IBM z/OS Connect Enterprise Edition

Introduction and Overview

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

IBM Z
Wildfire Team –
Washington System Center

Agenda

- z/OS Connect Introduction and overview
- Discuss enabling RESTful API to various z/OS resources, e.g.
  - CICS
  - Db2
  - IMS/TM
  - IMS/DB
  - MQ
  - MVS Batch
  - Outbound REST APIs
  - IBM DVM
  - 3270 screen-based applications (HATS)
  - IBM File Manager

- Overview of z/OS Connect Security, for more on security, see workshop *zOSSEC1 IBM z/OS Connect Security Wildfire Workshop*

mitchj@us.ibm.com

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

z/OS Connect EE

z/OS Connect EE

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

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

mitchj@us.ibm.com

/but_first, what_is_REST?

What makes an API “RESTful”?

mitchj@us.ibm.com

REST is an Architectural Style

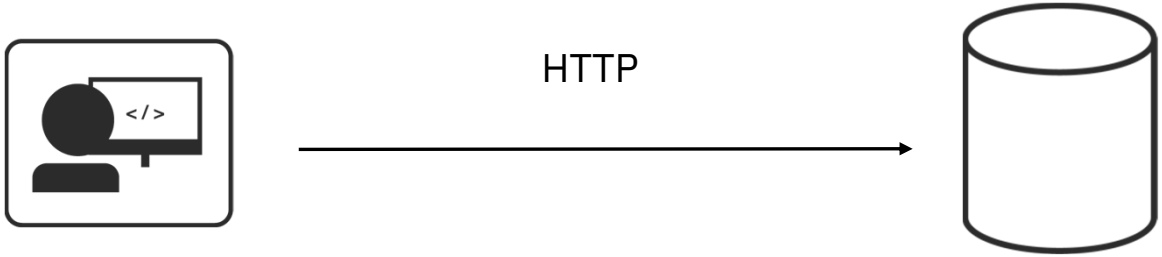
z/OS Connect EE

REST stands for Representational State Transfer.

An architectural style for **accessing** and **updating** data.

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

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



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

mitchj@us.ibm.com

Key Principles of REST

z/OS Connect EE

Path and Query parameters are used for refinement of the request

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

GET
POST
PUT
DELETE

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

URI path identifies a resource (or lists of resources)

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

z/OS Connect EE

- REST is an architectural style of development having these principles plus..
- It should be stateless
- It should access all the resources from the server 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 JSON

- REST based services follow some of the above principles and not all, whereas RESTful means it follows all the above principles.

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

mitchj@us.ibm.com

RESTful Examples

z/OS Connect EE

z/OS Connect Enterprise Edition:

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

GET /account?number=1234

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

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

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

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

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

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

mitchj@us.ibm.com

Not every REST API is a RESTful API

z/OS Connect EE

(How to know if an API is not RESTful)

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

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

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

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



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

3. Operational data embedded in the request body

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



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

mitchj@us.ibm.com

Why is REST popular?

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

mitchj@us.ibm.com

How do we describe a REST API?

mitchj@us.ibm.com

/swagger/open_api

The industry standard framework for describing RESTful APIs.

mitchj@us.ibm.com

Why use Swagger?

z/OS Connect EE

It is more than just an API framework

OPEN API INITIATIVE

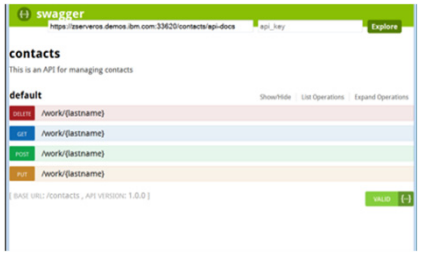
There are a number of tools available to aid consumption:

Consume Swagger

Swagger Codegen create stub code to consume APIs from various languages

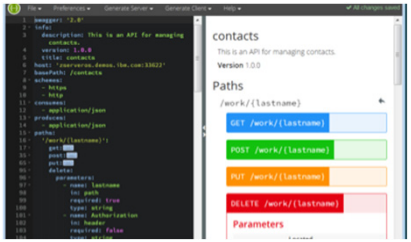
Read Swagger+

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



Write Swagger

Swagger Editor allows API developers to design their swagger documents.



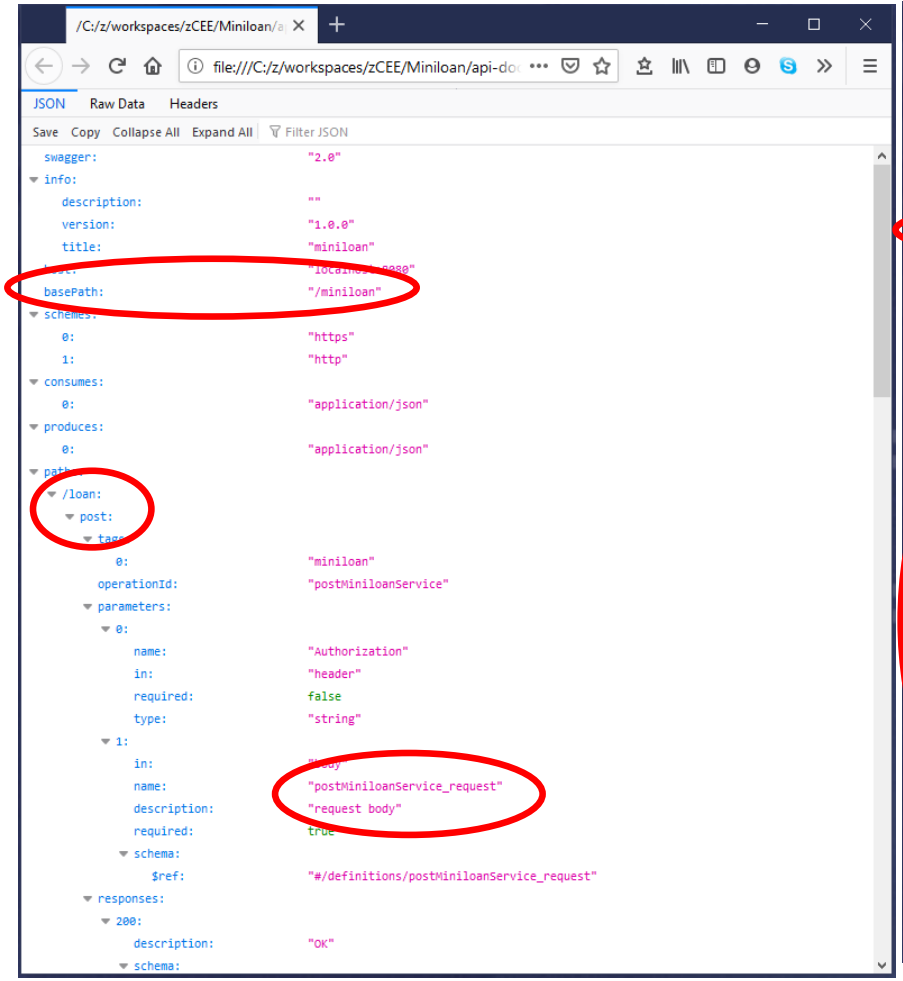
https://blog.readme.io/what-is-swagger-and-why-it-matters/

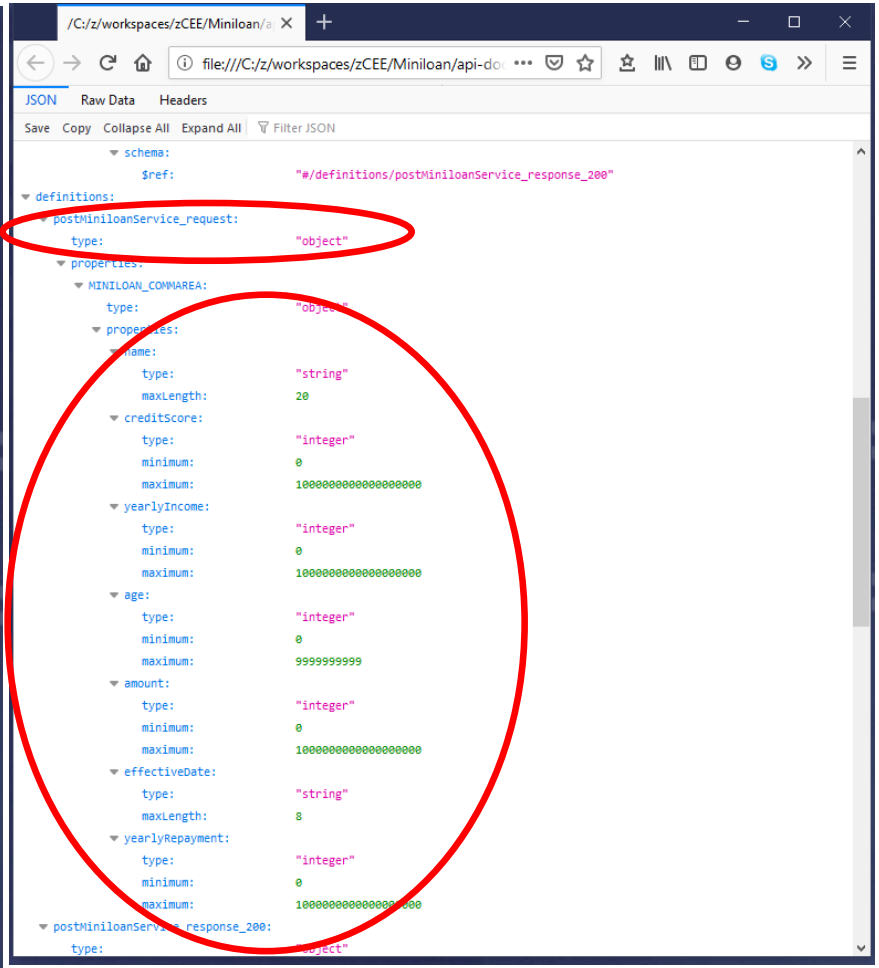
mitchj@us.ibm.com

+z/OS Connect, MQ, Zowe

Swagger Example

z/OS Connect EE





mitchj@us.ibm.com

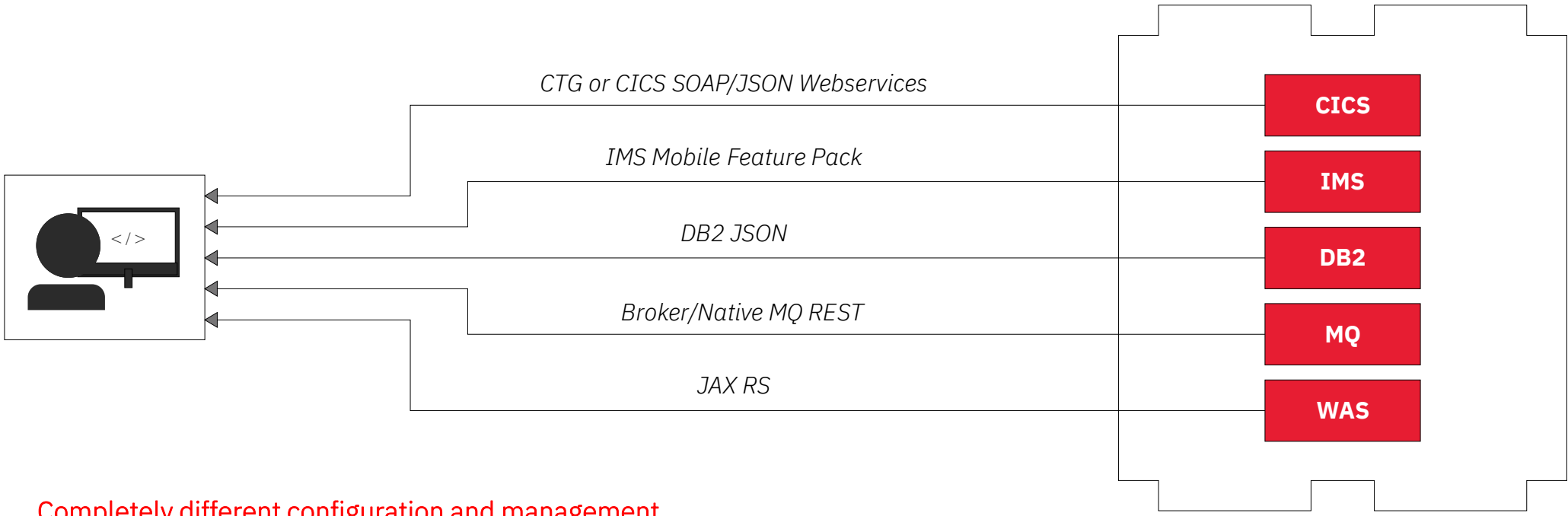
Why /zos_connect_ee?

Truly RESTful APIs to and from your mainframe.

mitchj@us.ibm.com

Could we not do REST before z/OS Connect? Yes, but....

z/OS Connect EE



Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

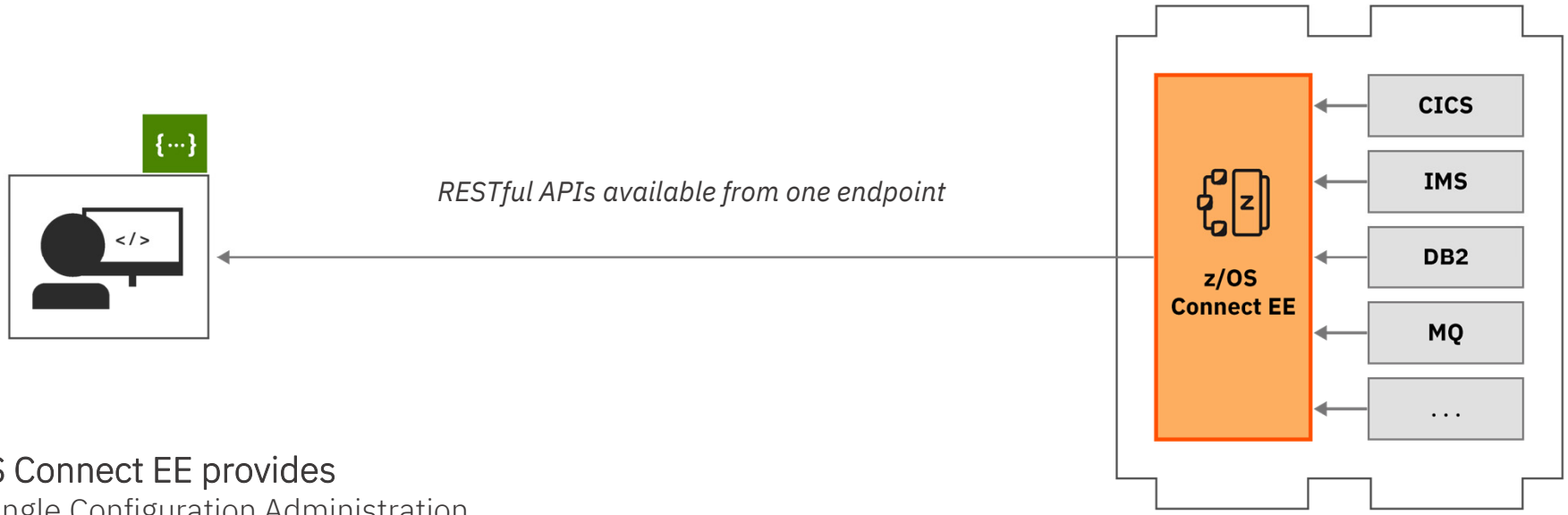
These are typically not RESTful!

mitchj@us.ibm.com

A single entry point was needed

z/OS Connect EE

Expose z/OS resources without writing any code.



z/OS Connect EE provides

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

mitchj@us.ibm.com

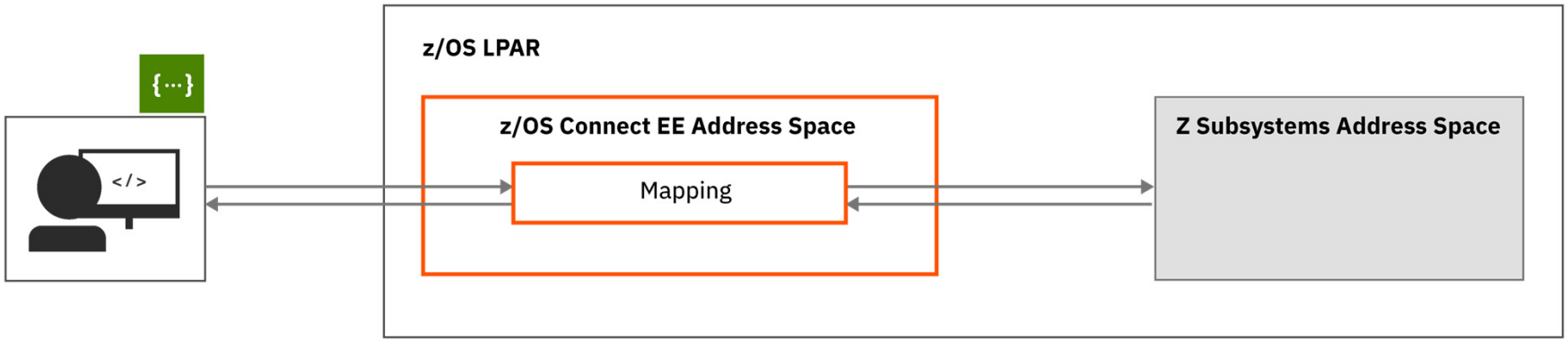
Other than a RESTful interface, what does z/OS Connect provide?

mitchj@us.ibm.com

Let’s Start with Data mapping

z/OS Connect EE

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



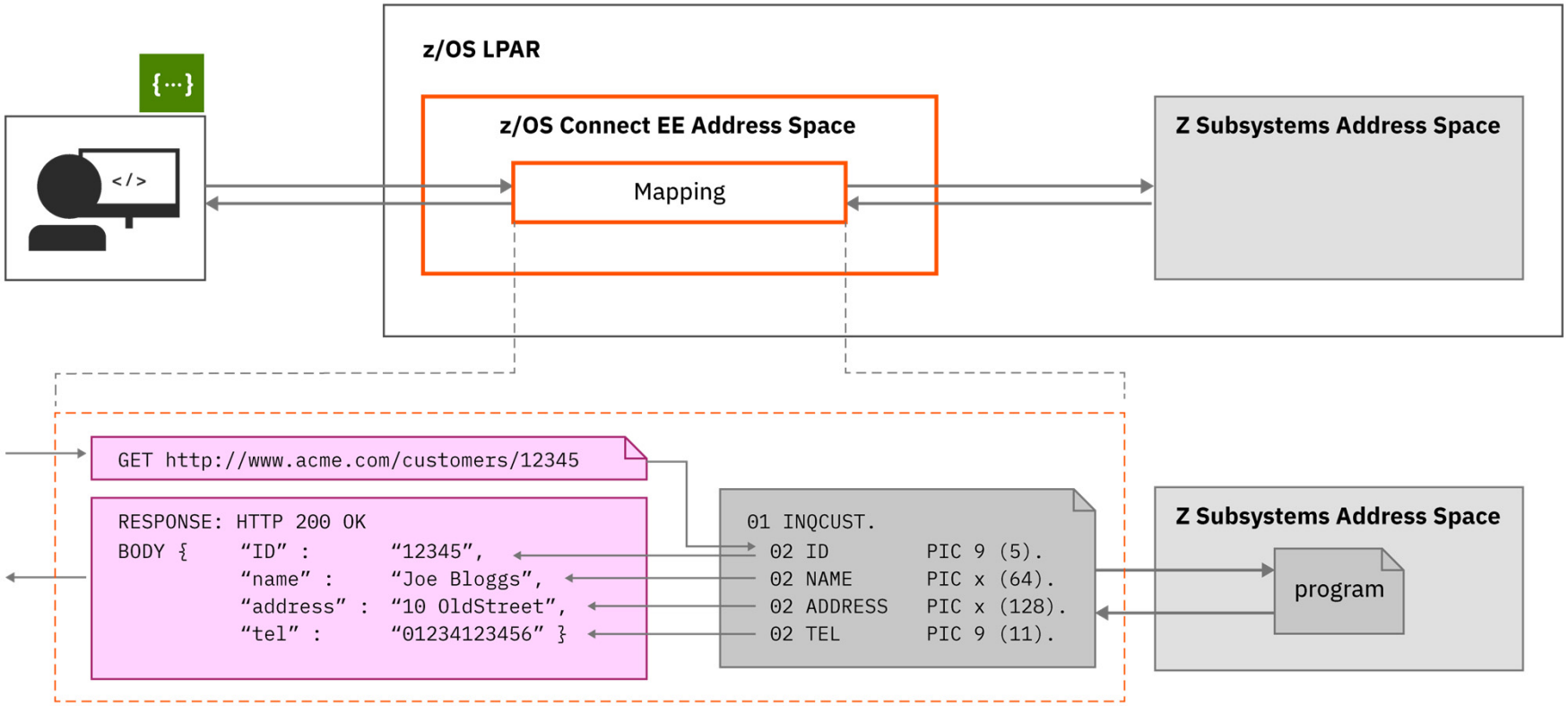
* Most z/OS subsystems depend information in a serial data format and do not normally work with JSON request/response messages. Examples of non-JSON formats are CICS COMMAREAs and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation refers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

mitchj@us.ibm.com

Data mapping Example

z/OS Connect EE

A closer look



mitchj@us.ibm.com

COBOL versus JSON Schema Example

z/OS Connect EE

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

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

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

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

miniloanRequest.si

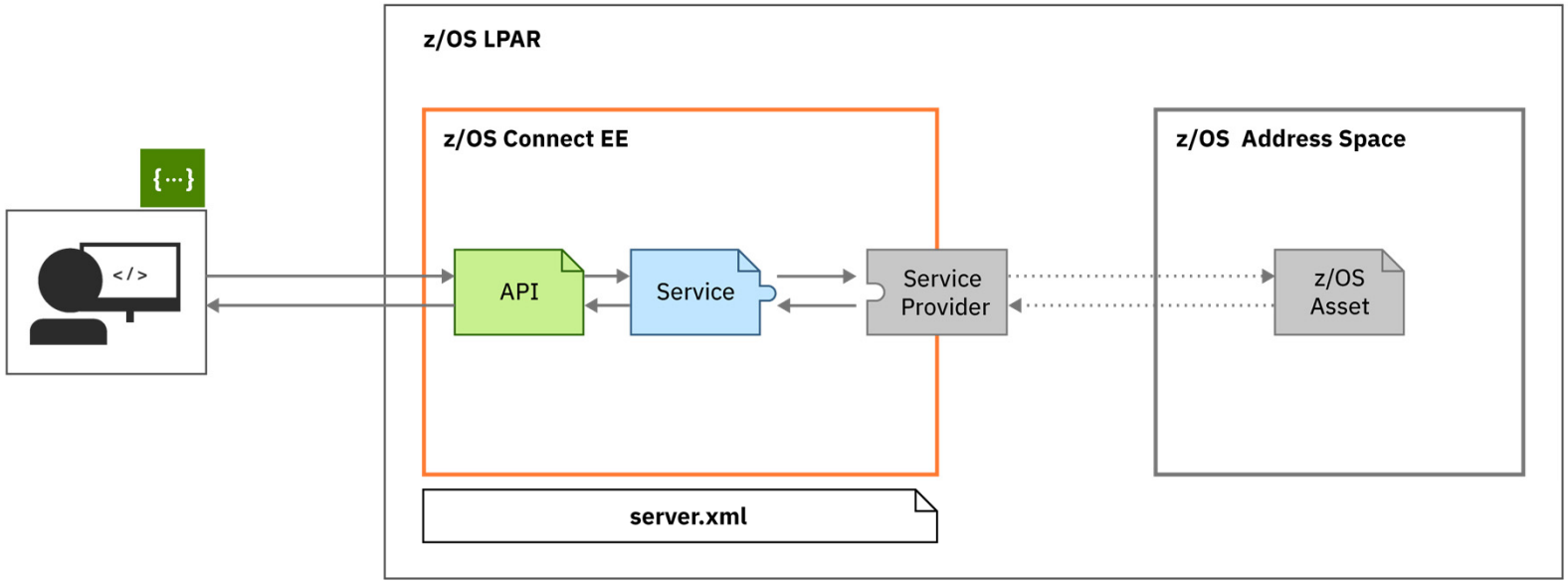
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:message xmlns:ns2="http://www.ibm.com/ims/Transaction" transactionCode="" messageName="miniloanRequest" direction="0" serviceType
    <message id="1" name="miniloanRequest">
        <segment id="1" name="COMMAREA" originalName="COMMAREA">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

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

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE



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

mitchj@us.ibm.com

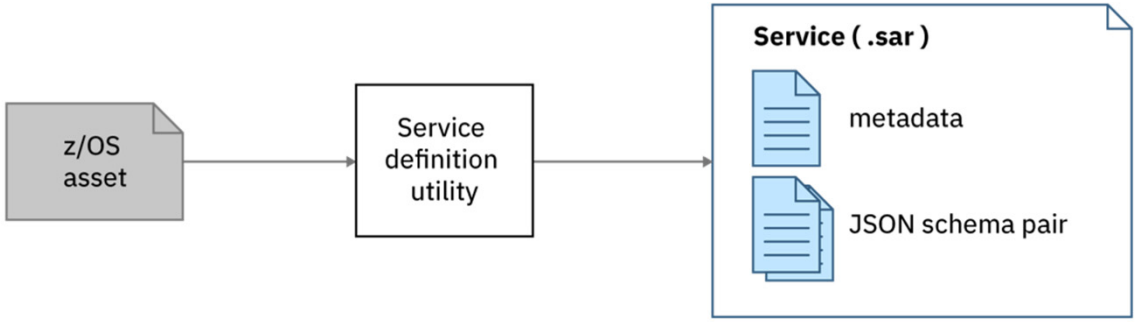
Steps to expose a z/OS application

z/OS Connect EE

1. Create a service

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



Use a system-appropriate utility to generate a `.sar` file for the z/OS application

