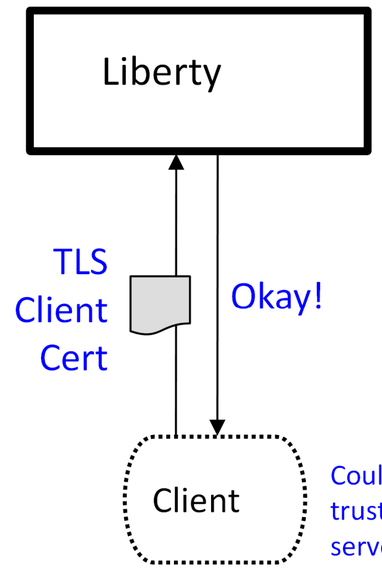


Server prompts for ID/PW

Client supplies ID/PW or ID/PassTicket

Server checks registry:
- Basic (server.xml)
- SAF

Client Certificate



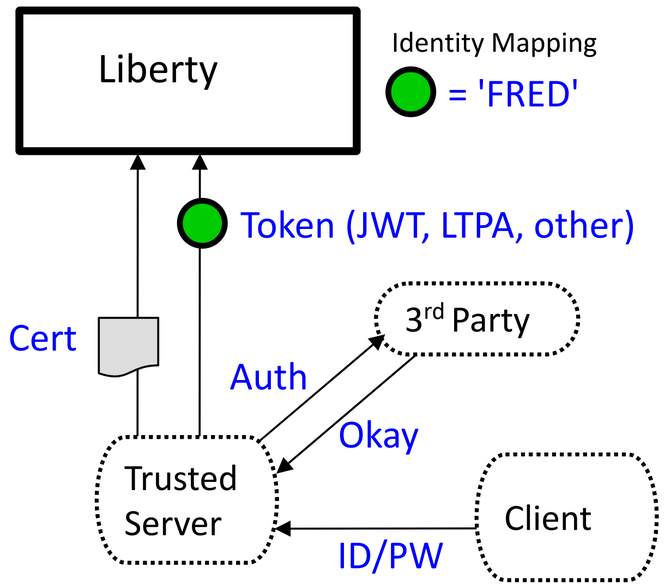
Server prompts for client certificate.

Client supplies personal certificate

Server validates client certificate and maps it to an identity

Registry options:
- **SAF**

Third Party Authentication



Client authenticates to 3rd party sever

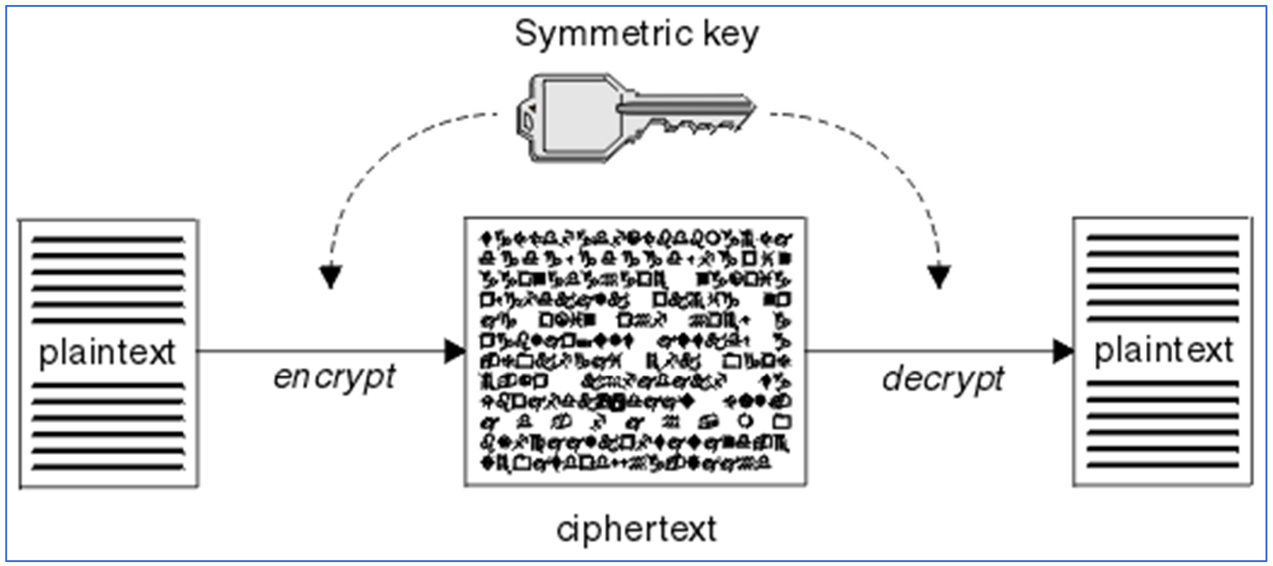
Client receives a trusted 3rd party token