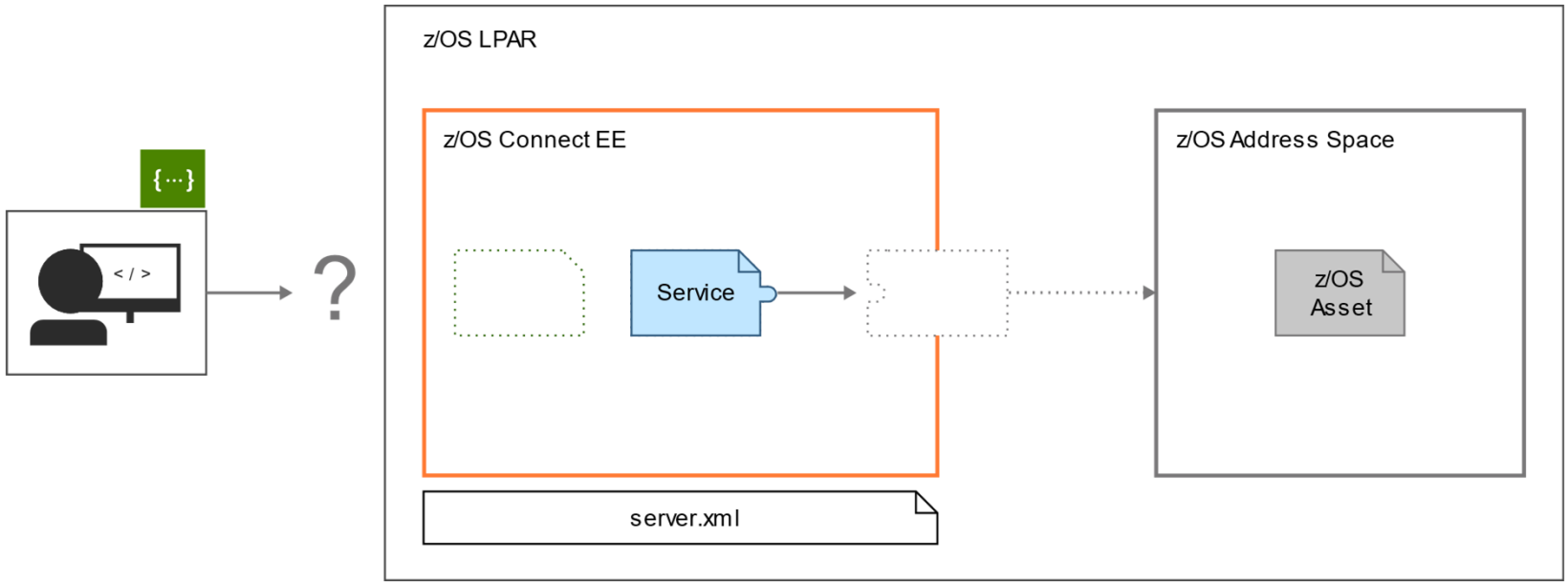
- Eclipse based - API Toolkit (CICS, IMS TM, IMS DB, Db2 and MQ)
- Command line - z/OS Connect EE Build Toolkit (MVS Batch, IBM File Manager and HATS)
- Eclipse based - DVM Toolkit

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

2. Deploy and export the service



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

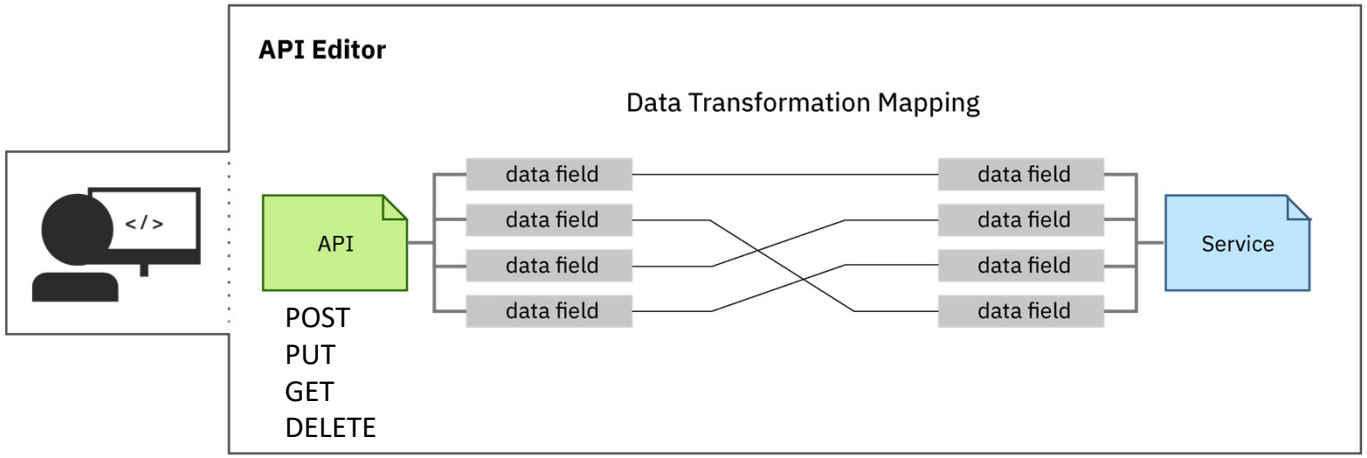
mitchj@us.ibm.com

© 2018, 2021 IBM Corporation

Steps to expose a z/OS application

z/OS Connect EE

3. Create an API using exported services



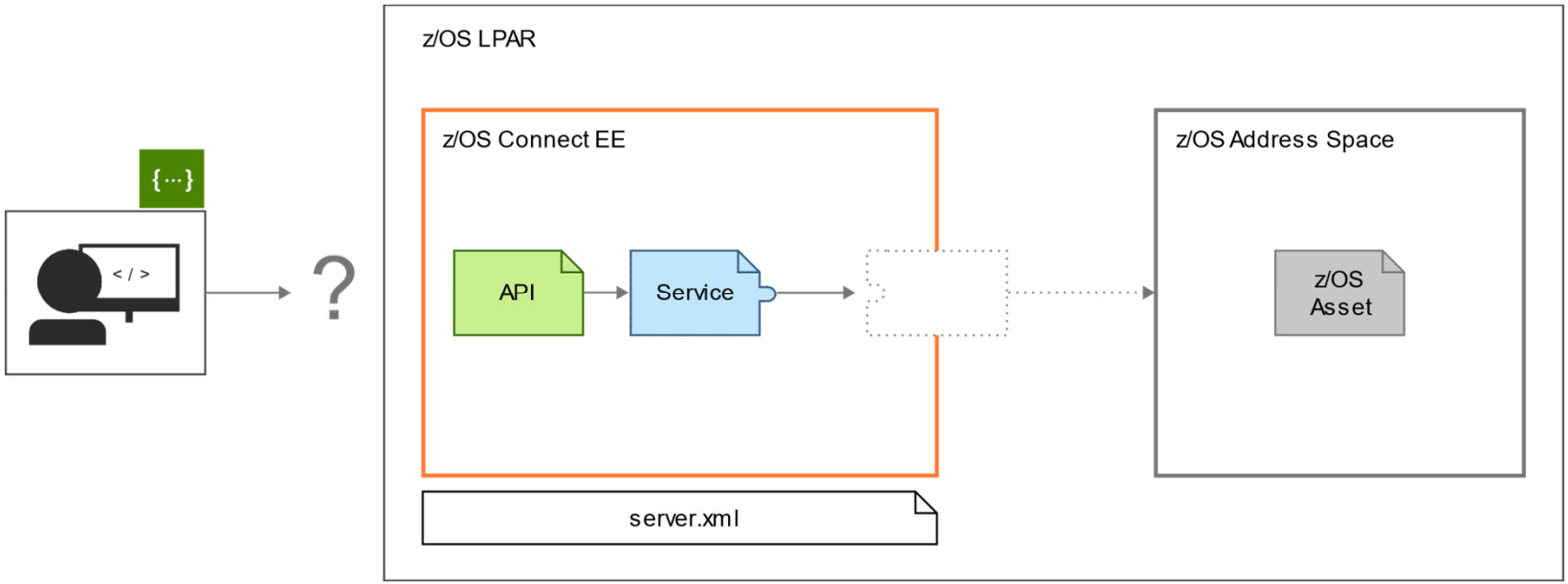
- Import the service archive file into the **API toolkit** and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

4. Deploy the API



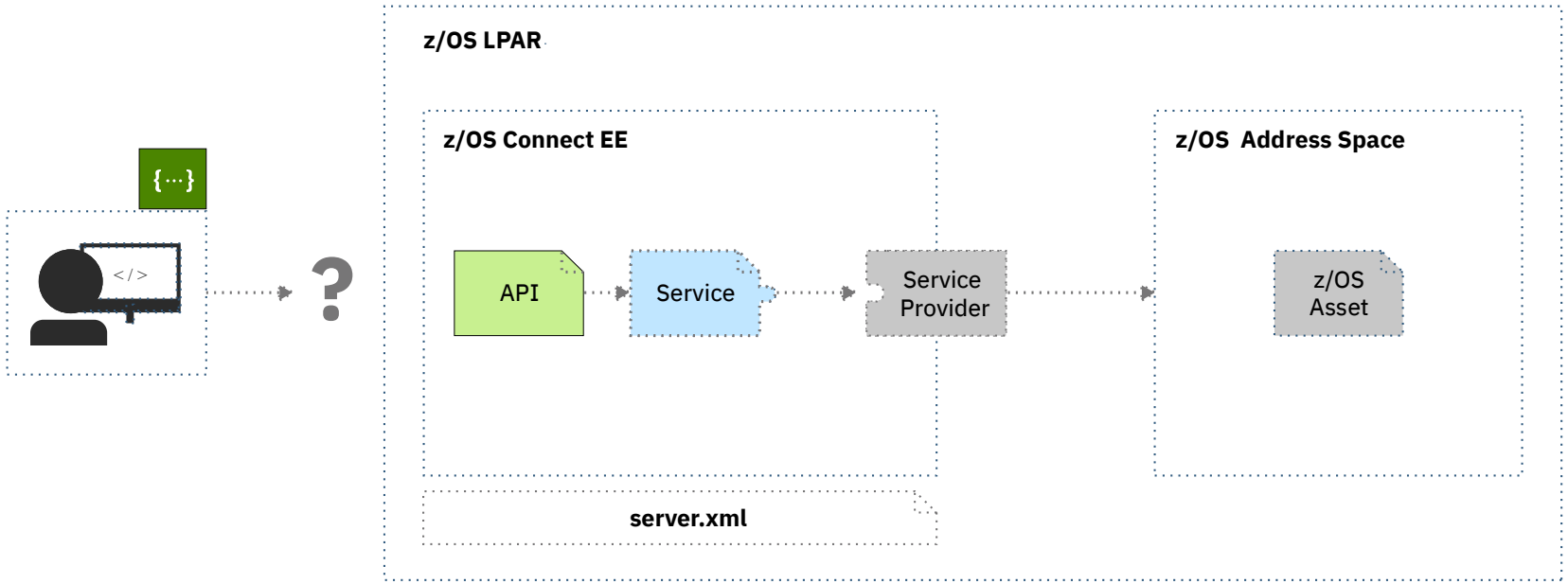
Deploy your API using the right-click deploy in **the API toolkit**

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

5. Configure the service provider



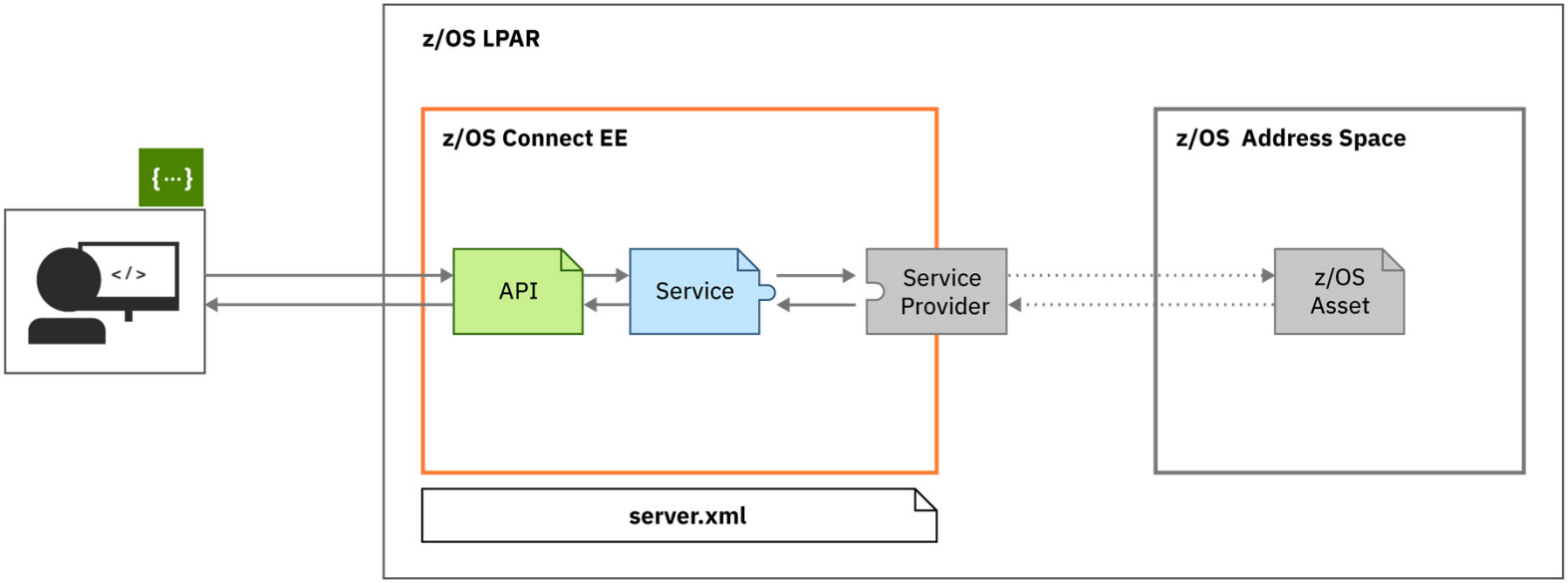
Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.

mitchj@us.ibm.com

Steps to expose a z/OS application

z/OS Connect EE

6. Done

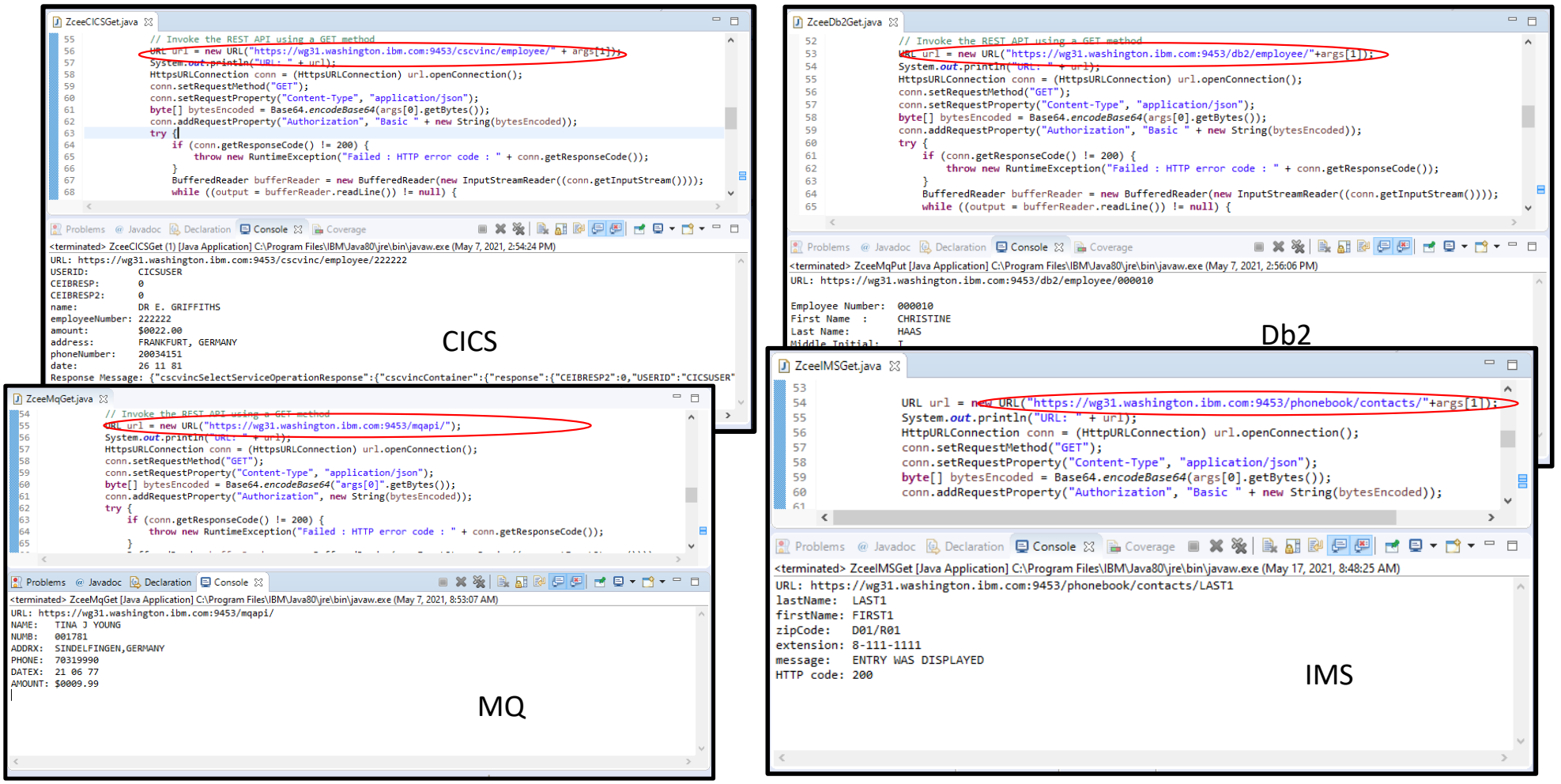


- The API is ready to be consumed and requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port)

mitchj@us.ibm.com

The Client code is unaware of the z/OS infrastructure

z/OS Connect EE



mitchj@us.ibm.com

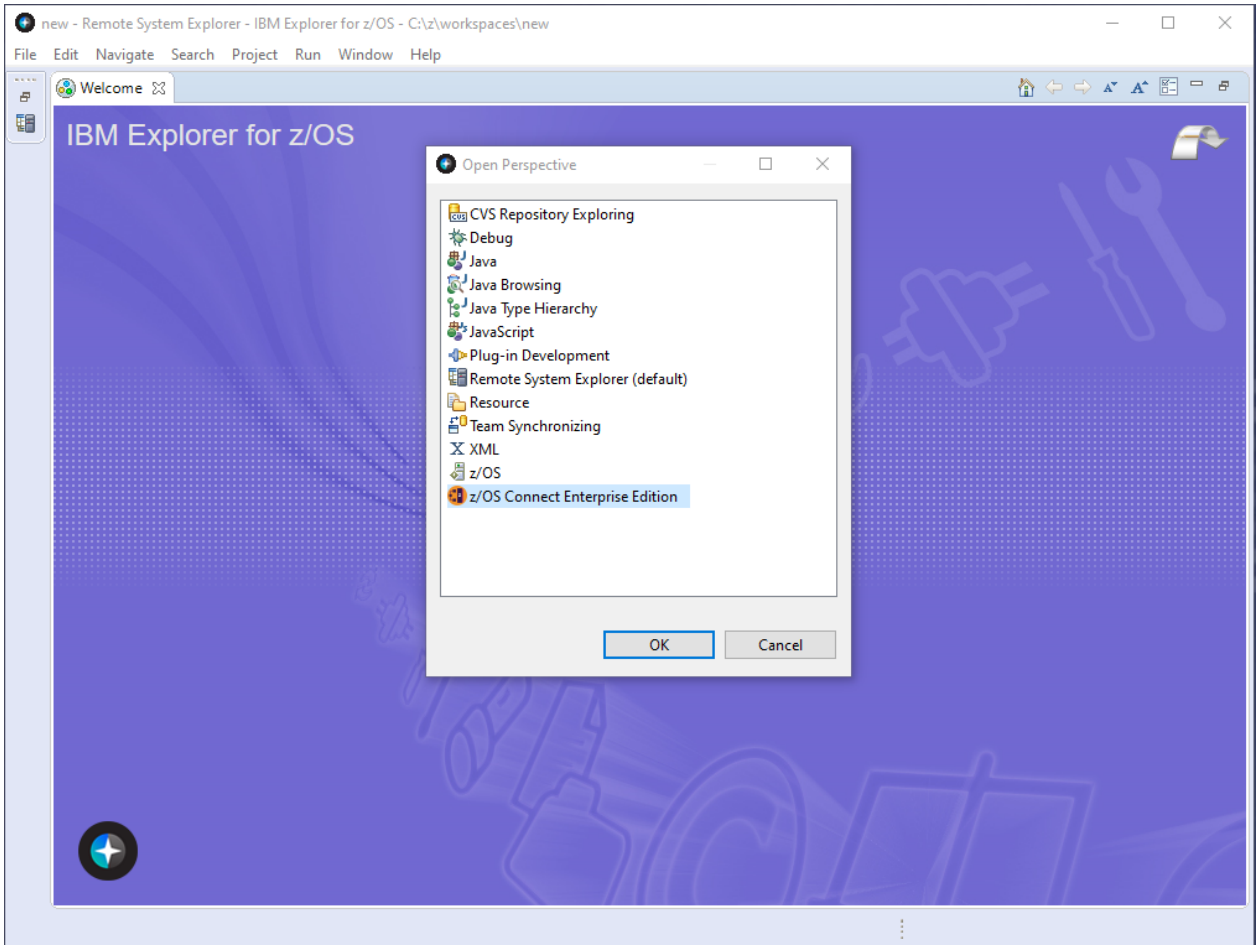
/api_toolkit/services

Simple **service creation**.

mitchj@us.ibm.com

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

z/OS Connect EE



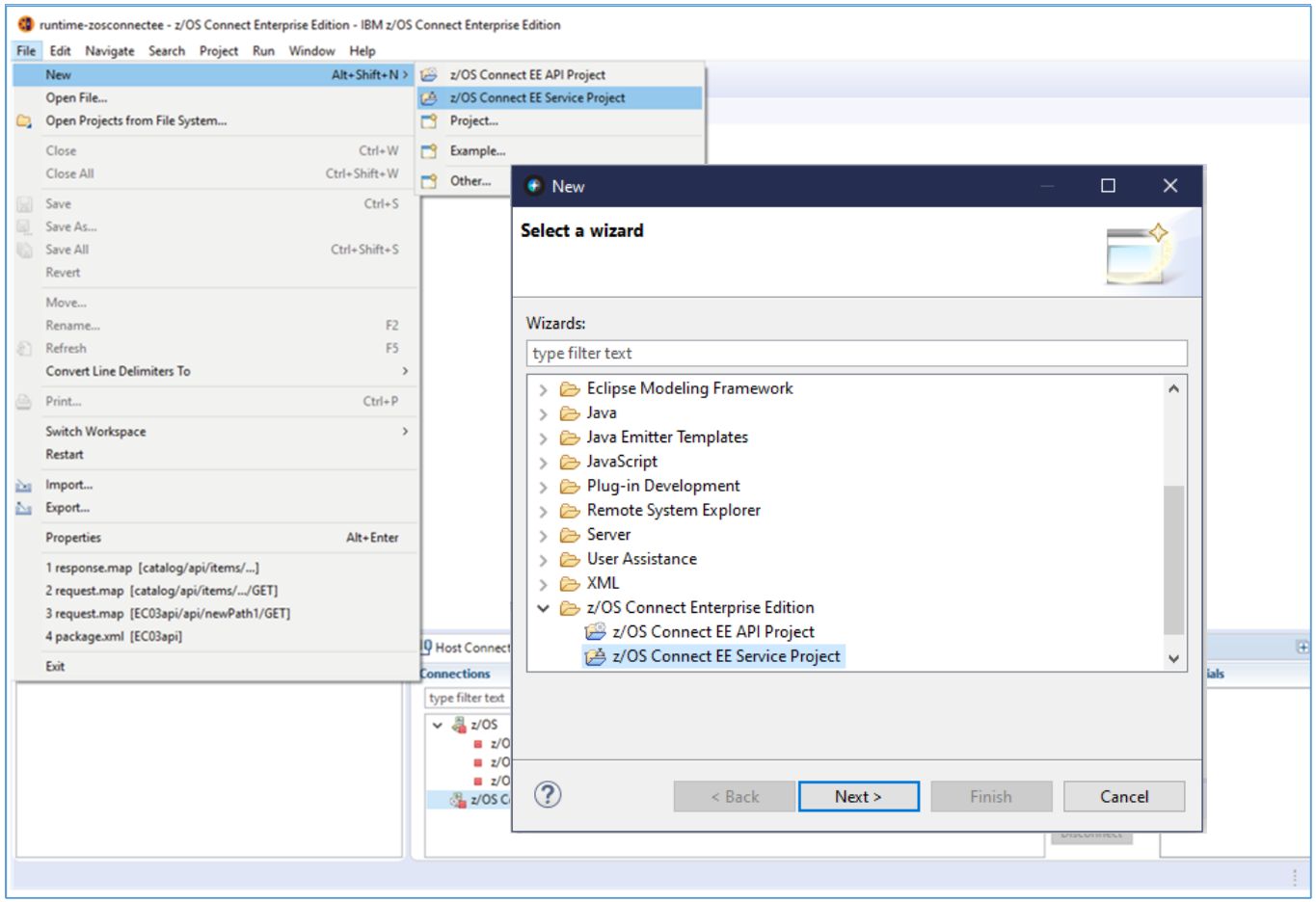
Use the **API toolkit** to create services
through Eclipse-based tooling.

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

mitchj@us.ibm.com

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

z/OS Connect EE



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

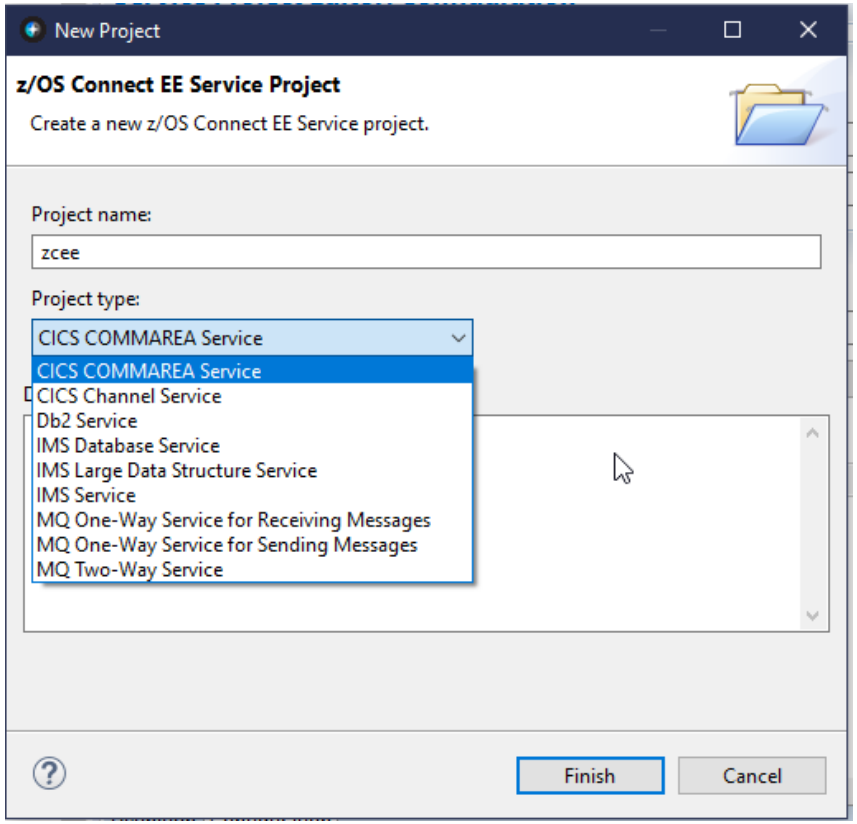
Services are described as Eclipse **Projects**, so they can be easily managed in source control.

mitchj@us.ibm.com

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

z/OS Connect EE

Service creation – a common interface



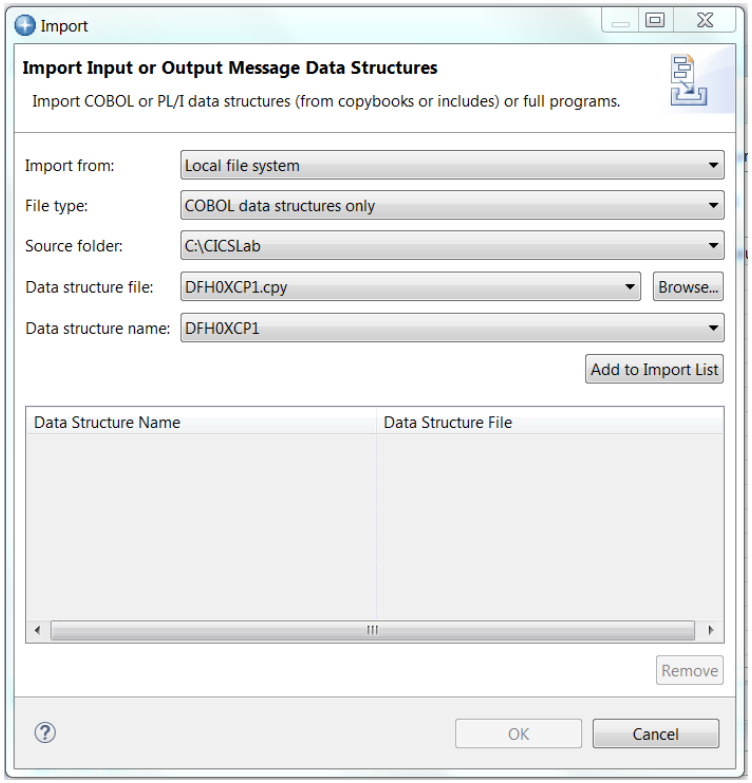
A common interface for service creation, agnostic of back end subsystem.

mitchj@us.ibm.com

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

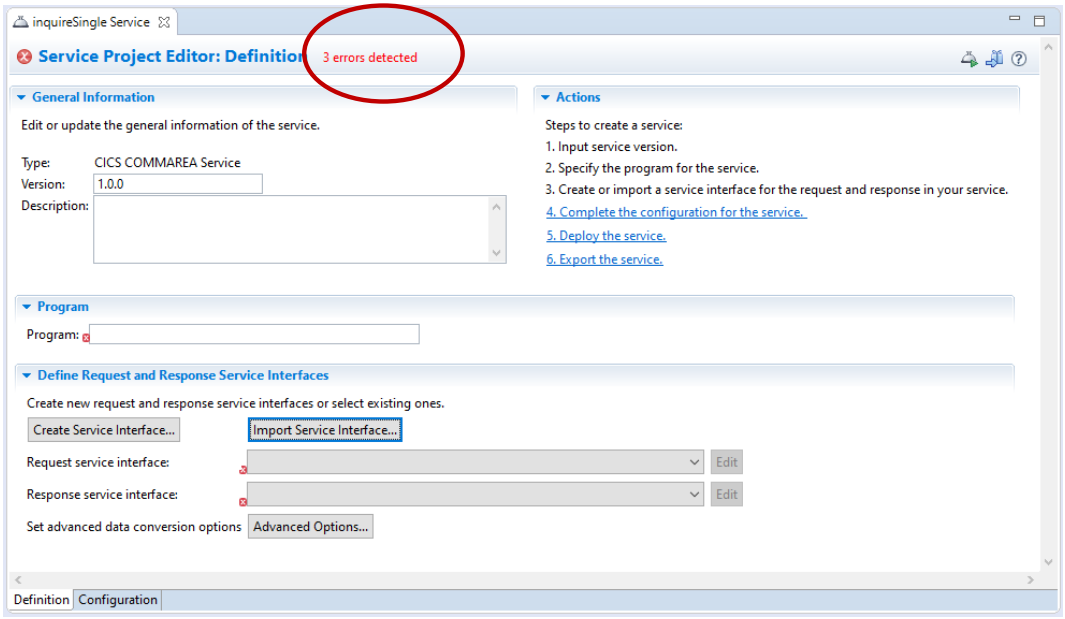
z/OS Connect EE

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



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

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



mitchj@us.ibm.com

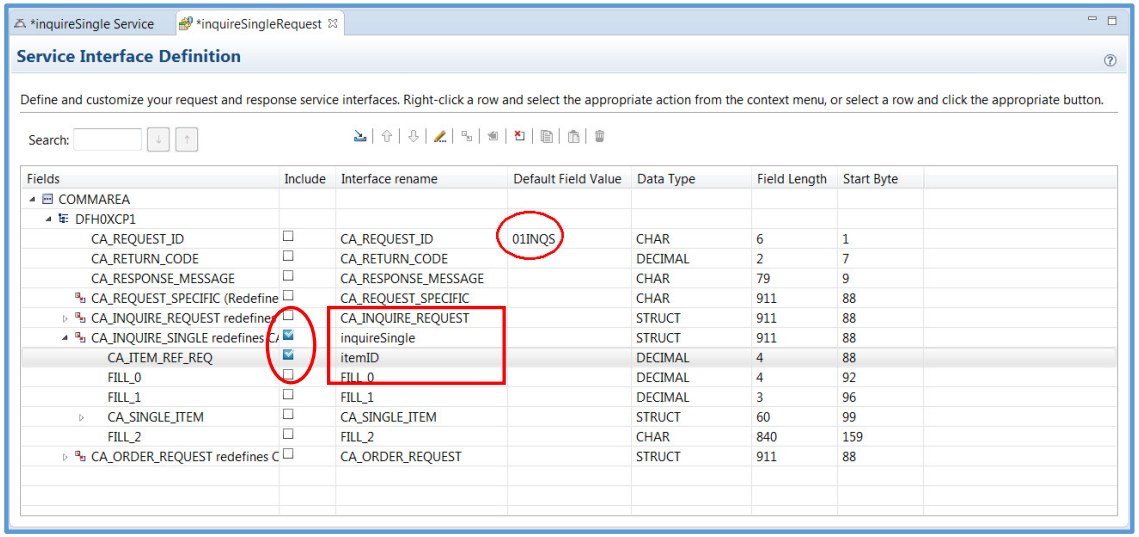
API toolkit – Creating Services for CICS, IMS TM and MQ

z/OS Connect EE

Allows editing a request service interface definition

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

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

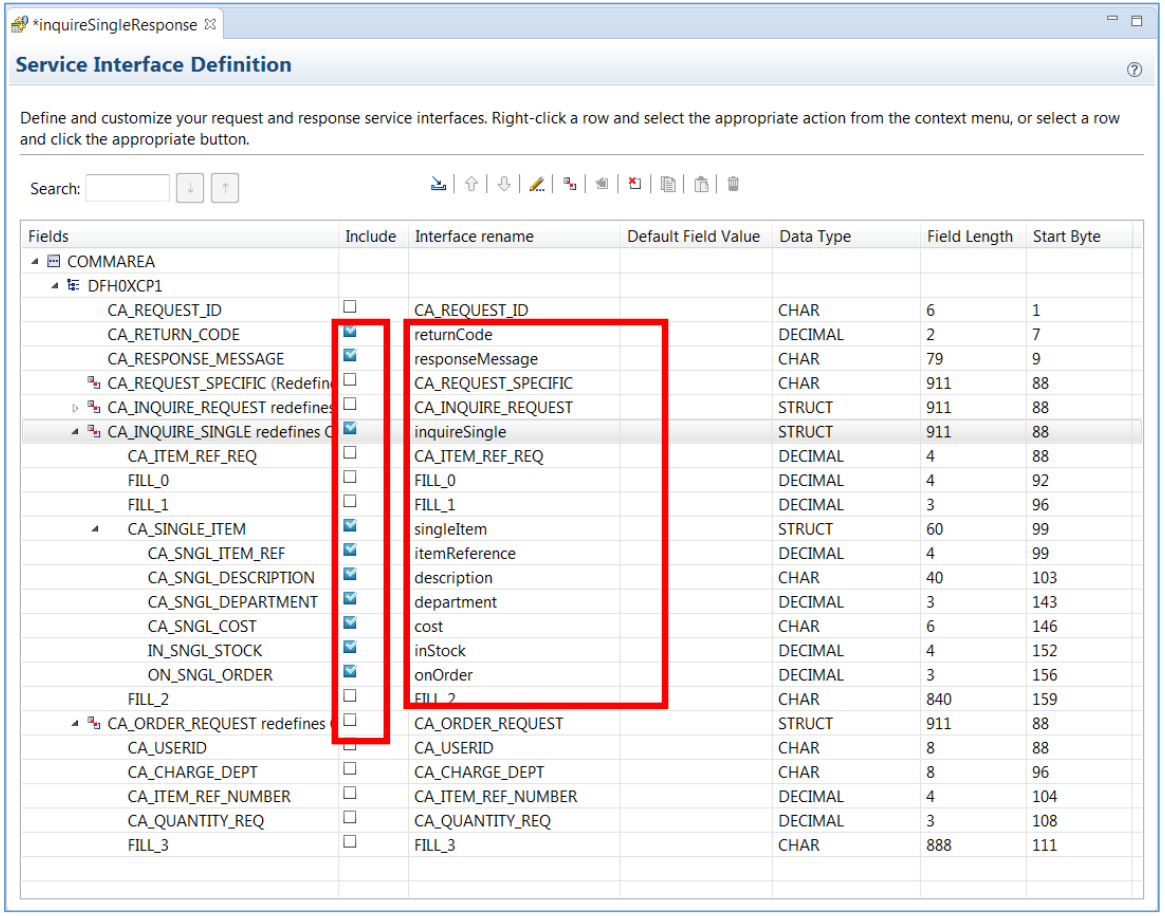


mitchj@us.ibm.com

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

And editing a response message service interface definition

z/OS Connect EE



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

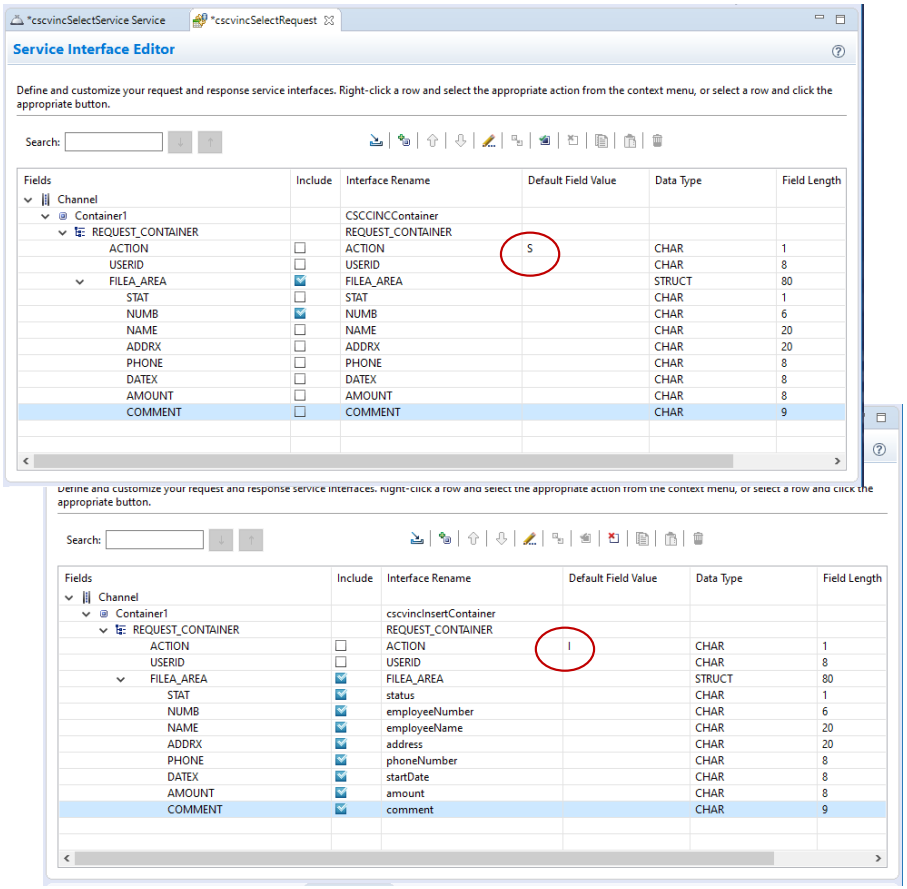
See the imported data structure and can **redact fields** and **rename fields**

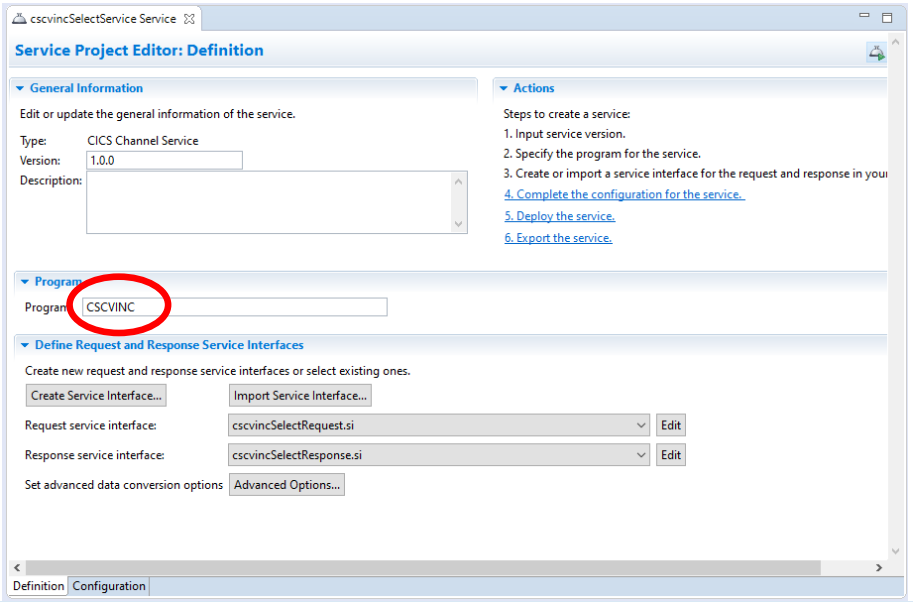
mitchj@us.ibm.com

API toolkit – Creating Services for CICS

Creating multiple services definitions to the same resource

z/OS Connect EE






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

mitchj@us.ibm.com

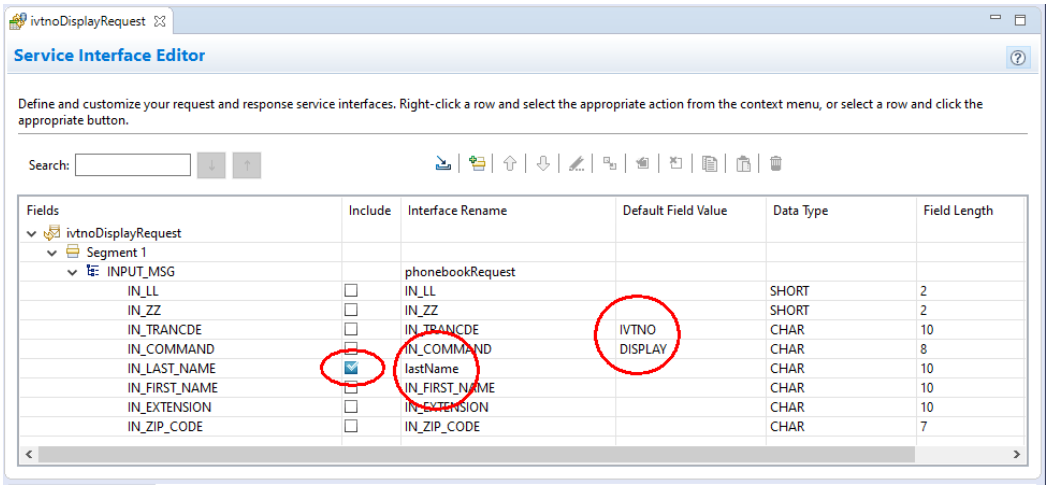
The service developer creates distinct services for each function by setting the ACTION field to S for select, I for insert, U for update or D for delete

API toolkit – Creating Services for IMS

Creating a “GET” service interface request definition

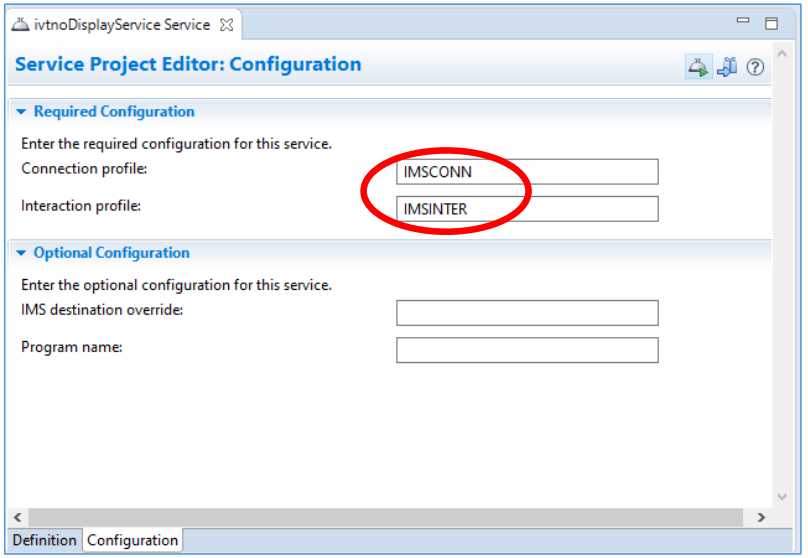
z/OS Connect EE

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



The service developer creates distinct services for each function.