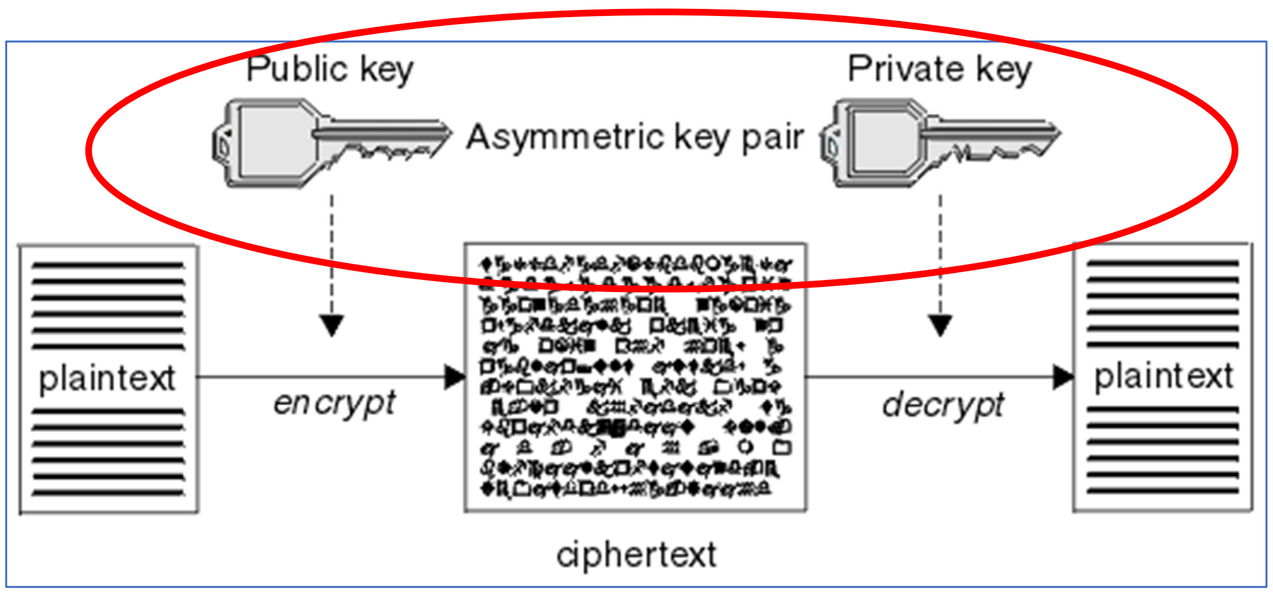
Token flows to Liberty z/OS and is mapped to an identity

Registry options:
- We may not need to know these details.

mitchj@us.ibm.com

Tech-Tip: Symmetric key v. Asymmetric key pairs





A symmetric key is a key shared by the endpoints. Both endpoints use the same key to encrypt and decrypt messages.

An asymmetric key pair is the preferred solution. There is no risk of compromise by sending a symmetric or shared key outside of a protected communication flow.

A message encrypted with a public key can only be decrypted by endpoint that has the private key. The privacy of the messages is ensured.

If an endpoint can successfully decrypt a message message encrypted received with a private key, the endpoint sending the message has successfully asserted its validity by proving it has the private used to encrypt the message.

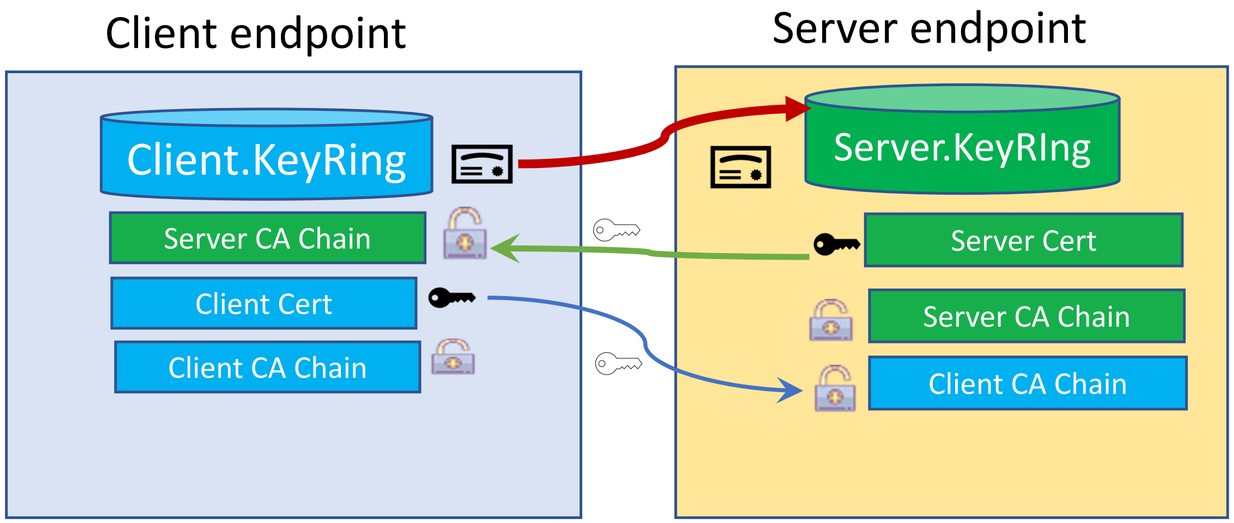
mitchj@us.ibm.com

The basic TLS Handshake Flow (HTTPS)