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



IMS/TM Meta Data

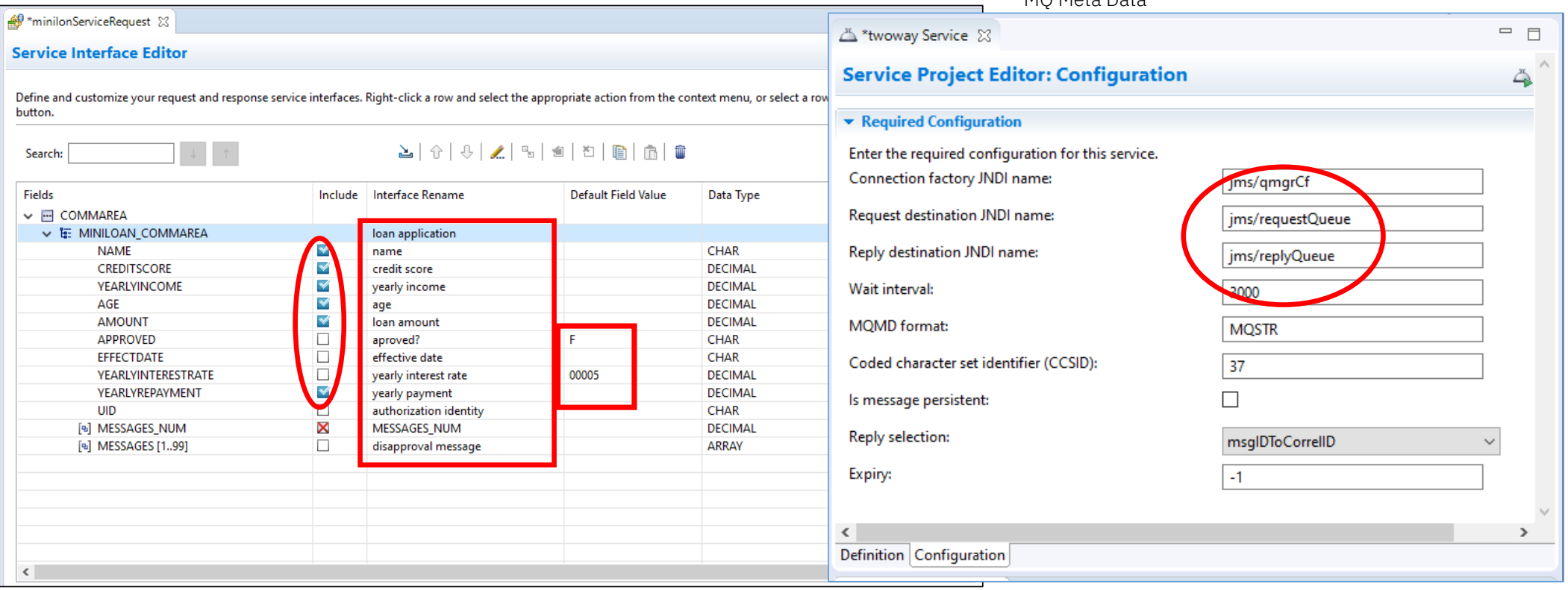
mitchj@us.ibm.com

API toolkit – Creating Services for MQ

z/OS Connect EE

Creating a “POST” service interface definition

MQ Meta Data



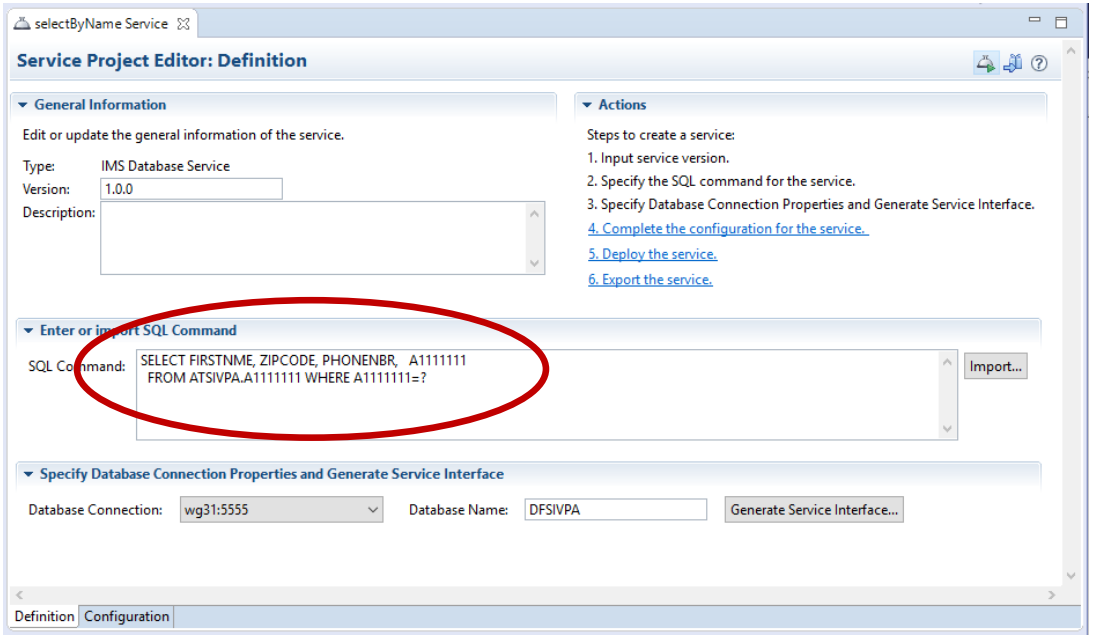
Again the service developer can then see the imported data structure and can **redact fields**, **rename fields**, and **add default values to fields** to make the service more consumable for an API developer.

mitchj@us.ibm.com

API toolkit – Creating Services for IMS DB

z/OS Connect EE

Creating a service project from the IMS Catalog



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

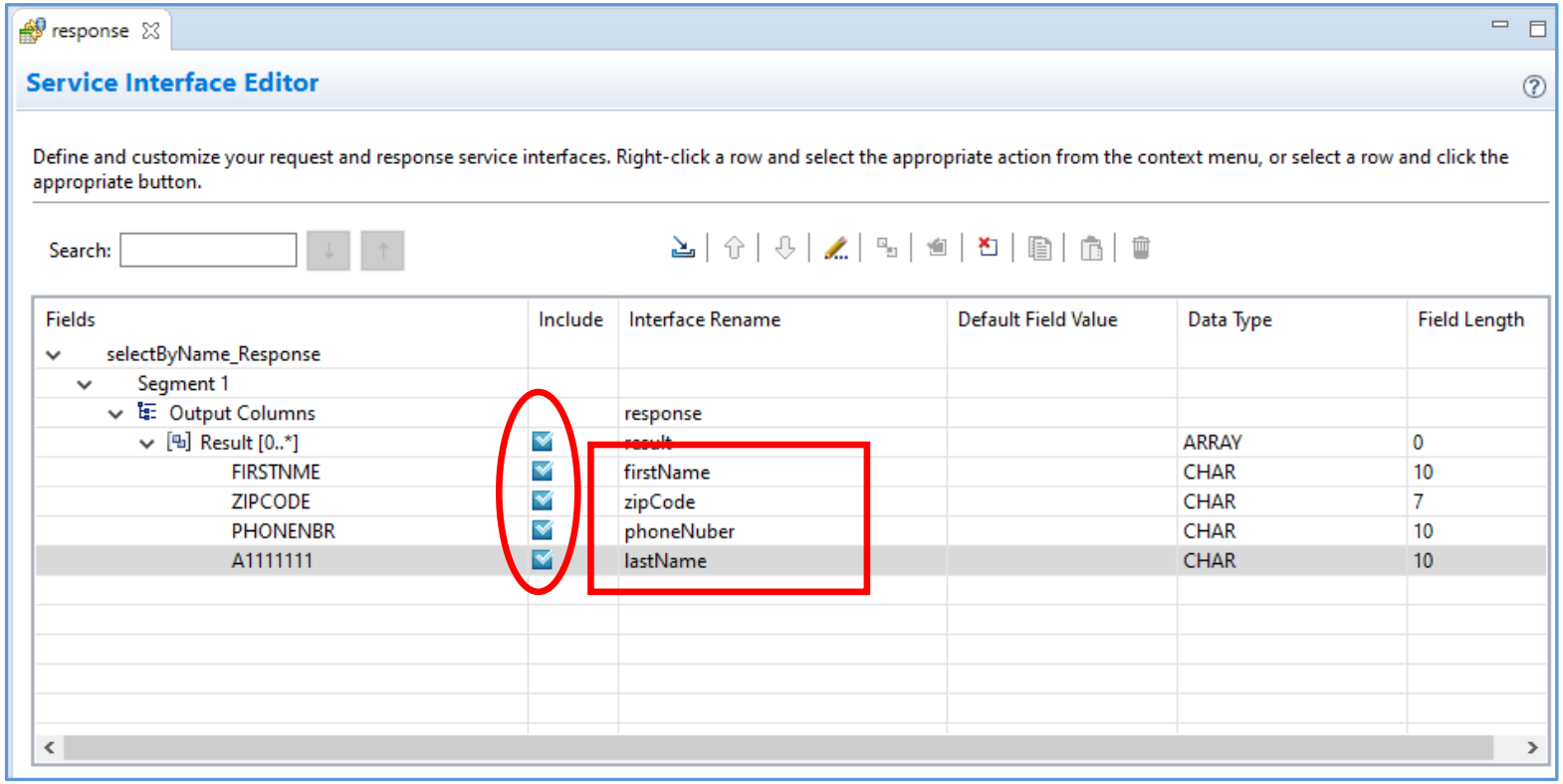
```
*-------------------------------------------------------------------*
*  SEGMENT DESCRIPTION                                              *
*    ROOT ONLY DATABASE                                             *
*      BYTES  1-10  LAST NAME (CHARACTER) - KEY                     *
*      BYTES  11-20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

mitchj@us.ibm.com

API toolkit – Creating Services for IMS DB

z/OS Connect EE

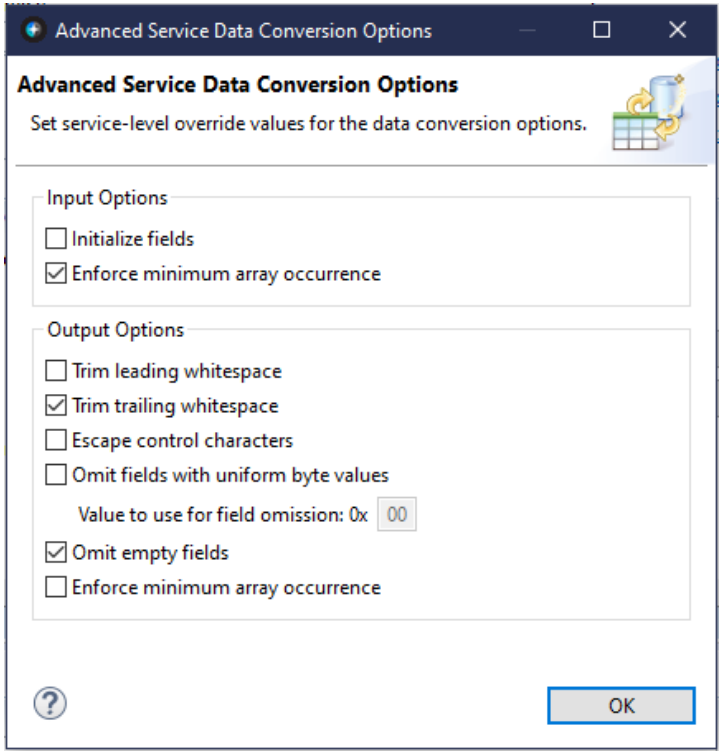
The Toolkit allows editing a service interface definitions*



mitchj@us.ibm.com

*Using a slightly different process

API toolkit – Advanced Data Conversion Options



Request Messages:

- Initialize fields
- Enforce minimum array occurrence

Response Messages:

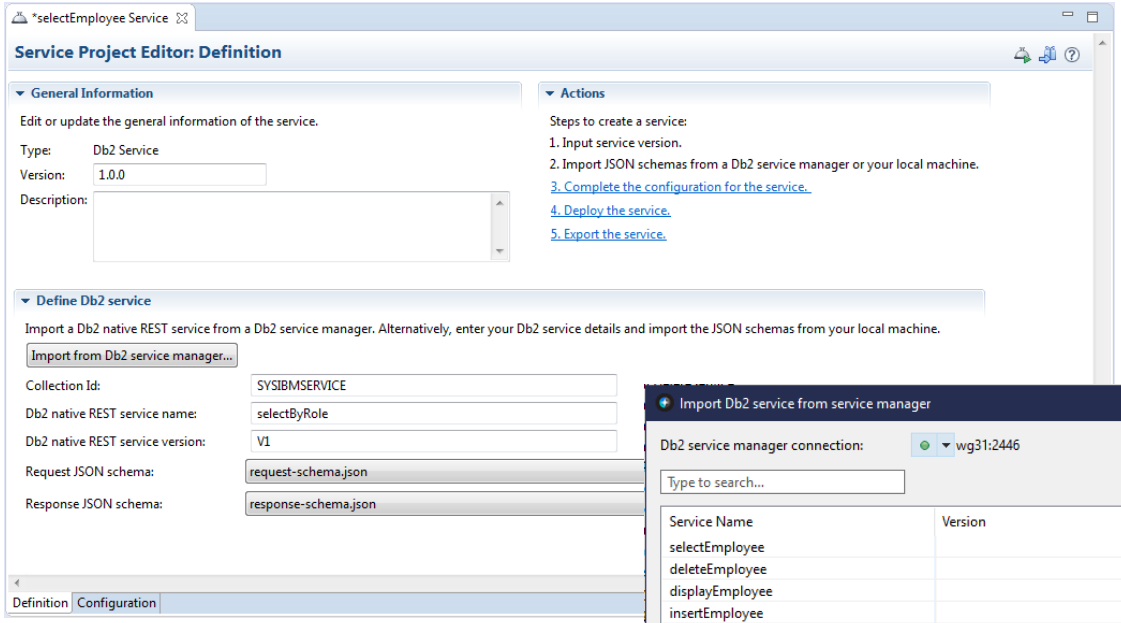
- Trim leading whitespace
- Trim trailing whitespace
- Escape control characters
- Omit fields with uniform byte values
- Omit empty fields
- Enforce minimum array occurrence

mitchj@us.ibm.com

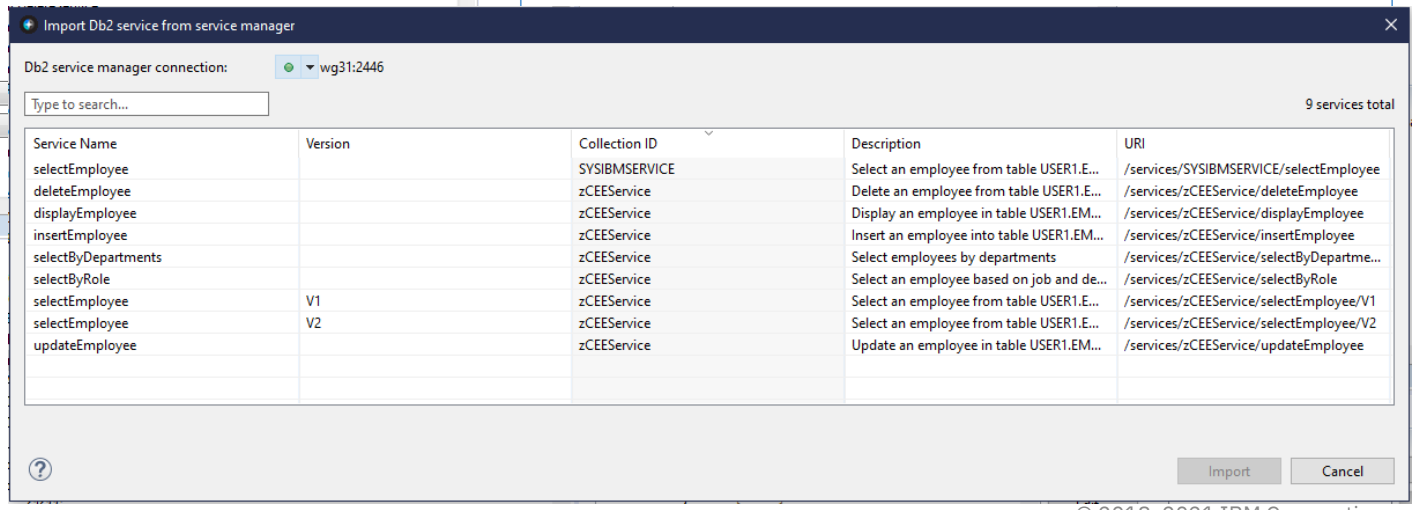
API toolkit – Creating Services for Db2

Creating a service project from Db2 REST service

z/OS Connect EE




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



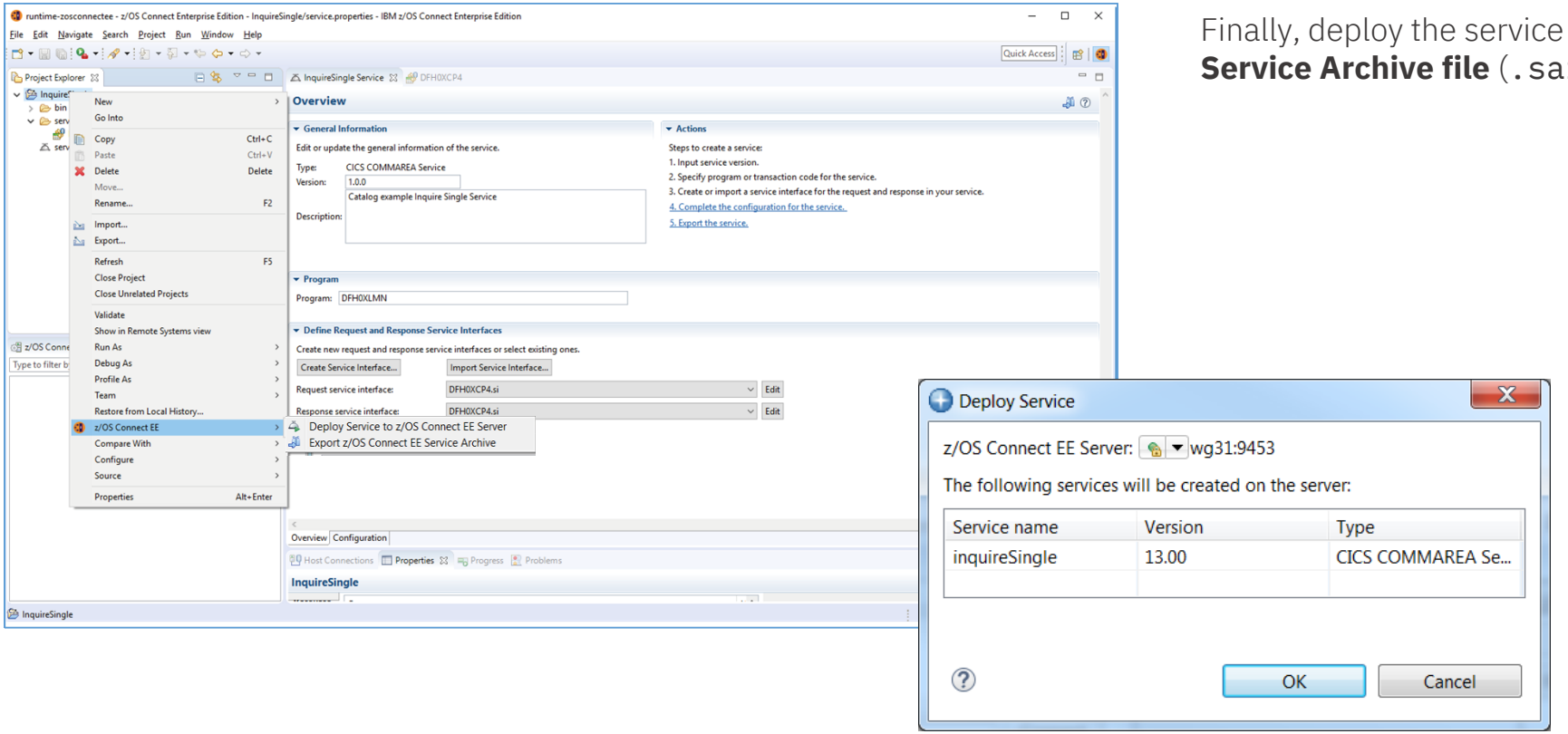
The service developer retrieves details about the Db2 REST services

Note there is no service interface editor available

mitchj@us.ibm.com

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

z/OS Connect EE

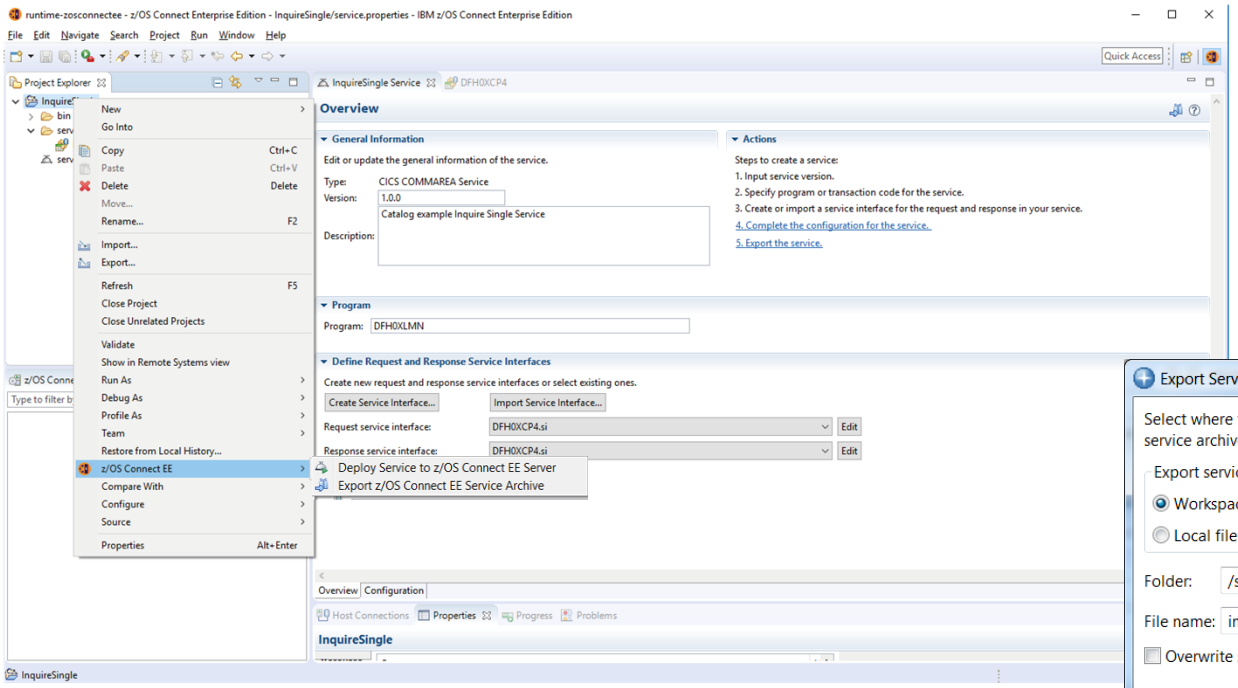


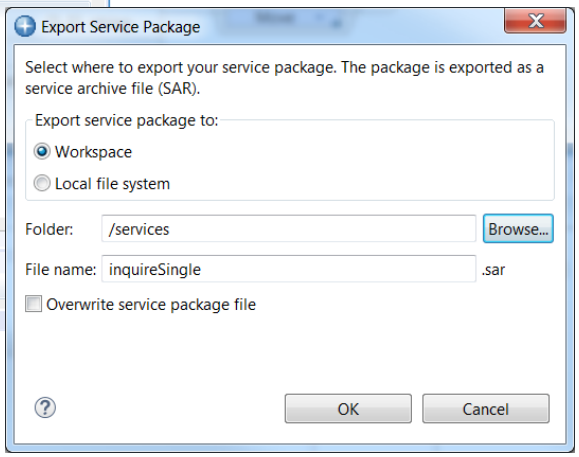
Finally, deploy the service project as a **Service Archive file** (`.sar`)

mitchj@us.ibm.com

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

z/OS Connect EE





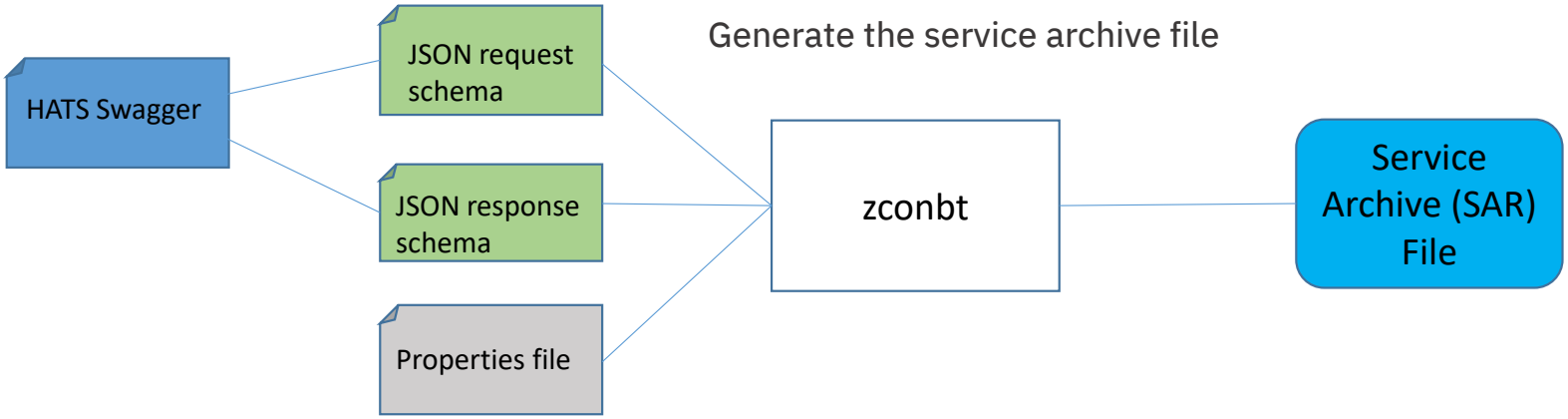
And export the service project as a **Service Archive file** (`.sar`).

mitchj@us.ibm.com

Creating Services using zconbt – REST

z/OS Connect EE

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




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

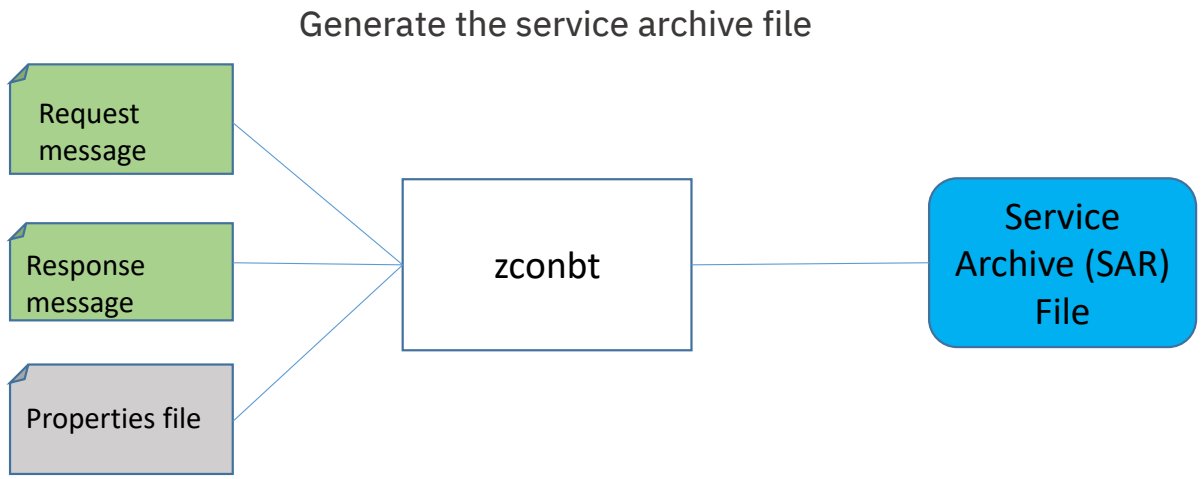
mitchj@us.ibm.com

Creating Services using zconbt – MVS Batch

z/OS Connect EE

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

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



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

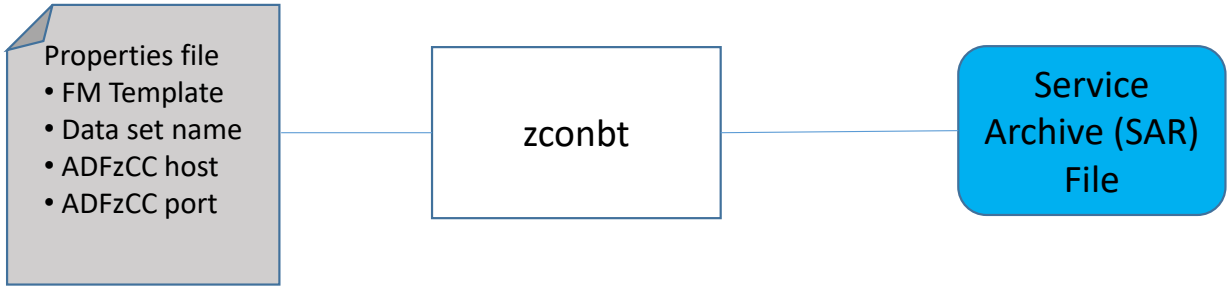
mitchj@us.ibm.com

Creating Services using zconbt – File Manager

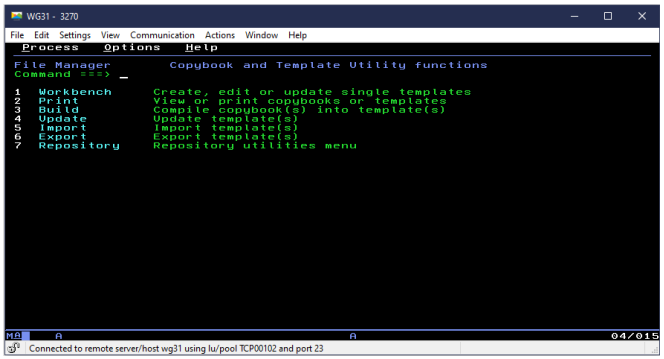
z/OS Connect EE

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

Generate the service archive file




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

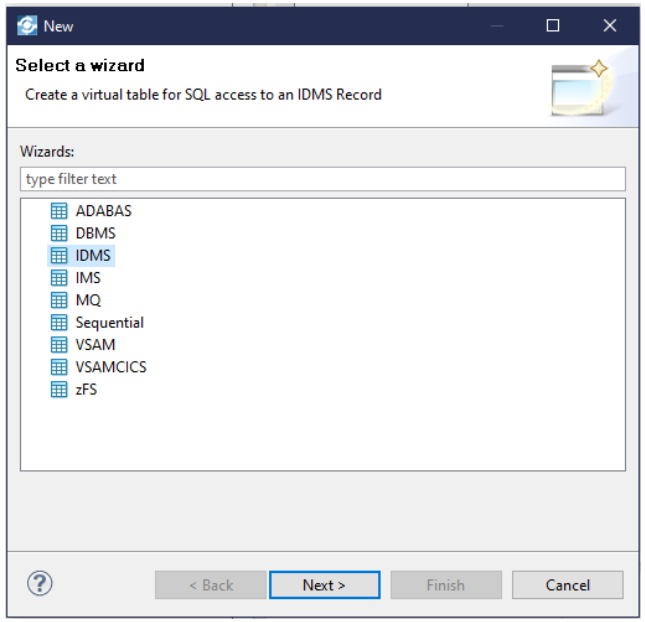


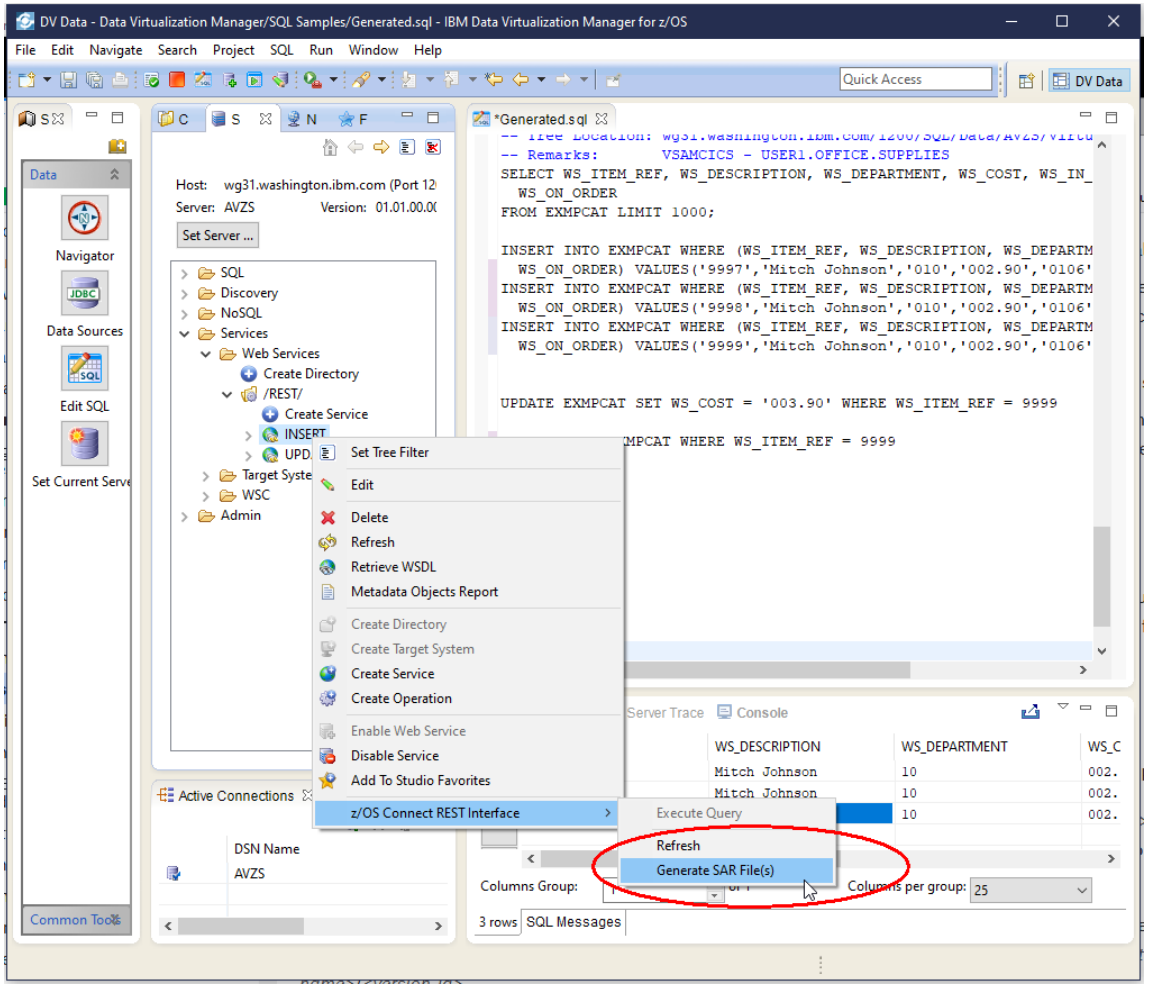
mitchj@us.ibm.com

Creating Services - DVM

z/OS Connect EE

For DVM use the DVM Studio





mitchj@us.ibm.com

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

Quick and easy **API mapping.**

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

mitchj@us.ibm.com

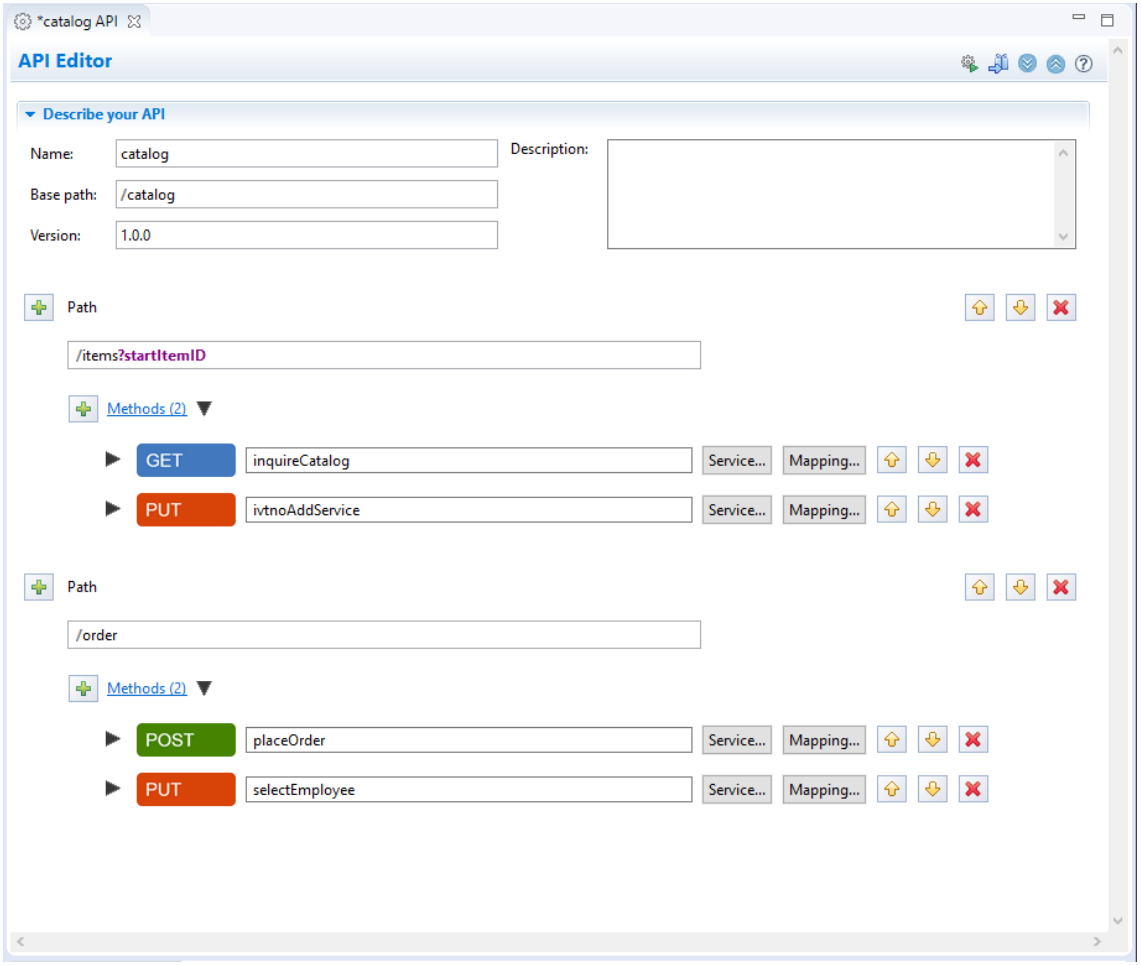
/api_toolkit/api_editor

Quick and easy **API mapping.**

mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE



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

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

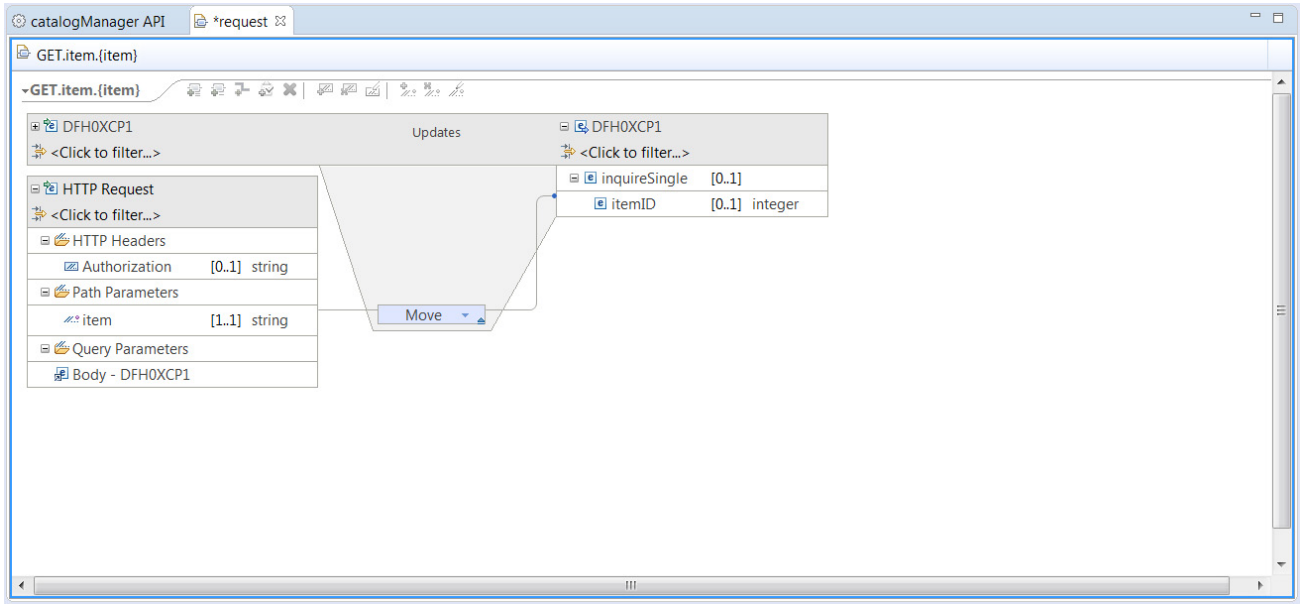
Your services are represented by a `.sar` files, which you import into the **API toolkit**, regardless of how the service archive file was generated.

mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

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



Map both the request and response for each API.

Map path and query parameters to native data structures.

Assign static values to fields, useful for Op codes.

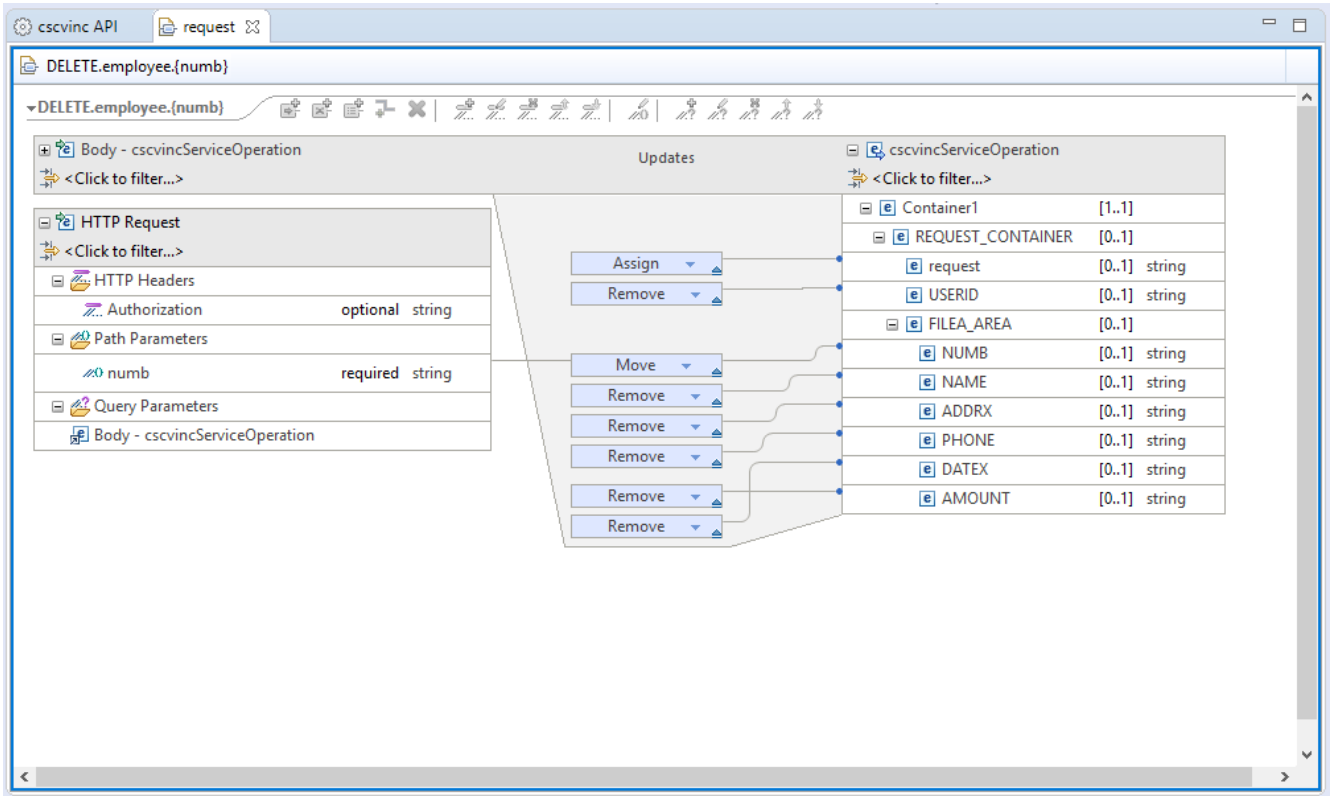
Remove unwanted fields to simplify the API (remember request was set to 01INQC in the SAR).

mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

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

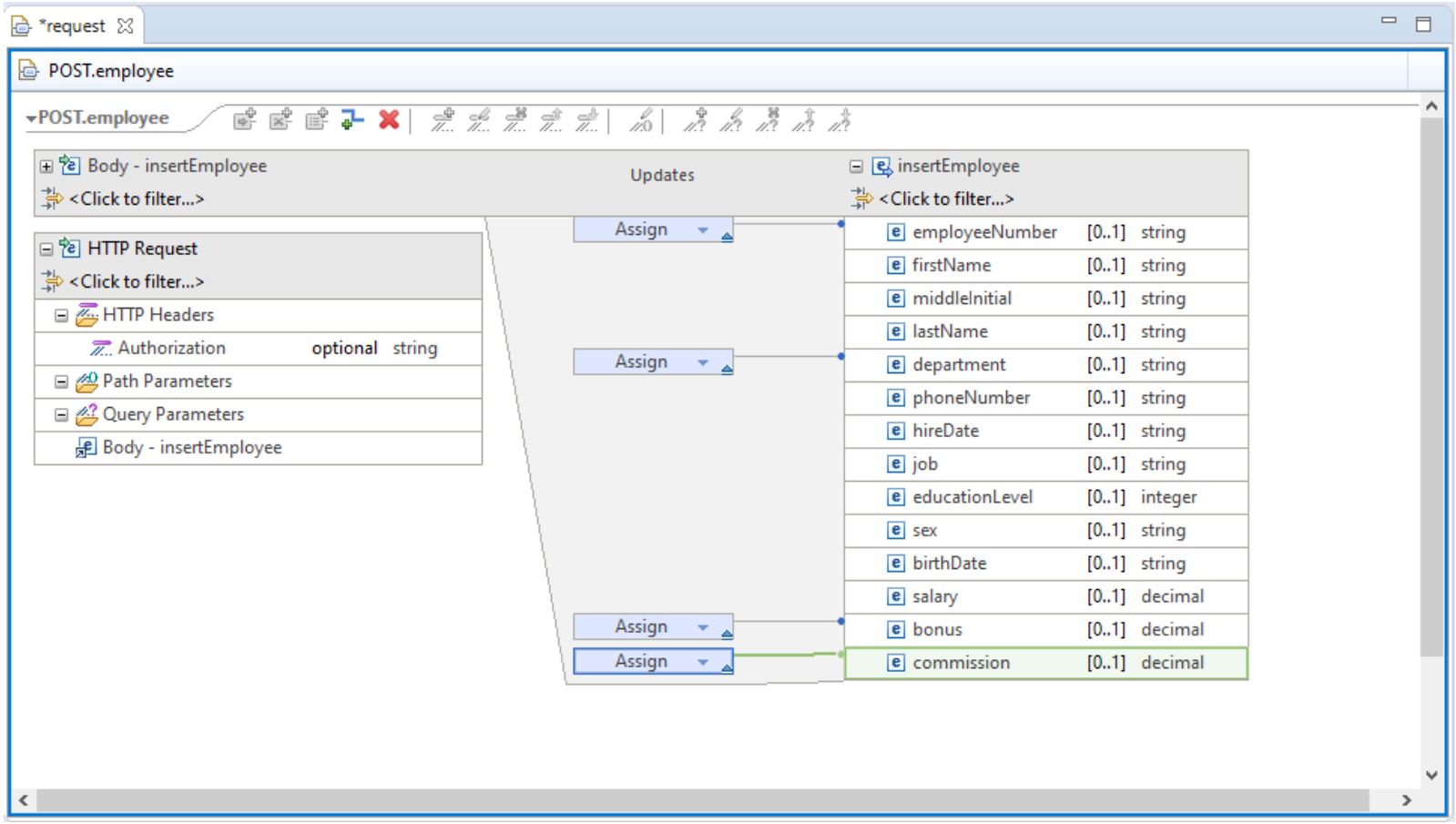


mitchj@us.ibm.com

API toolkit – API Editor and Db2 REST service

z/OS Connect EE

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

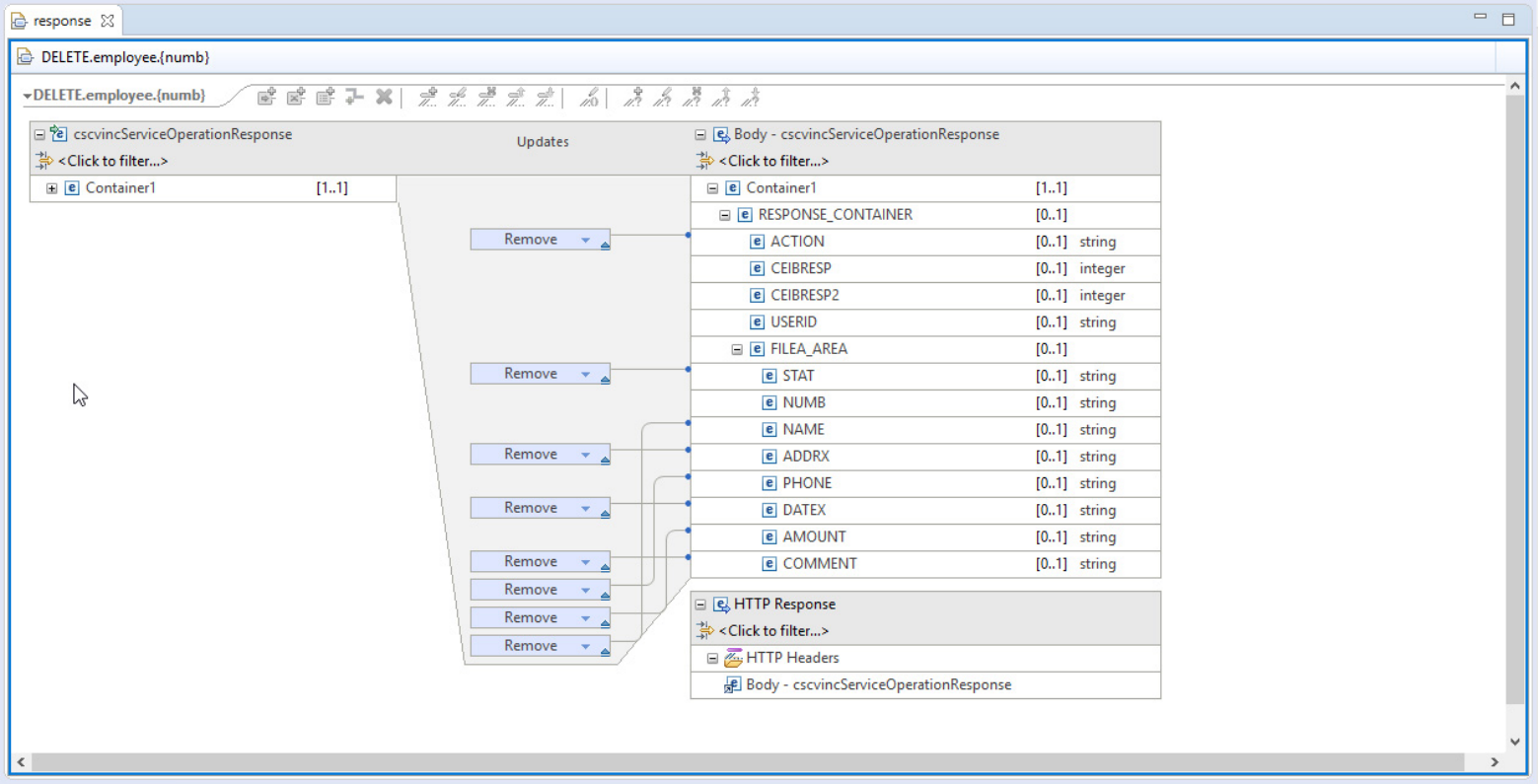


mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

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

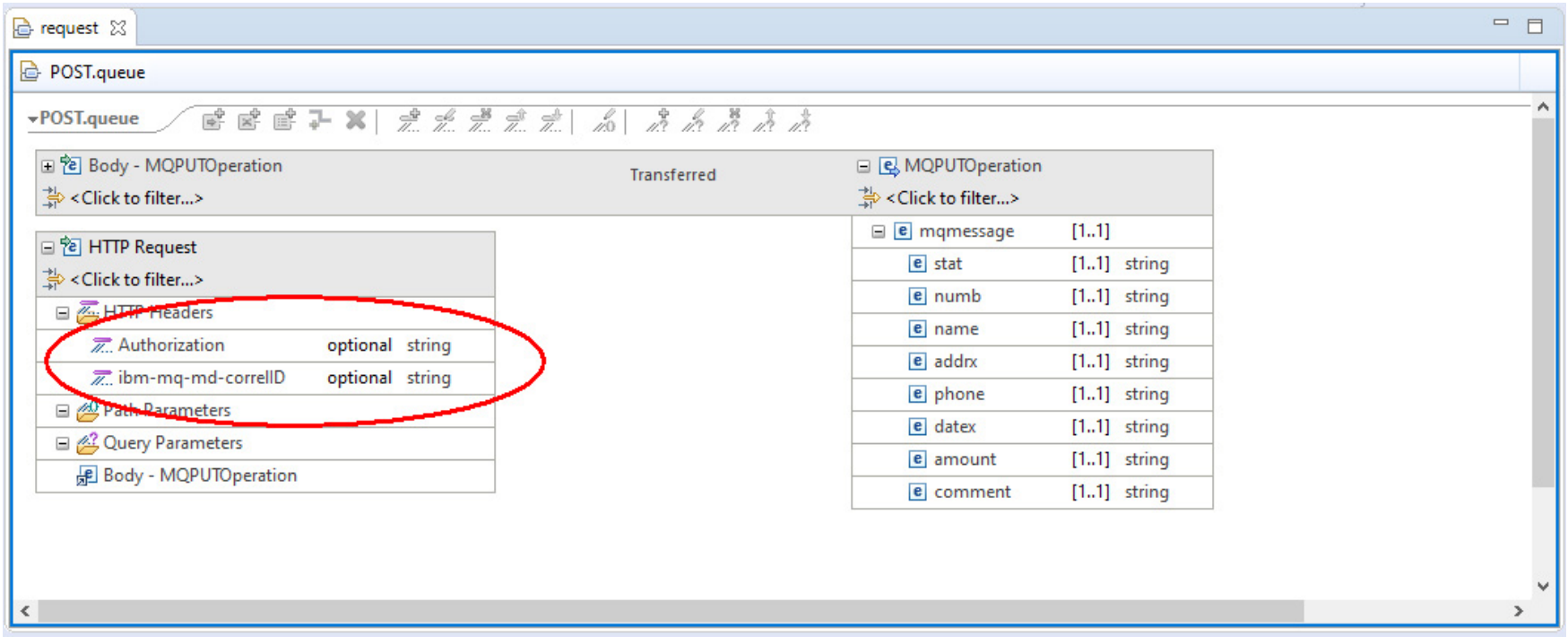


mitchj@us.ibm.com

API toolkit – API Editor

z/OS Connect EE

API mapping: Allows adding HTTP header properties

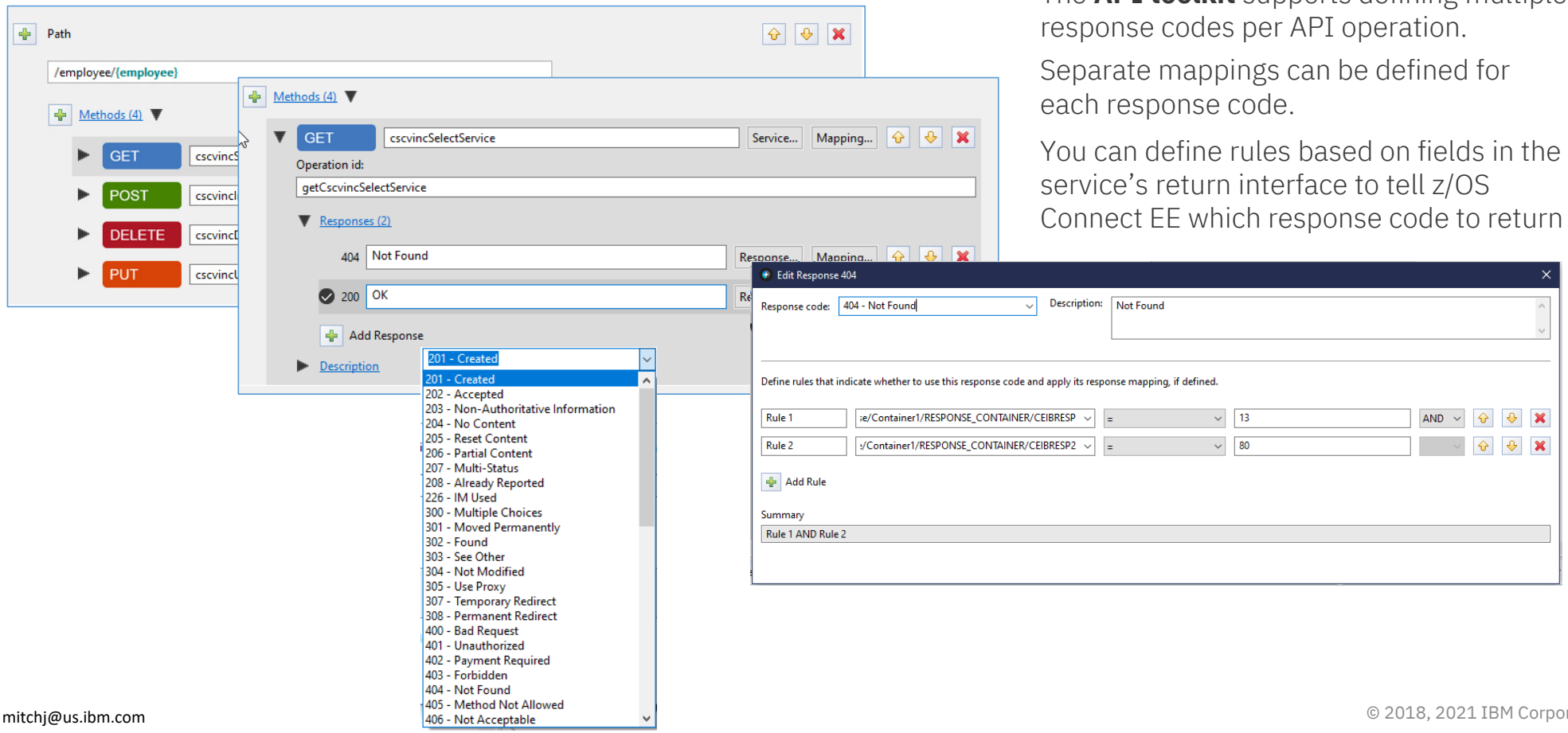


mitchj@us.ibm.com

API toolkit

z/OS Connect EE

API mapping: API definition with multiple response codes



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

Separate mappings can be defined for each response code.

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

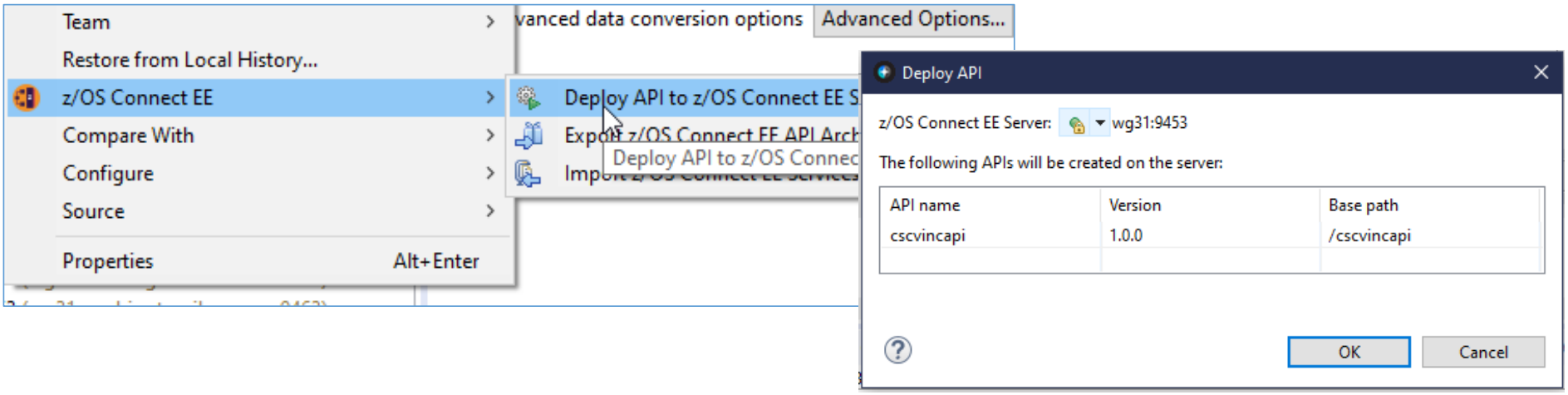
mitchj@us.ibm.com

API toolkit – API Editor

Server connection and API deployment

z/OS Connect EE

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



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

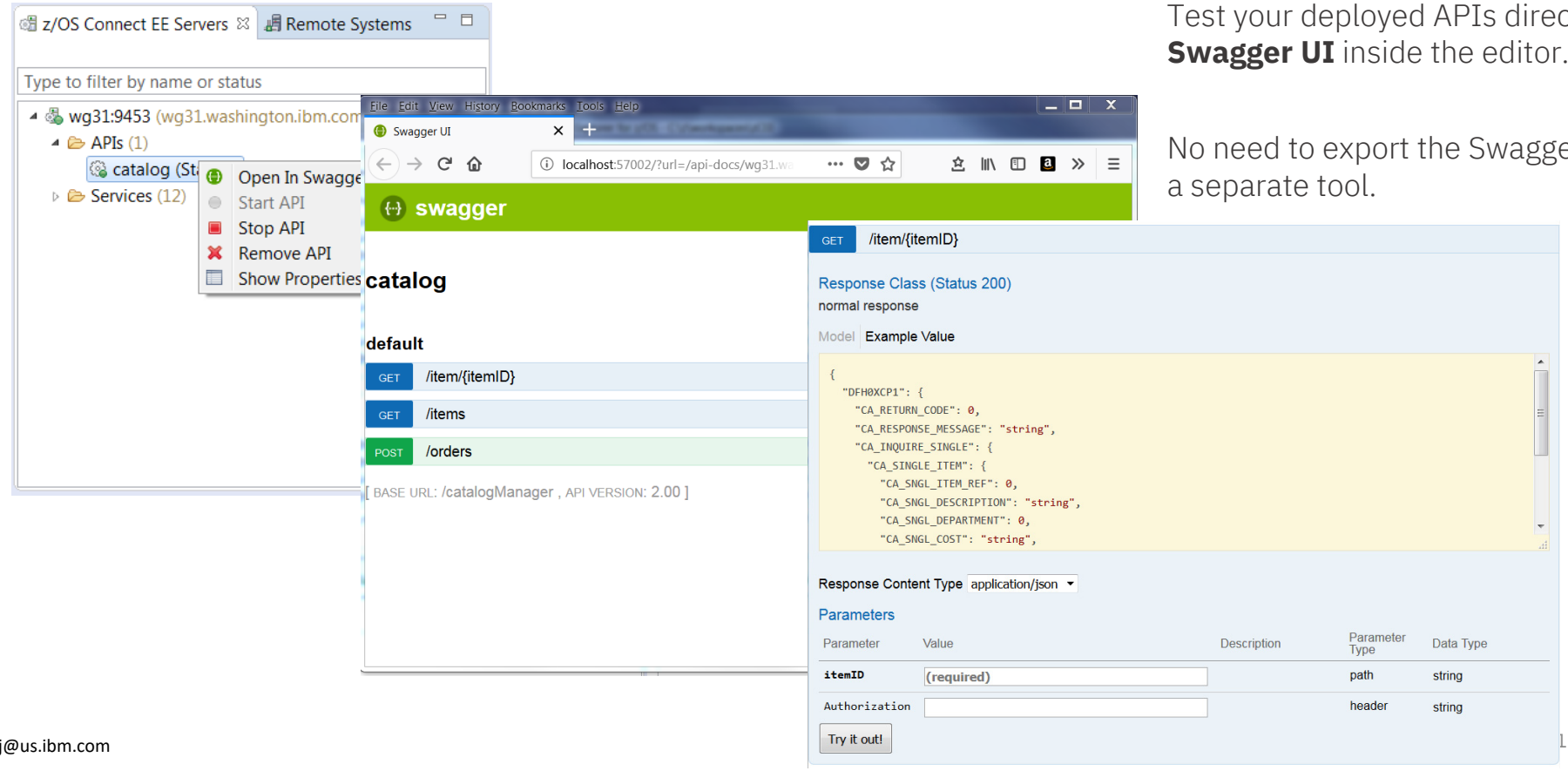
z/OS Connect EE servers view allows you to start, stop, and remove APIs from a running server.

mitchj@us.ibm.com

API toolkit – API Editor

Testing with Swagger UI

z/OS Connect EE



Test your deployed APIs directly with **Swagger UI** inside the editor.

No need to export the Swagger doc to a separate tool.

mitchj@us.ibm.com

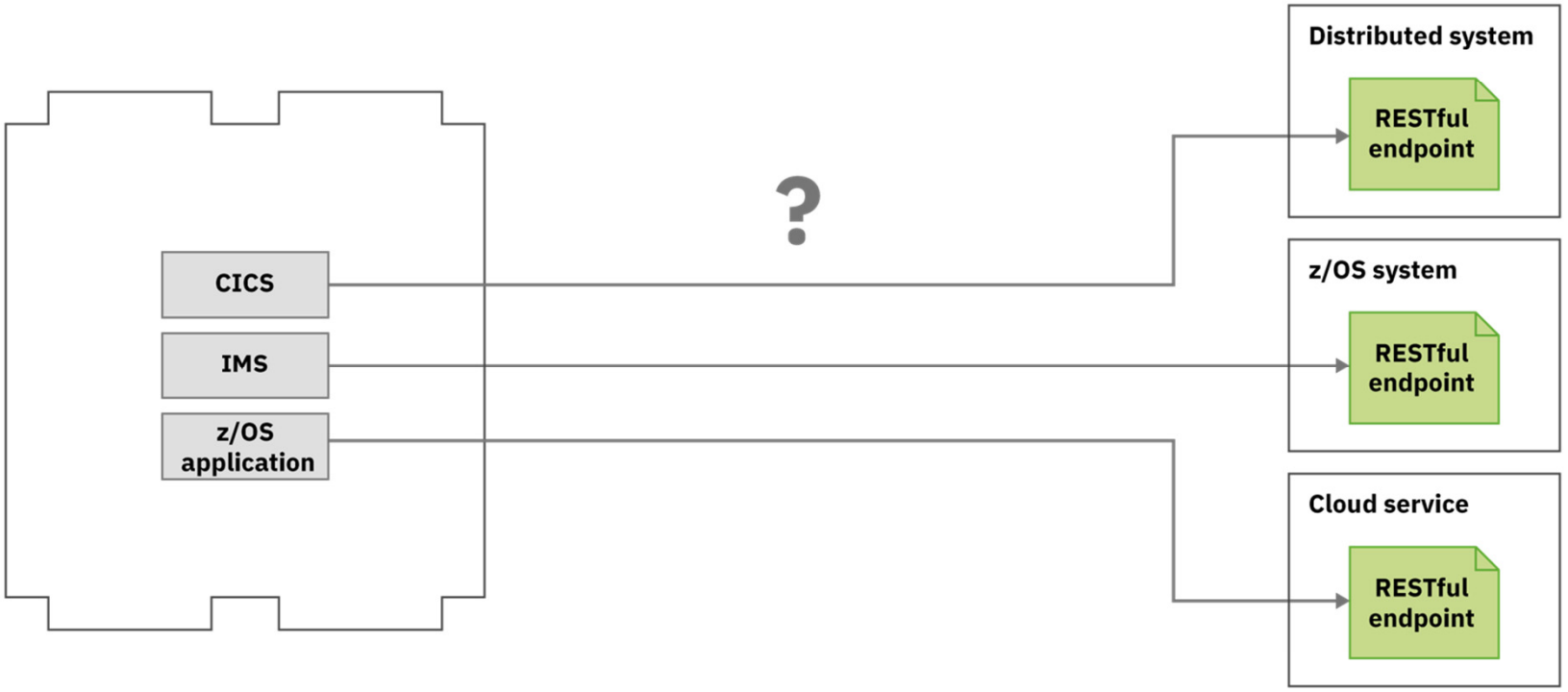
/api_toolkit/apiRequesters

Quick and easy **API mapping.**

mitchj@us.ibm.com

What about calling external APIs from my z/OS assets?

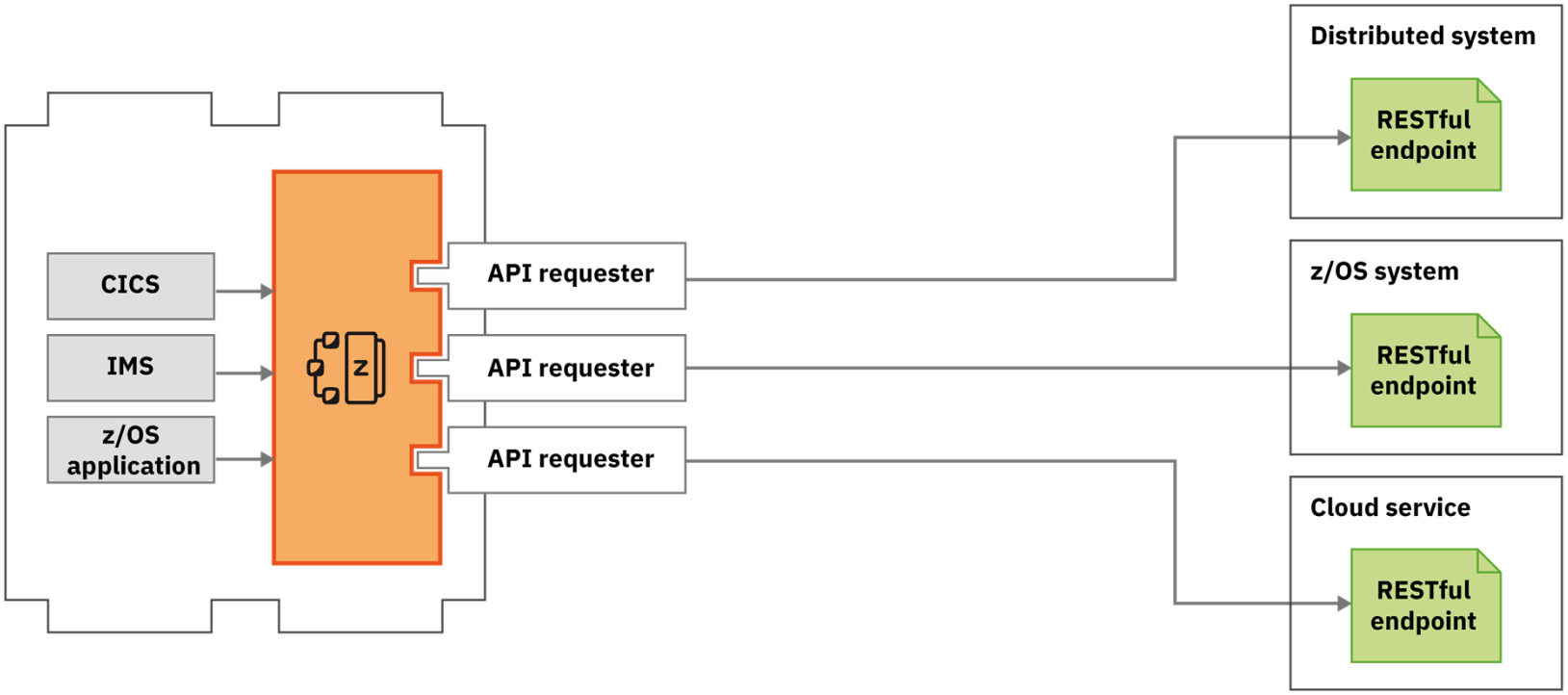
z/OS Connect EE



mitchj@us.ibm.com

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

z/OS Connect EE

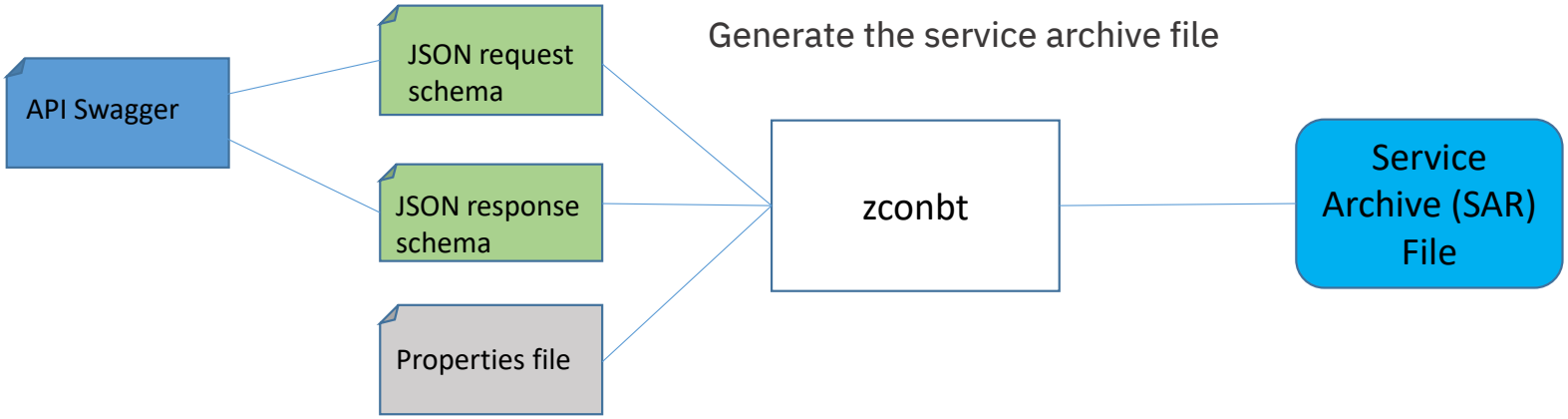


mitchj@us.ibm.com

Creating Services using zconbt – REST

z/OS Connect EE

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



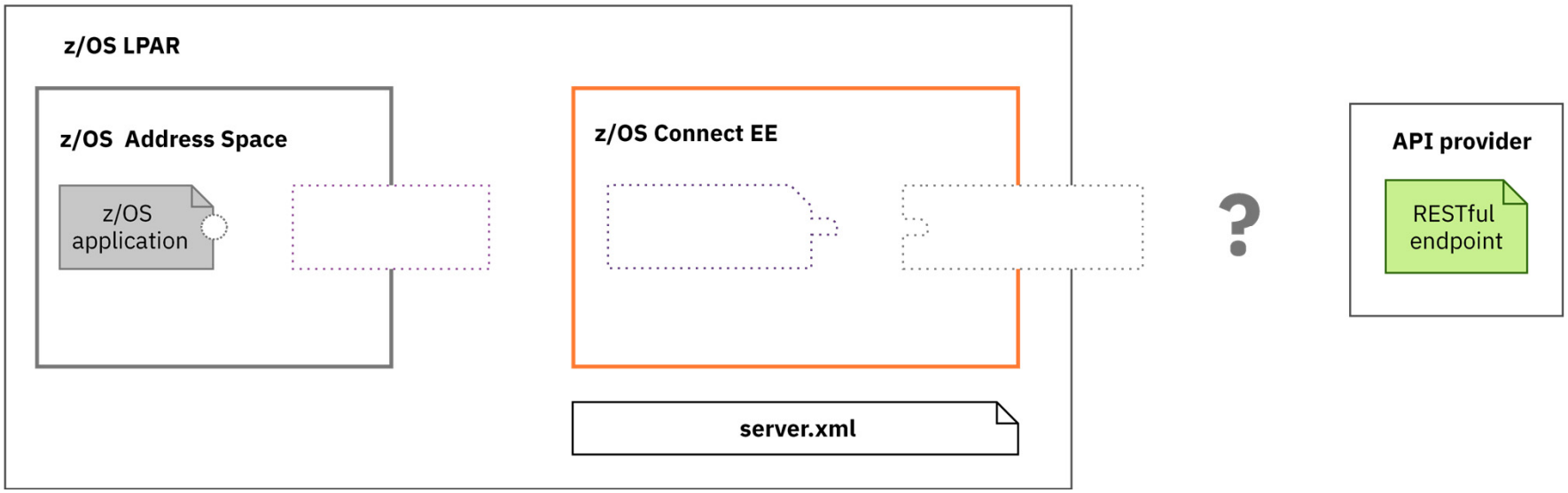