The HTTPS protocol involves a TLS handshake –

Server Authentication (always occurs when HTTPS is the protocol)

Mutual Authentication (optional, at the request of the server endpoint)



Certificate with a private key*

Certificate and public key

Certificate Authority (CA) certificate chain#

*For server and/or mutual authentication to work, the endpoint sending the client certificate must use a personal certificate with a private key. The private key is required to decrypt (or encrypt) a message digest that is sent from the other endpoint during the handshake flow. Generation of a message digest also requires access to the CA certificate used to sign the certificate.

#Refers to the set or of certificates used to issue the server or client personal certificate including any intermediate certificates up to and including the root CA.

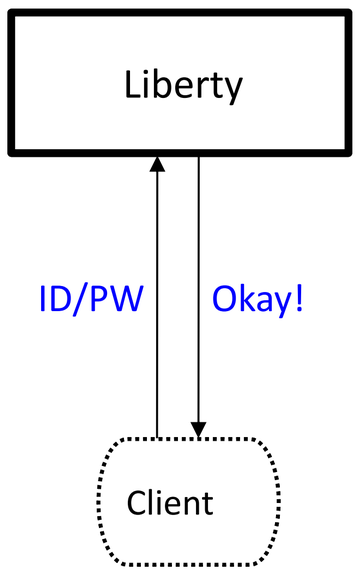
mitchj@us.ibm.com

Slide 135

Authentication - Third Party Authentication

Several different ways this can be accomplished:

Basic Authentication

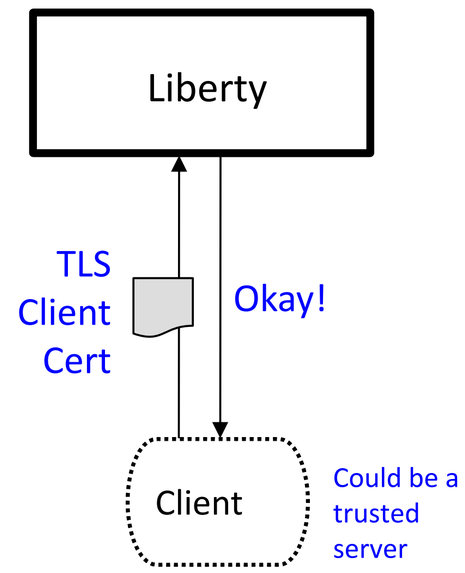


Server prompts for ID/PW

Client supplies ID/PW or ID/PassTicket

Server checks registry:
- Basic (server.xml)
- SAF

Client Certificate



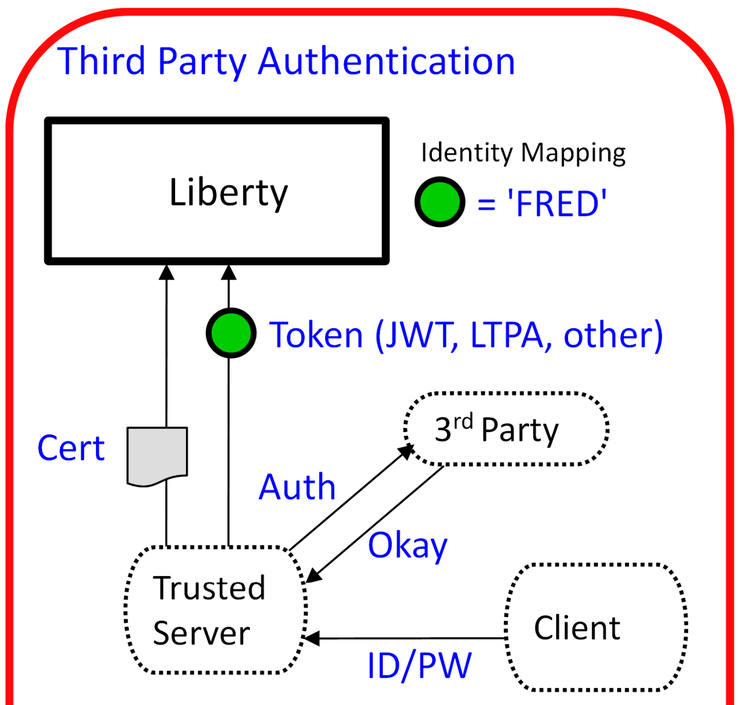
Server prompts for client certificate.

Client supplies certificate

Server validates client certificate and maps to an identity

Registry options:
- SAF

Third Party Authentication



Client authenticates to 3rd party sever

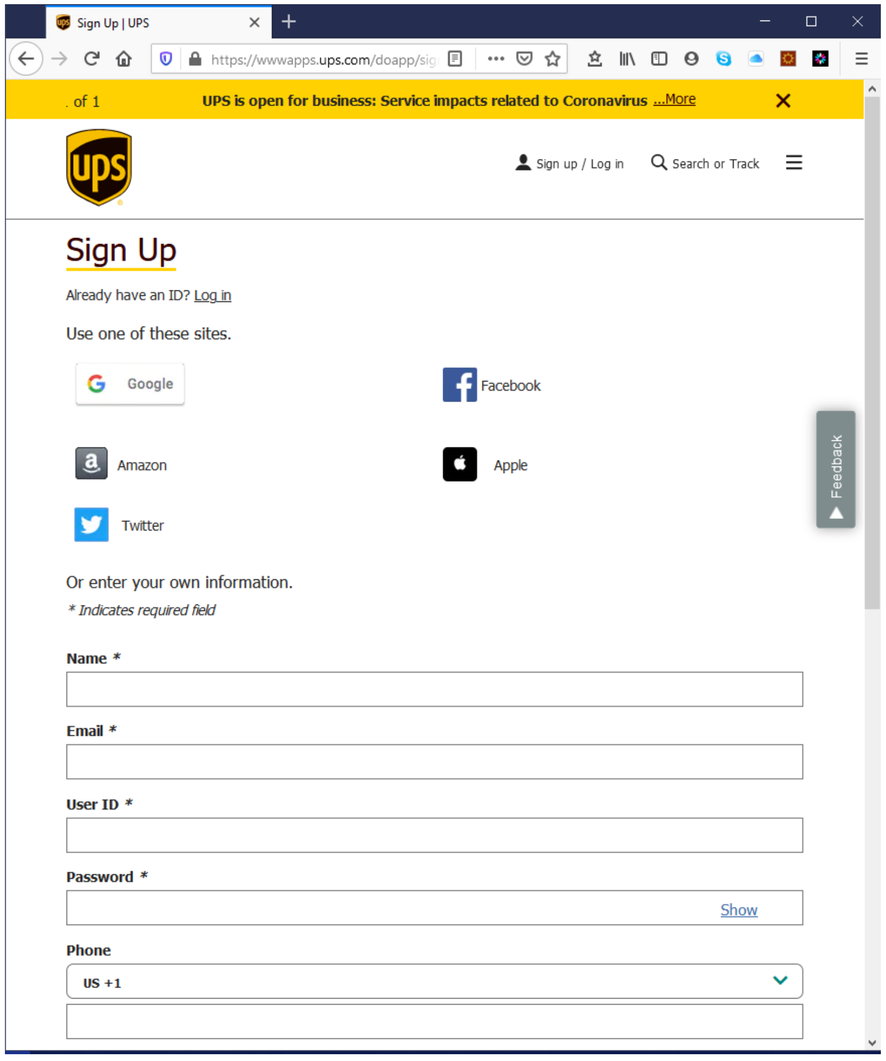
Client receives a trusted 3rd party token

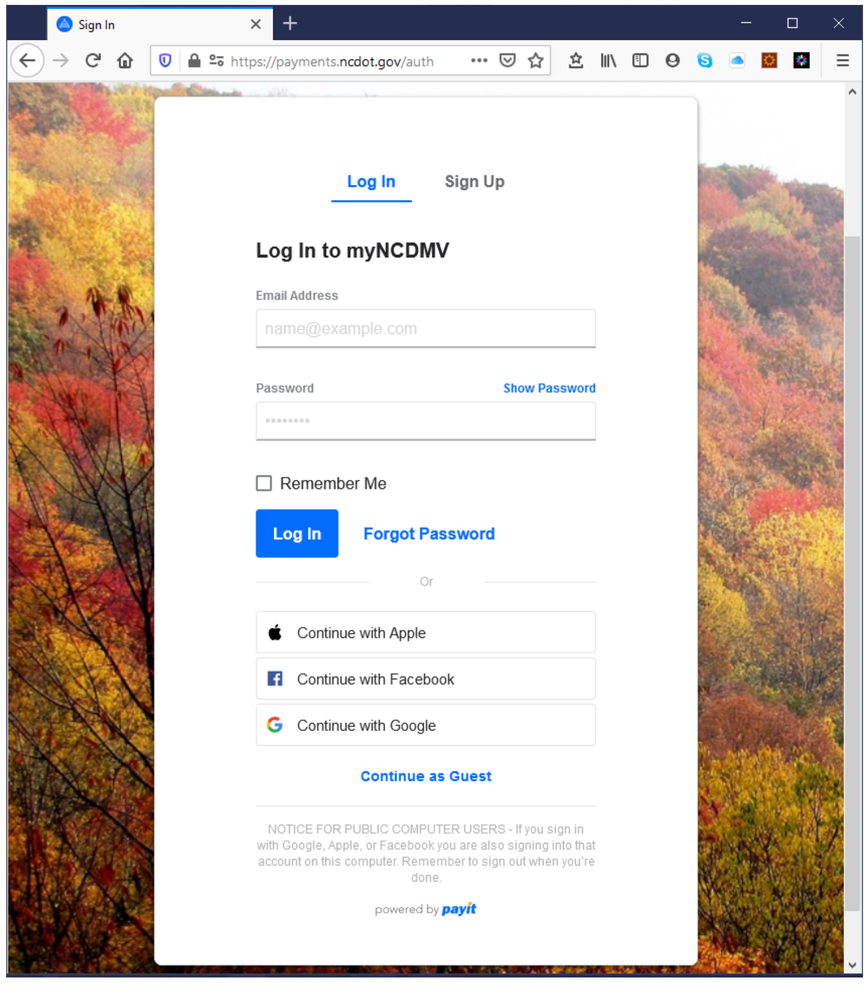
Token flows to Liberty z/OS and is mapped to an identity