```
provider=rest
name=getCompany
version=1.0
description=Obtain a list of companies
requestSchemaFile=getCompanyRequest.json
responseSchemaFile=getCompanyResponse.json
verb=POST
uri=/Trader/rest/GetCompany
connectionRef=HatsConn
```

mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Starting point

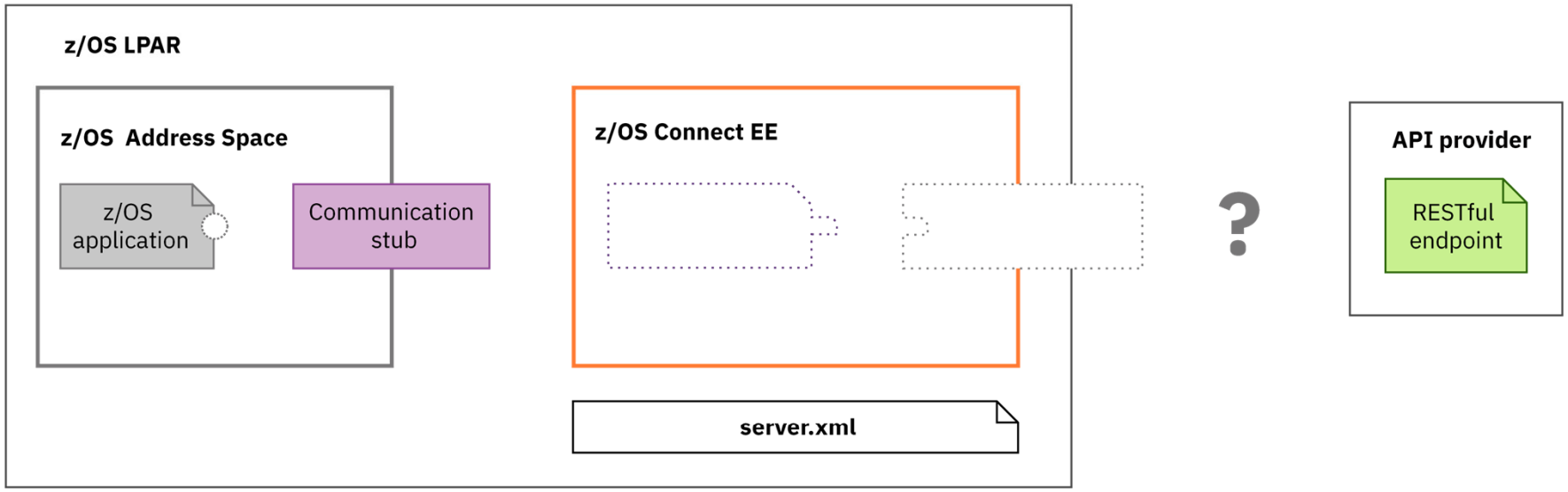


mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Step 1. Configure communication stub



Configure a communication stub.

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

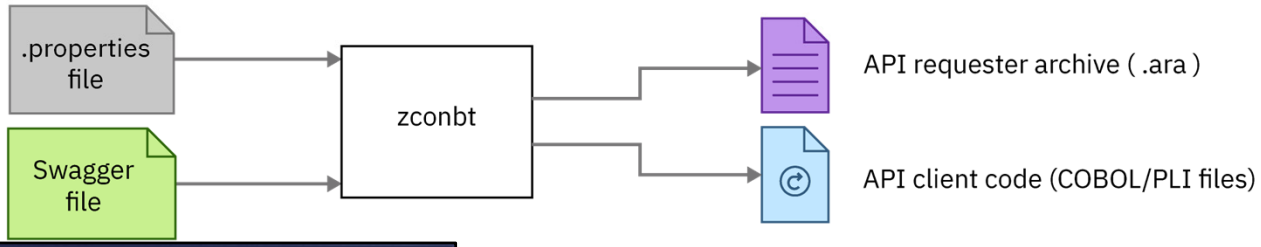
ibm.biz/zosconnect-configure-comms-stub

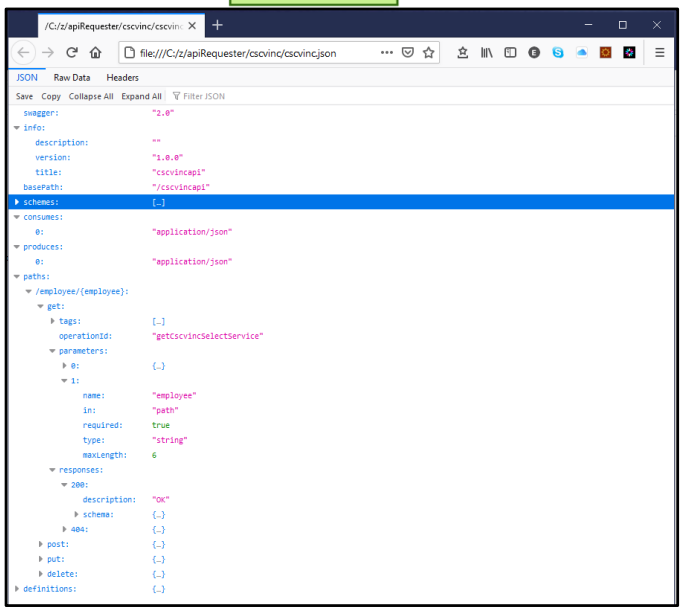
mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Step 2. Generate API requester archive and API client code from Swagger





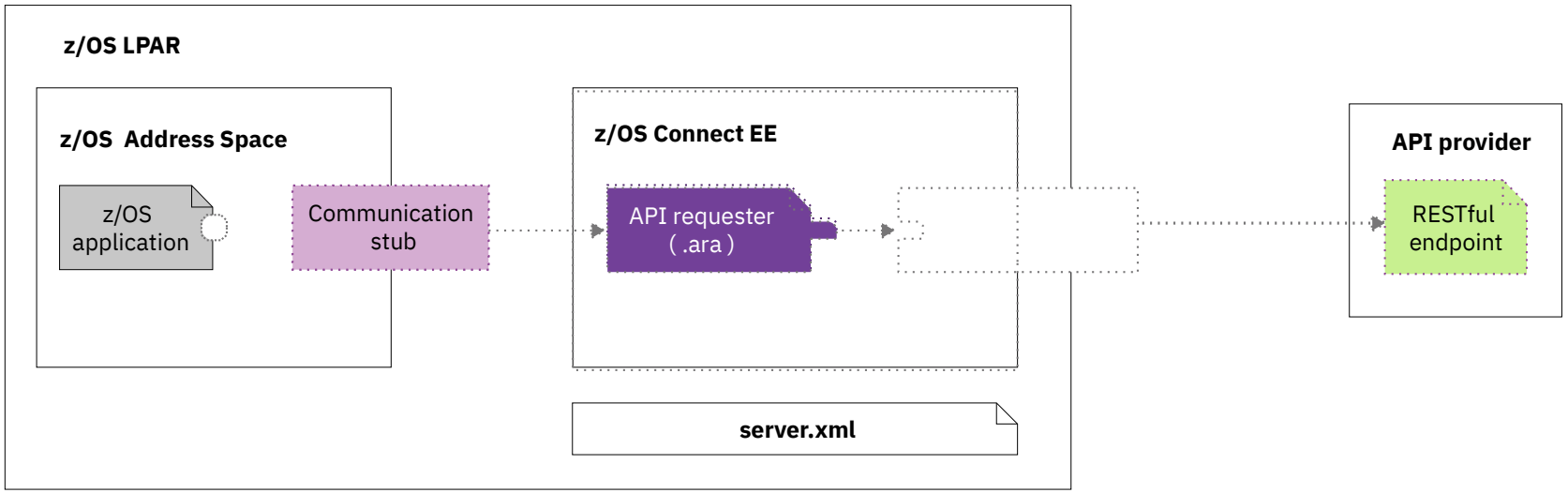