Registry options:
- **We may know these detail.**

mitchj@us.ibm.com

Third Party Authentication Examples





mitchj@us.ibm.com

Open security standards

- **OAuth** is an open standard for access delegation, used as a way to grant websites or applications access to their information without requiring a password.
- **OpenID Connect** is an authentication layer on top of Oauth. It allows the verification of the identity of an end-user based on authentication performed by an authorization server.
- **JWT** (JSON Web token) defines a compact and self-contained way for securely transmitting information between parties as a JSON object

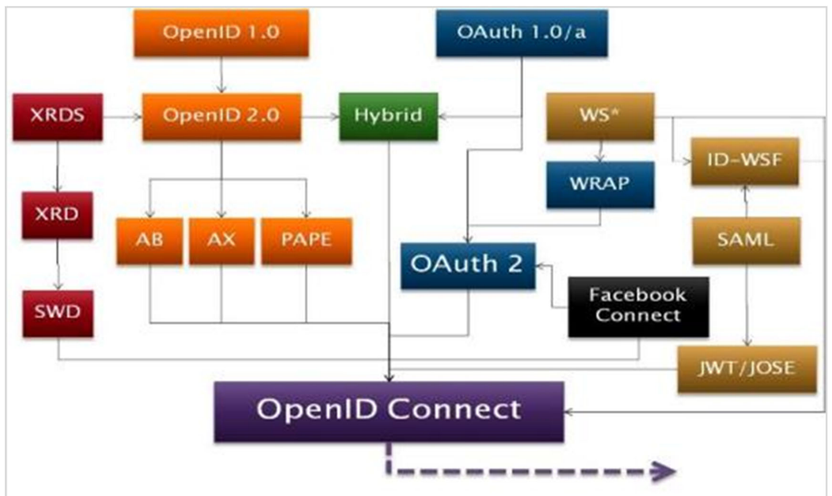
See the YouTube videos:

OAuth 2.0 and OpenID Connect (in plain English)

https://www.youtube.com/watch?v=996OiexHze0

OpenID Connect on Liberty

https://www.youtube.com/watch?v=fuajCS5bG4c



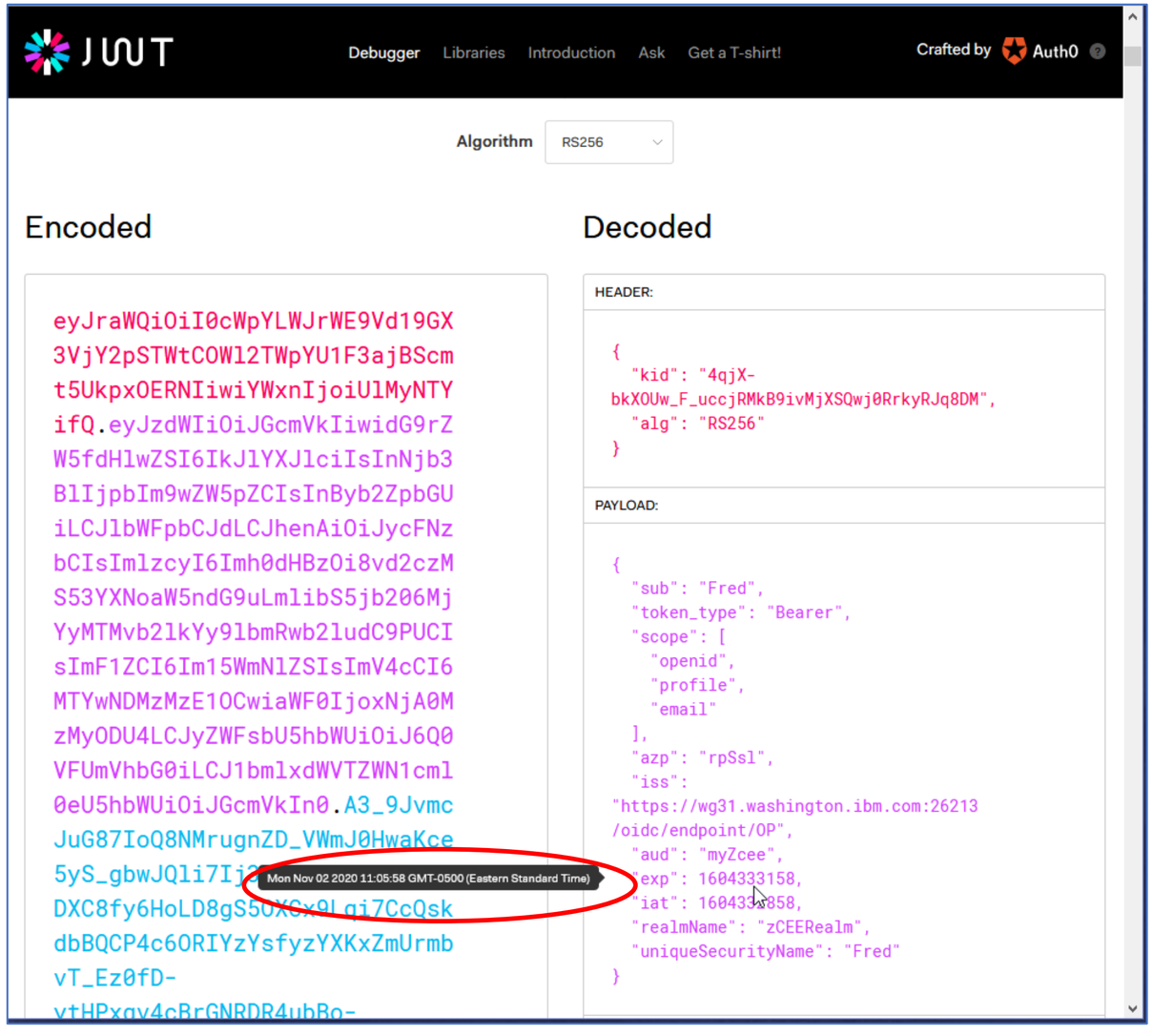
mitchj@us.ibm.com

What is a JWT (JSON Web Token) ?

- JWT is a compact way of representing claims that are to be transferred between two parties

- Normally transmitted via HTTP header

- Consists of three parts
  - Header
  - Payload
  - Signature



Values derived from the OAUTH configuration:

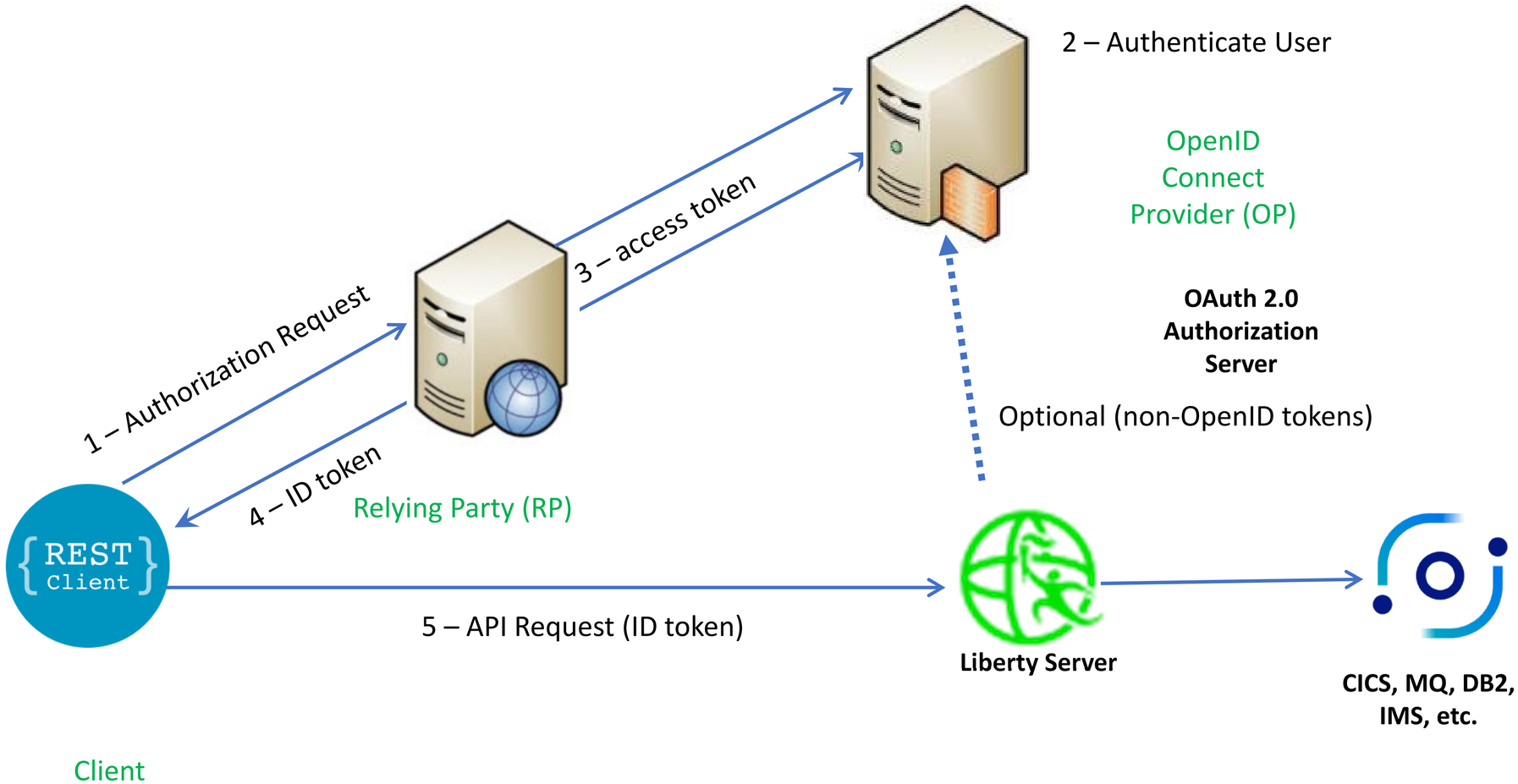
- signatureAlgorithm="**RS256**“
- accessTokenLifetime="**300**"
- resourceIds="**myZcee**"