```
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Step 3. Deploy API requester (.ara) archive



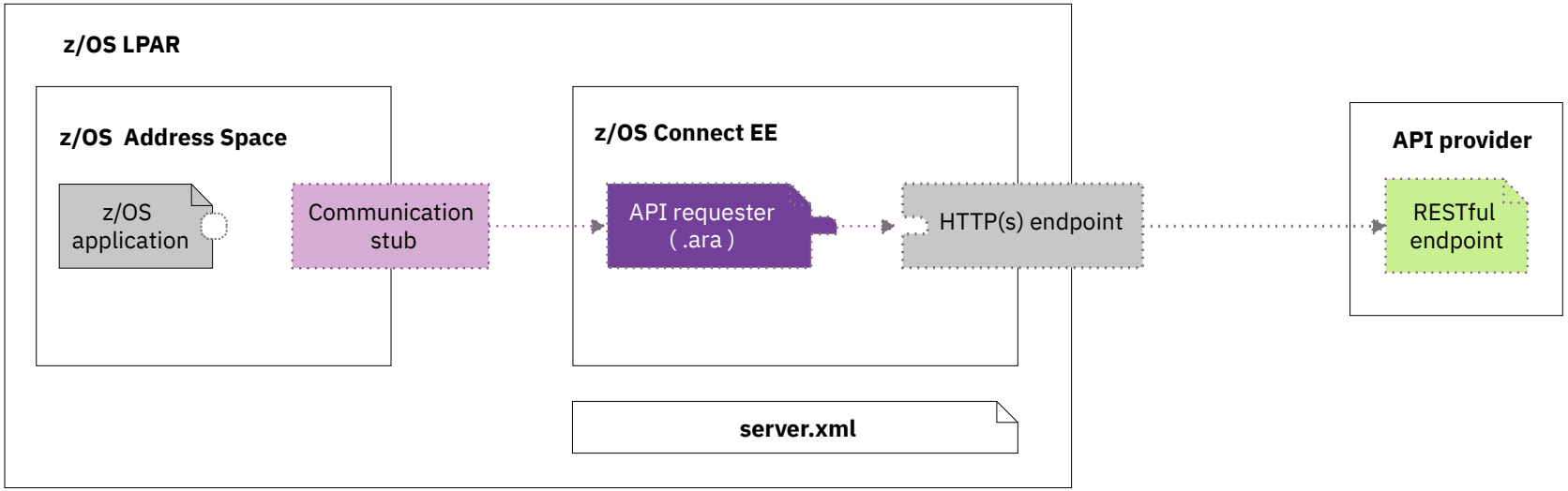
Deploy your API requester archive to the *apiRequesters* directory.

mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Step 4. Configure HTTP(S) endpoint



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

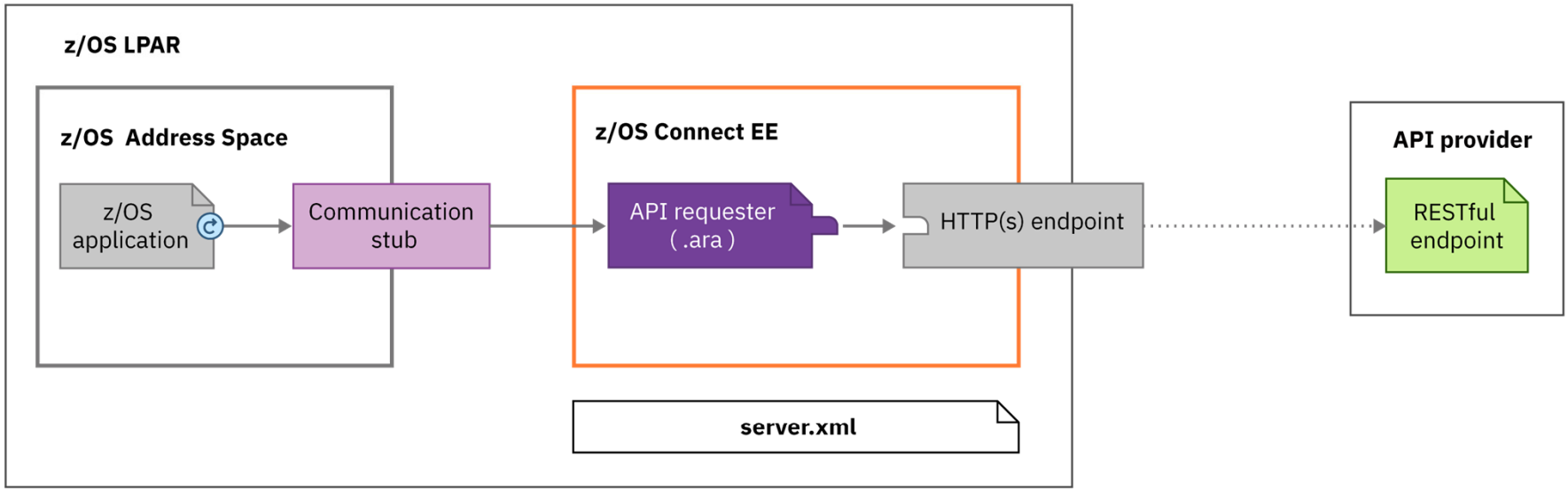
ibm.biz/zosconnect-configure-endpoint-connection

mitchj@us.ibm.com

Steps to calling an external API

z/OS Connect EE

Step 5. Update z/OS application



Finally, add the generated API client code to your existing application and use it to make the external API call.

ibm.biz/zosconnect-configure-requester-zos-application

mitchj@us.ibm.com

Steps to calling an external API

Step 5a. Update the z/OS application to include new copy books

GETAPI

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

      * Copy API Requester required copybook
       COPY BAQRINFO.

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

      * Request and Response segm
```

apirs.xml

```
<server description="API Requester">
    <!-- Enable features -->
    <featureManager>
        <feature>zosconnect:apiRequester-1.0</feature>
    </featureManager>

<zosconnect_apiRequesters location="">
        <zosconnect_apiRequester name="cscvinc_1.0.0"/>
</zosconnect_apiRequesters>

<zosconnect_endpointConnection id="cscvincAPI"
        host="http://wg31.washington.ibm.com"
        port="9120"
        basicAuthRef="myBasicAuth"
        connectionTimeout="10s"
        receiveTimeout="20s" />

<zosconnect_authData id="myBasicAuth"
        user="Fred"
        password="fredpwd" />

</server>
```

CSC02I01

```
           03 BAQ-APINAME                PIC X(255)
              VALUE 'cscvinc_1.0.0'.
           03 BAQ-APINAME-LEN            PIC S9(9) COMP-5 SYNC
              VALUE 13.
           03 BAQ-APIPATH                PIC X(255)
              VALUE '/cscvinc/employee/{numb}'.
           03 BAQ-APIPATH-LEN            PIC S9(9) COMP-5 SYNC
              VALUE 24.
           03 BAQ-APIMETHOD              PIC X(255)
              VALUE 'GET'.
           03 BAQ-APIMETHOD-LEN          PIC S9(9) COMP-5 SYNC
              VALUE 3.
```

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

mitchj@us.ibm.com

Steps to calling an external API

Step 5b. Update the z/OS application to call the stub

GETAPI

```
      *---------------------------------------------------------------*
      * Set up the data for the API Requester call                    *
      *---------------------------------------------------------------*
           MOVE numb     of PARM-DATA TO numb IN API-REQUEST.
           MOVE LENGTH of numb in API-REQUEST to
               numb-length IN API-REQUEST.

      *---------------------------------------------------------------*
      * Initialize API Requester PTRs & LENs                          *
      *---------------------------------------------------------------*
      * Use pointer and length to specify the location of
      *  request and response segment.
      * This procedure is general and necessary.
           SET BAQ-REQUEST-PTR TO ADDRESS OF API-REQUEST.
           MOVE LENGTH OF API-REQUEST TO BAQ-REQUEST-LEN.
           SET BAQ-RESPONSE-PTR TO ADDRESS OF API_RESPONSE.
           MOVE LENGTH OF API_RESPONSE TO BAQ-RESPONSE-LEN.

      *---------------------------------------------------------------*
      * Call the communication stub                                   *
      *---------------------------------------------------------------*
      * Call the subsystem-supplied stub code to send
      * API request to zCEE
           CALL COMM-STUB-PGM-NAME USING
                BY REFERENCE    API-INFO-OPER1
                BY REFERENCE    BAQ-REQUEST-INFO
                BY REFERENCE    BAQ-REQUEST-PTR
                BY REFERENCE    BAQ-REQUEST-LEN
                BY REFERENCE    BAQ-RESPONSE-INFO
                BY REFERENCE    BAQ-RESPONSE-PTR
                BY REFERENCE    BAQ-RESPONSE-LEN.
      * The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
```

mitchj@us.ibm.com

Steps to calling an external API

Step 5c. Update the z/OS application to access the results

GETAPI

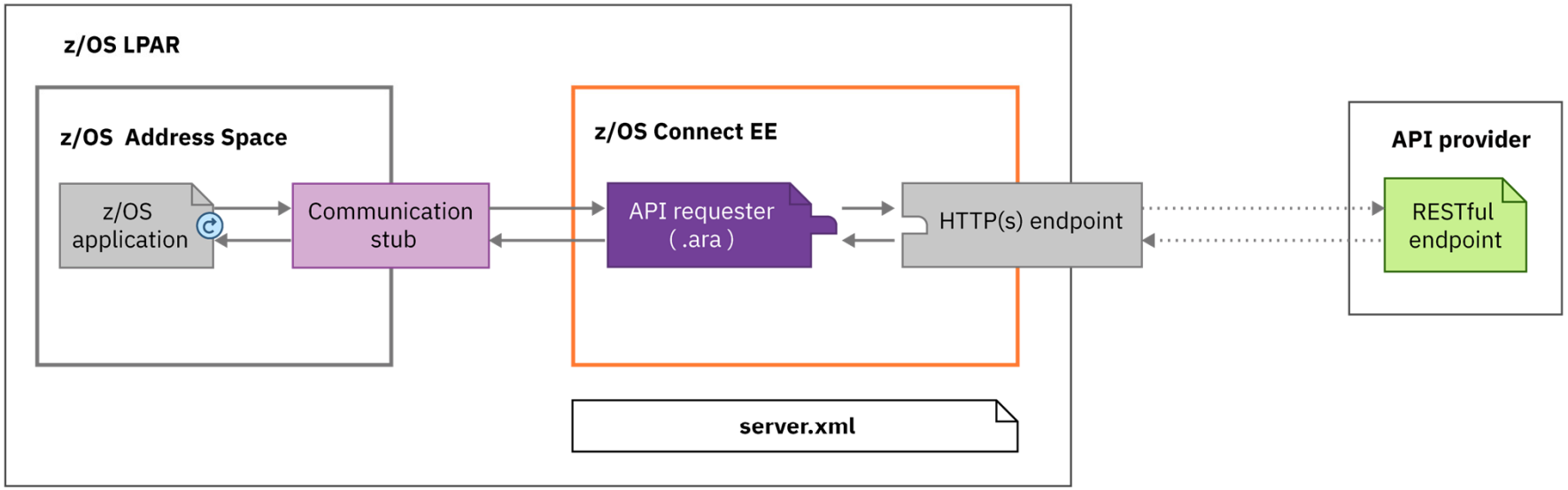
```
                BY REFERENCE   BAQ-RESPONSE-LEN.
      * The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
      * API call is successful.

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

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

mitchj@us.ibm.com

Steps to calling an external API

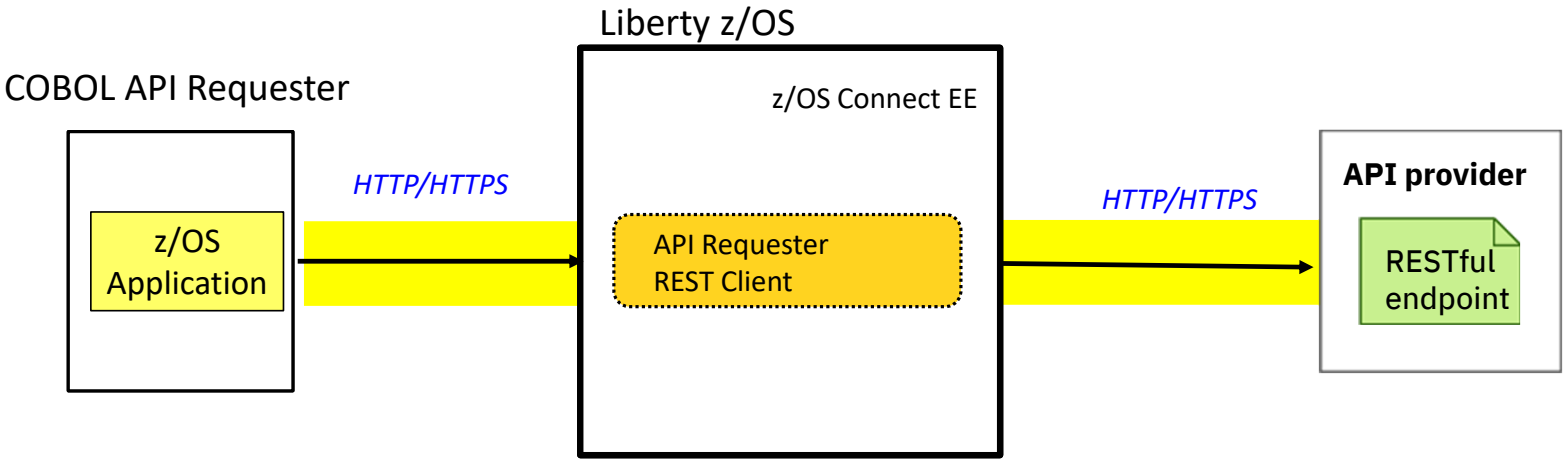
z/OS Connect EE

Done



mitchj@us.ibm.com

API requester to API Provider connection overview



MVS Batch and IMS HTTP connection details provided by:

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

CICS HTTP connection details provided by:

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

mitchj@us.ibm.com

/common_scenarios

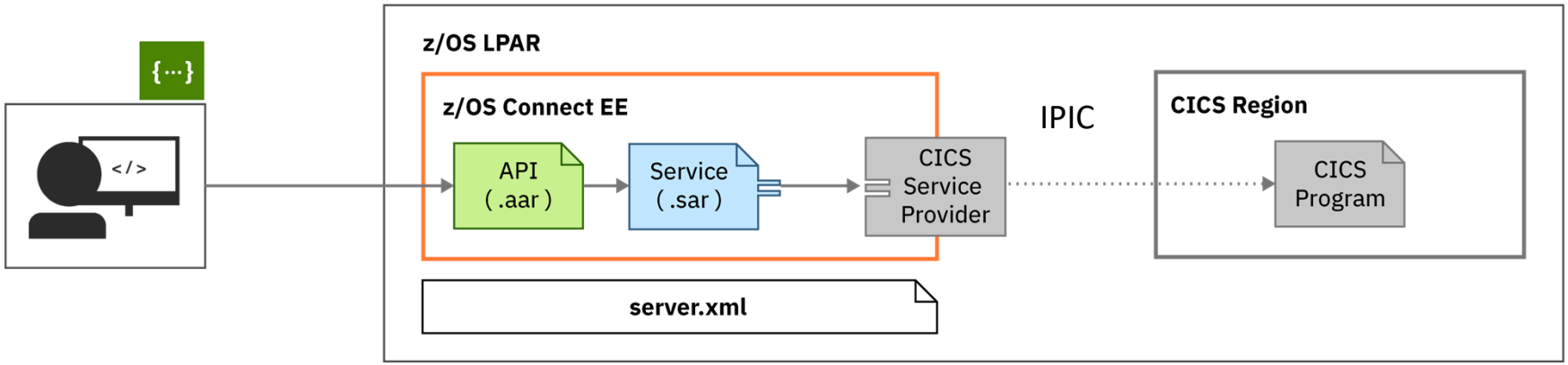
Typical connection patterns to different subsystems.

mitchj@us.ibm.com

Connections to CICS

z/OS Connect EE

Topology



Connection to CICS is configured in `server.xml`.

An IPIC connection must be configured in CICS.