https://jwt.io

mitchj@us.ibm.com

Slide 139

Typical Authorization Flow for am OpenID Connect token to a z/OS Connect API Provider



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

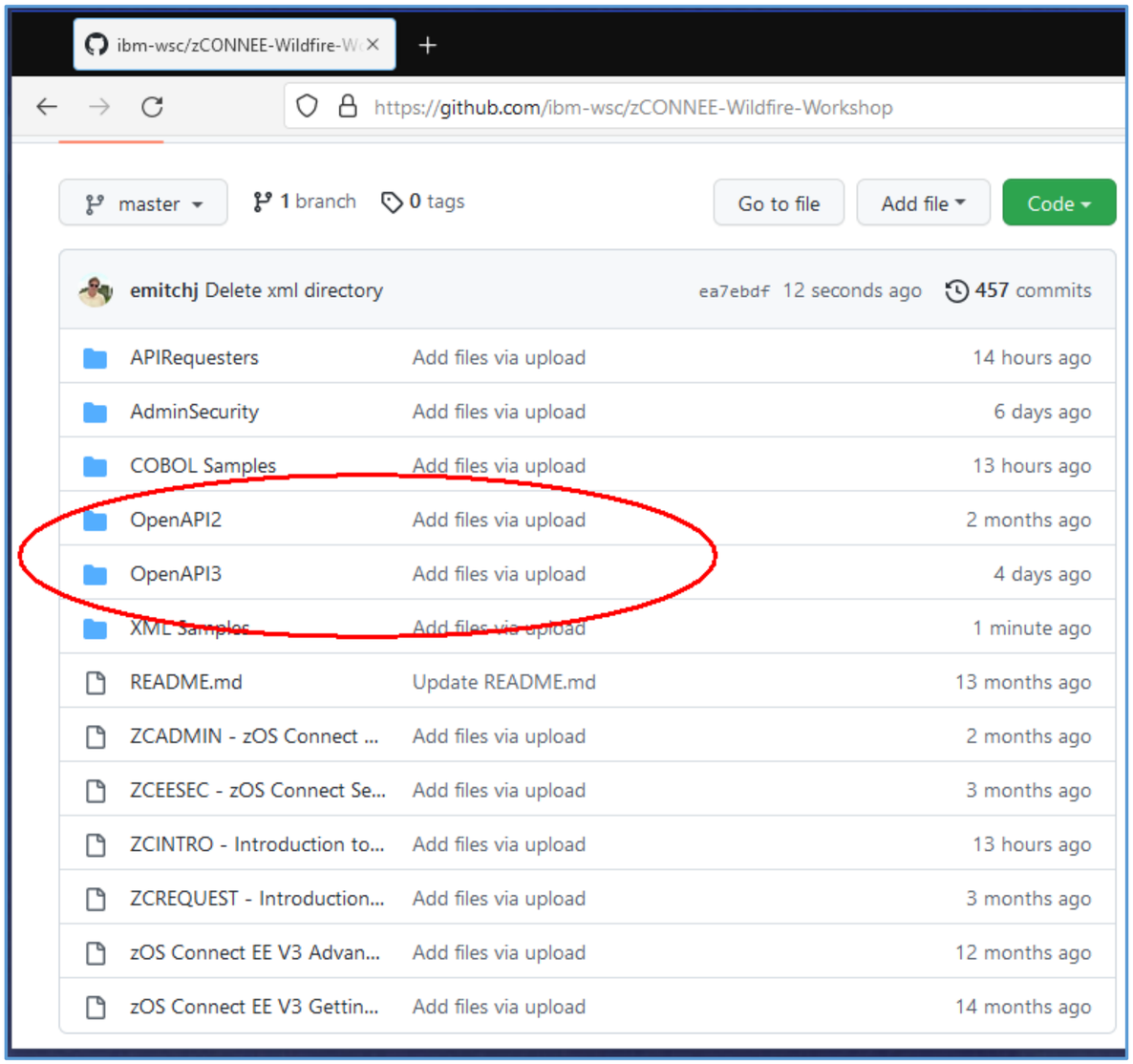
- A brief overview of z/OS Connect Security*

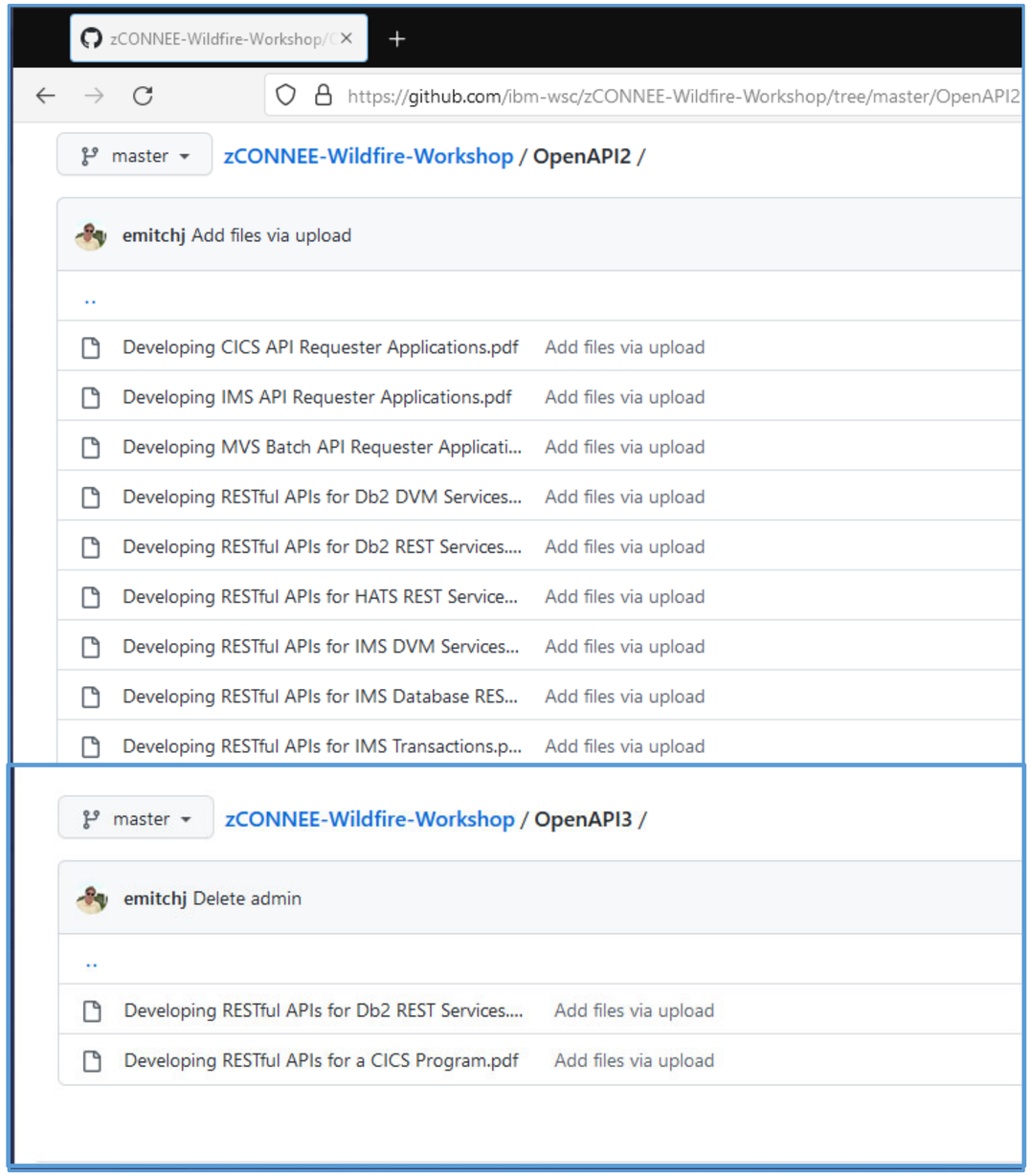
*For more on security, contact your local IBM rep regarding the schedule of workshop *zOSSEC1 IBM z/OS Connect Administration/Security Wildfire Workshop*

mitchj@us.ibm.com

z/OS Connect Wildfire Github Site

https://ibm.biz/zCEEWorkshopMaterial





mitchj@us.ibm.com

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

Thank you for listening and your questions

mitchj@us.ibm.com