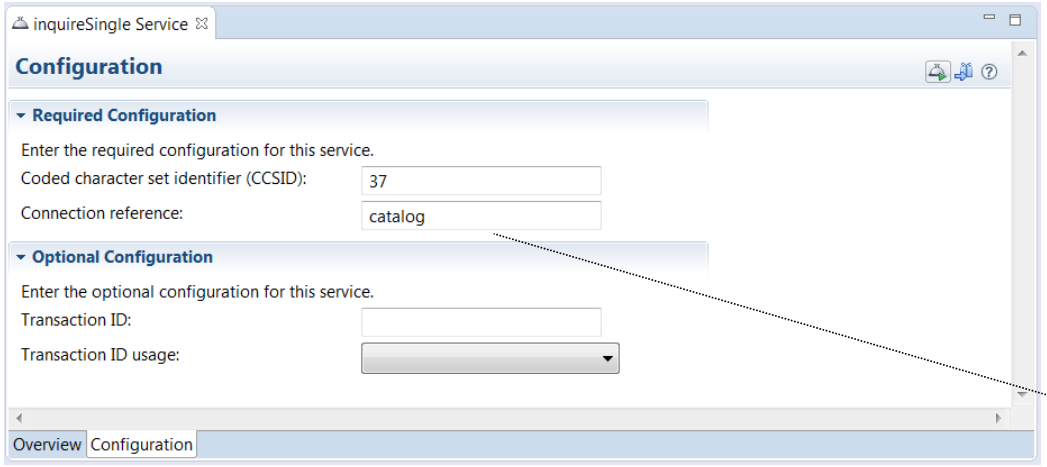
ibm.biz/zosconnect-scenarios

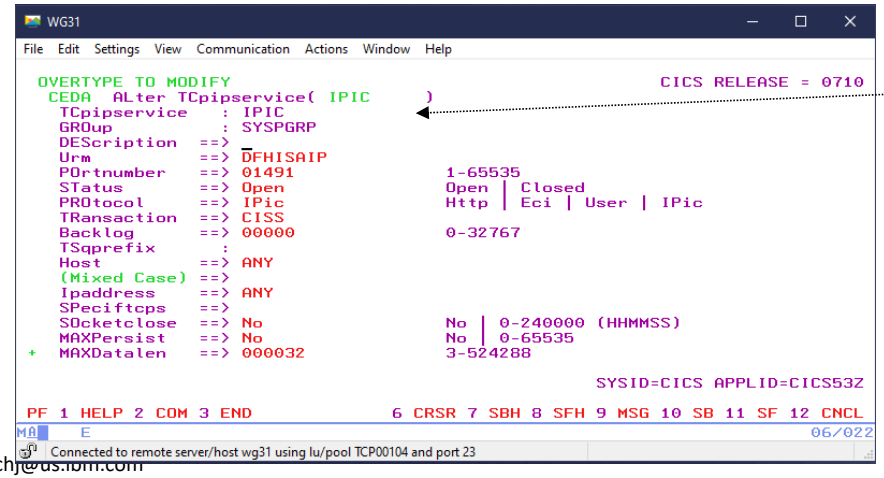
mitchj@us.ibm.com

CICS IPIC (server.xml)

z/OS Connect EE

The server.xml file is the key configuration file:





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

catalog.xml

Design Source

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

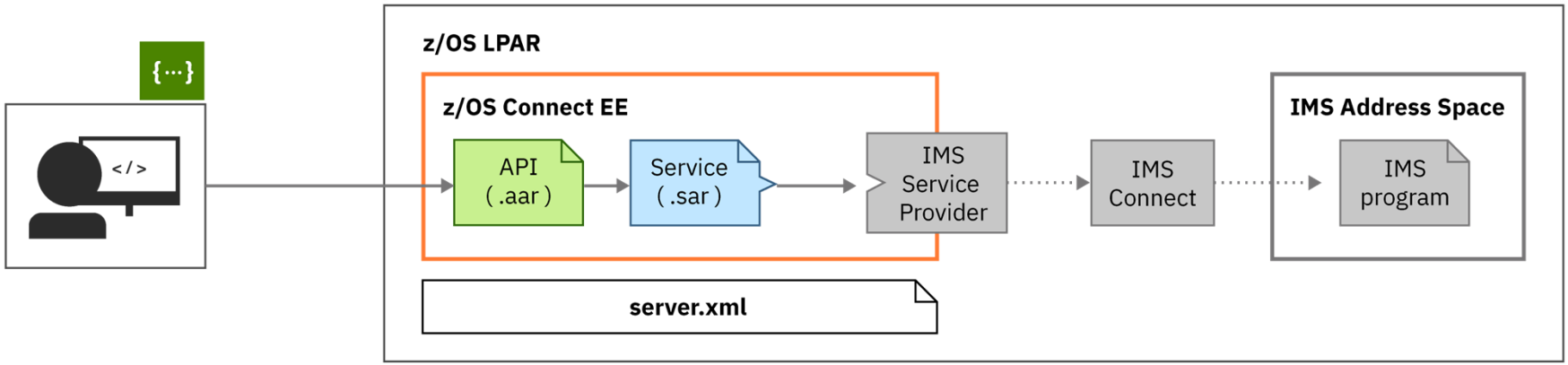
Define IPIC connection to CICS

mitchj@us.ibm.com

Connections to IMS TM

z/OS Connect EE

Topology



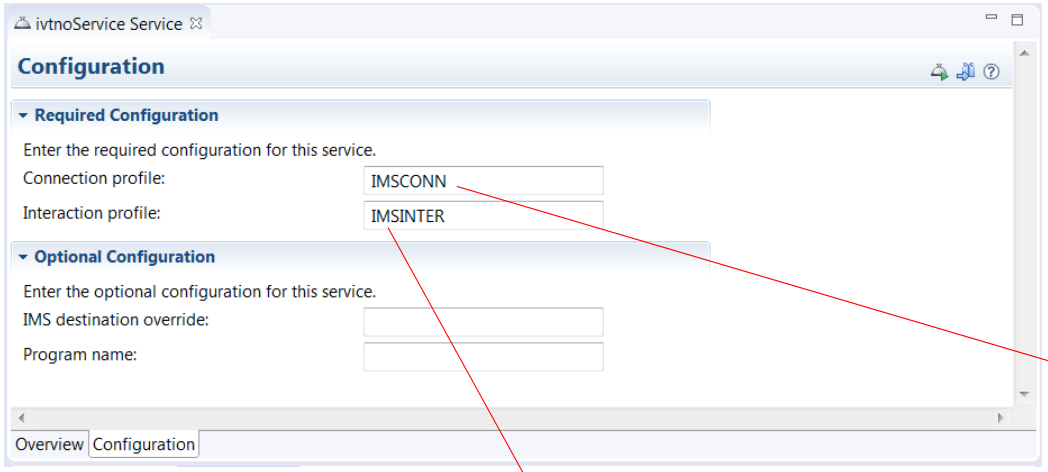
Configure the connection to IMS through `ims-connections.xml` and `ims-interactions.xml` in the IMS service registry.

ibm.biz/zosconnect-scenarios

mitchj@us.ibm.com

IMS Connections and Interactions

z/OS Connect EE



IMS Connect HWSCFG

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

Connection

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

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

Interaction

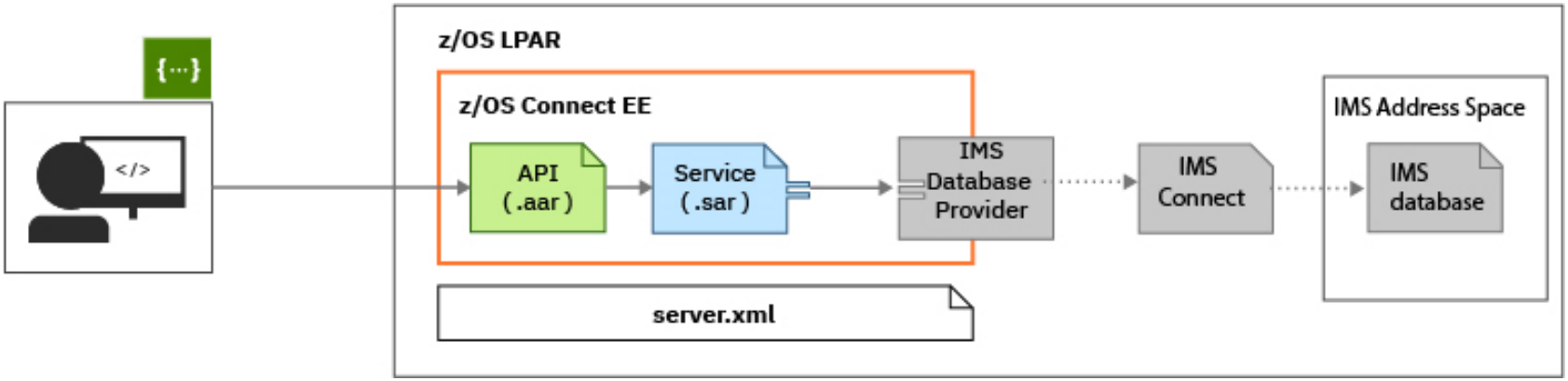
```
<server>
<imsmobile_interaction comment="" commitMode="1" id="IMSINTER" imsConnectCodepage="Cp1047" imsConnectTimeout="0"
          imsDatastoreName="IVP1" interactionTimeout="-1" ltermOverrideName="" syncLevel="0"/>
</server>
```

mitchj@us.ibm.com

Connections to IMS DB

z/OS Connect EE

Topology



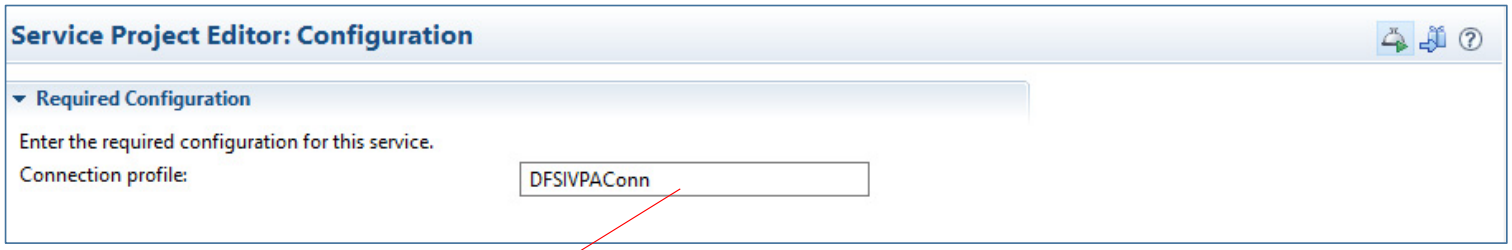
Configure the connection to IMS using a Connection Factory in server.xml

Use the **API toolkit** to configure the service.

mitchj@us.ibm.com

IMS Connection Factory

z/OS Connect EE



ConnectionFactory

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

IMS Connect HWSCFG

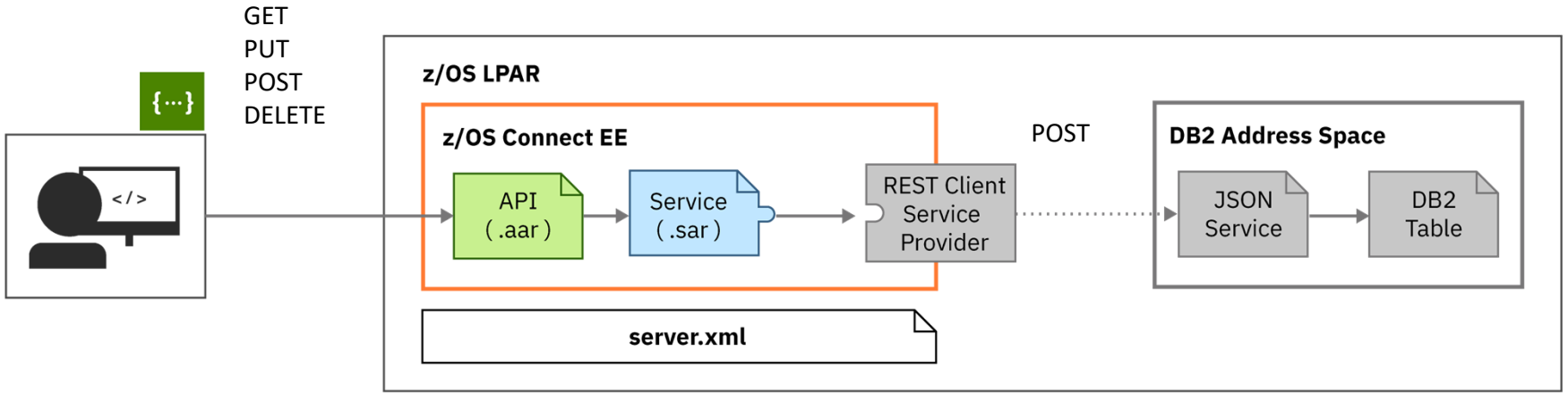
```
HWS=(ID=IMS14HWS,XIBAREA=100,RACF=N,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)
DATASTORE=(GROUP=OTMAGRP,ID=IVP1,MEMBER=HWSMEM,TMEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS,TMEMBER=PLEX1)
ODACCESS=(ODBMAUTOCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000),ODBMTMOT=6000)
```


mitchj@us.ibm.com

Connections to Db2

z/OS Connect EE

Topology



Connection to the JSON Service is configured in `server.xml`.

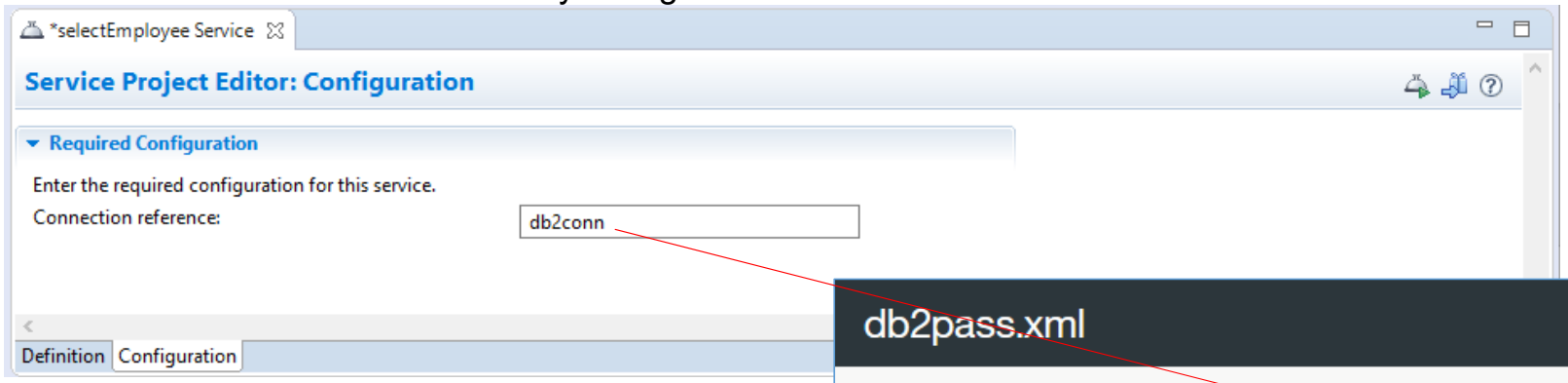
A Db2 REST Service must be configured in DB2.

mitchj@us.ibm.com

The server.xml File (Db2)

z/OS Connect EE

The server.xml file is the key configuration file:




```
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

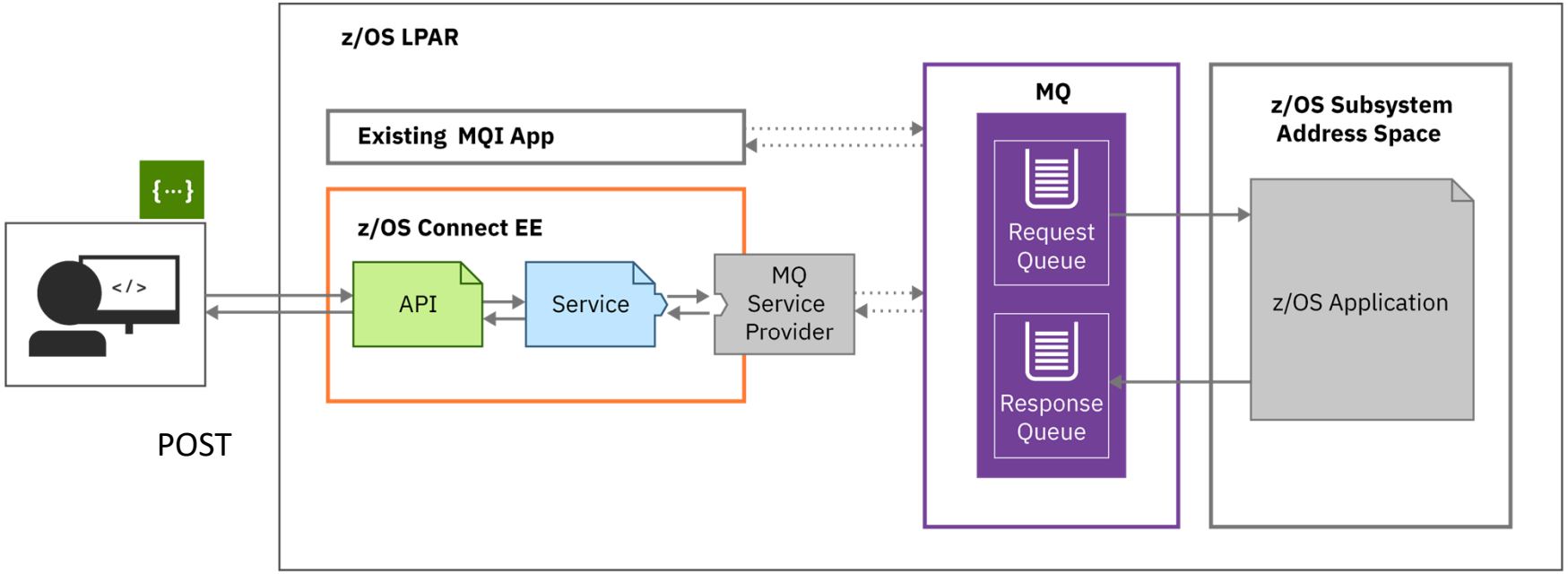
```
1  <server description="DB2 REST">
2
3   <zosconnect_zosConnectServiceRestClientConnection id="db2conn"
4         host="wg31.washington.ibm.com"
5         port="2446"
6         basicAuthRef="dsn2Auth" />
7
8   <zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
9         applName="DSN2APPL"/>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

mitchj@us.ibm.com

Connections to MQ

z/OS Connect EE

Topology (Two-way service example)



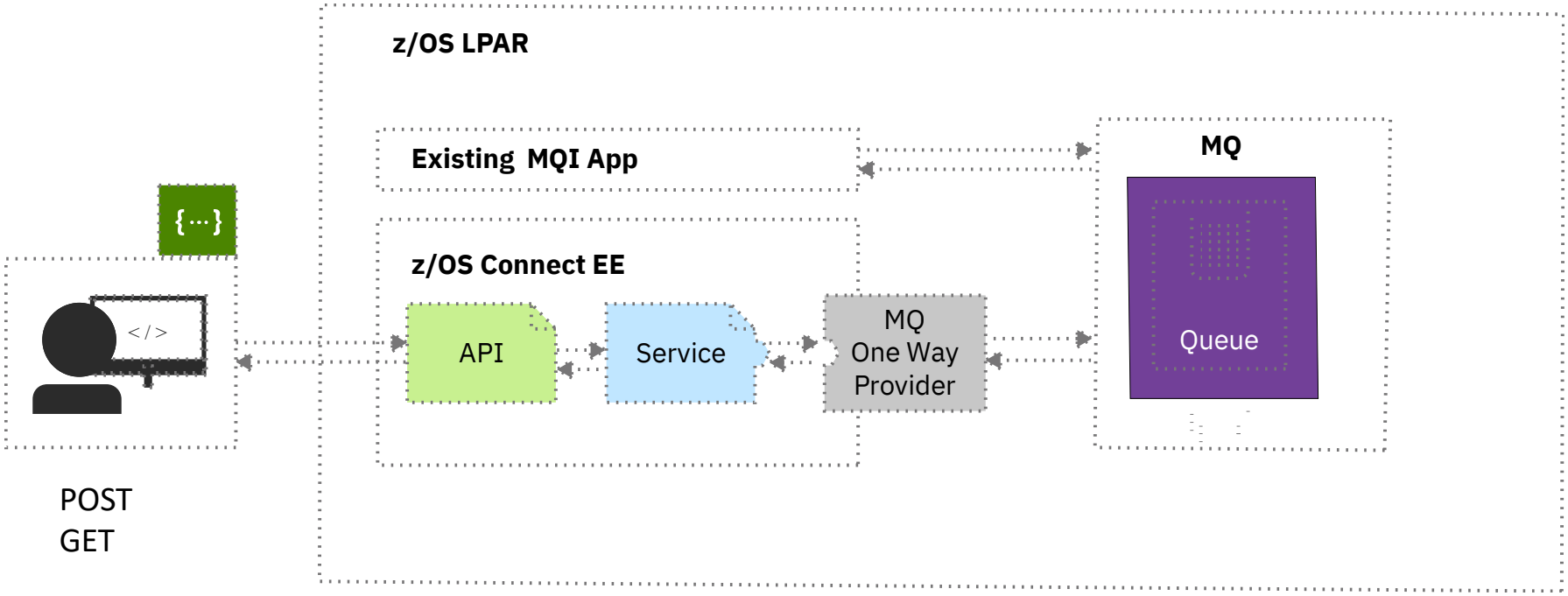
You can also configure one-way services.

mitchj@us.ibm.com

Connections to MQ

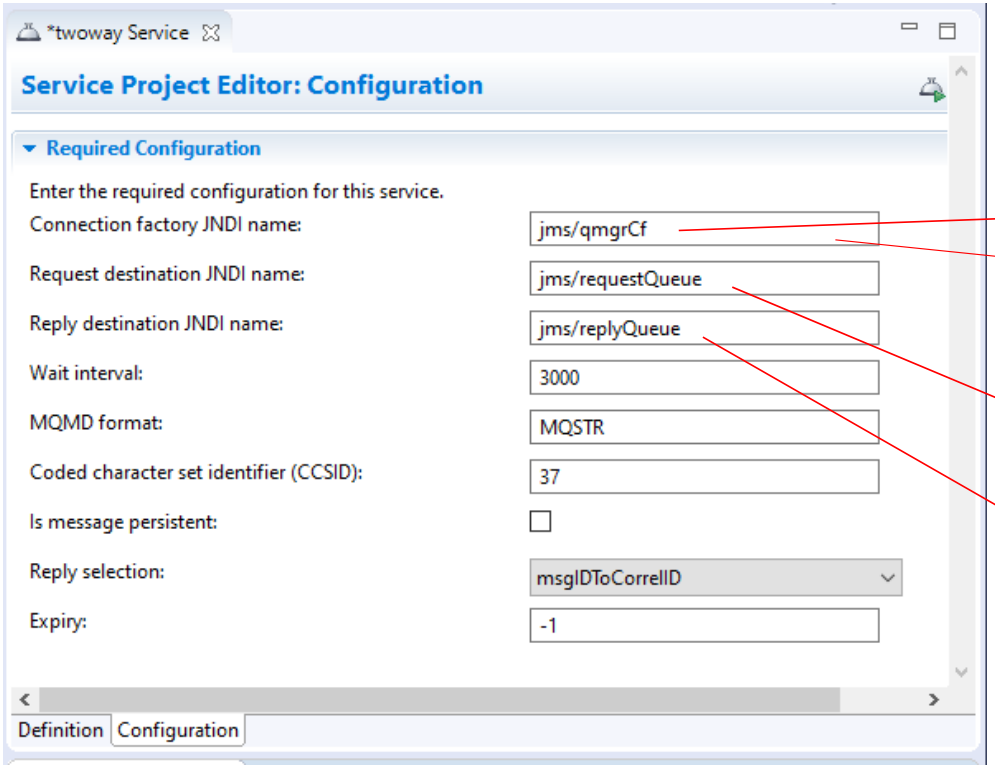
z/OS Connect EE

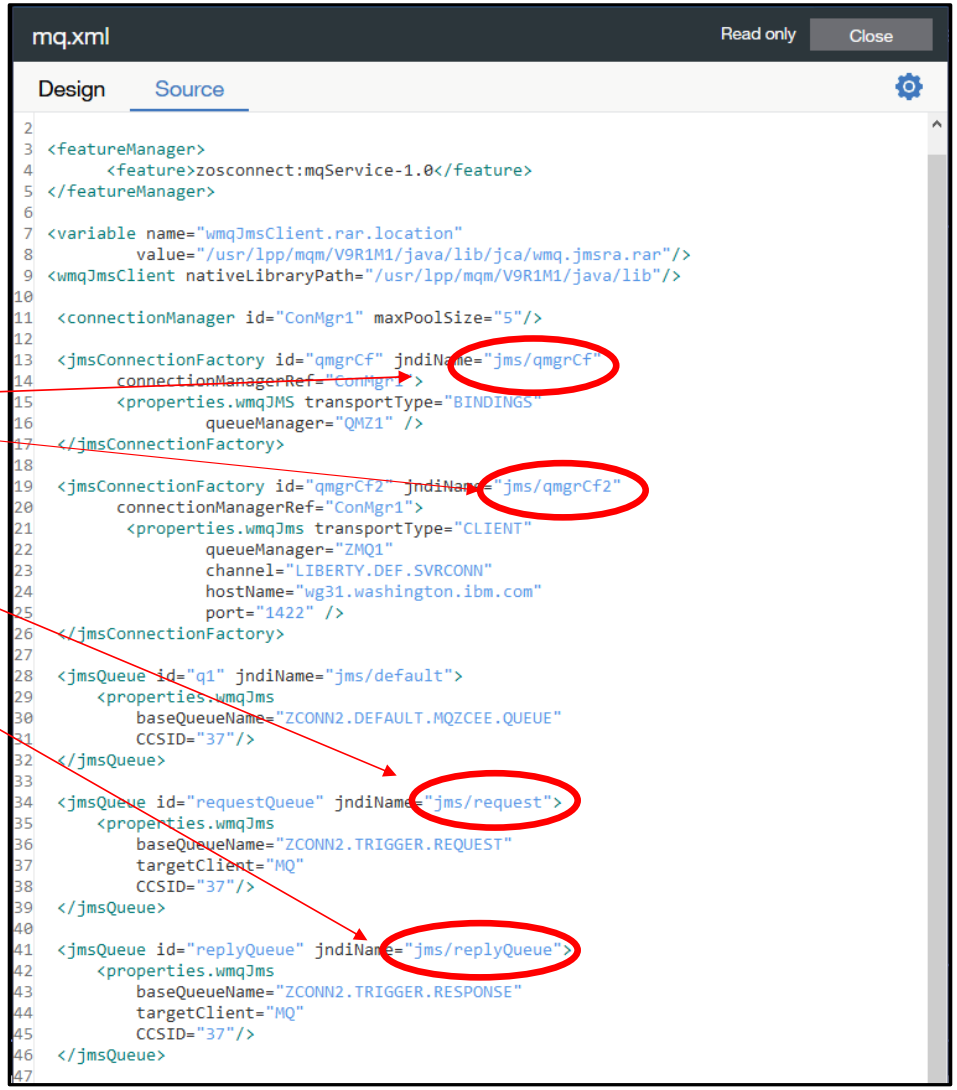
Topology (One-way service example)



mitchj@us.ibm.com

The server.xml File (MQ)



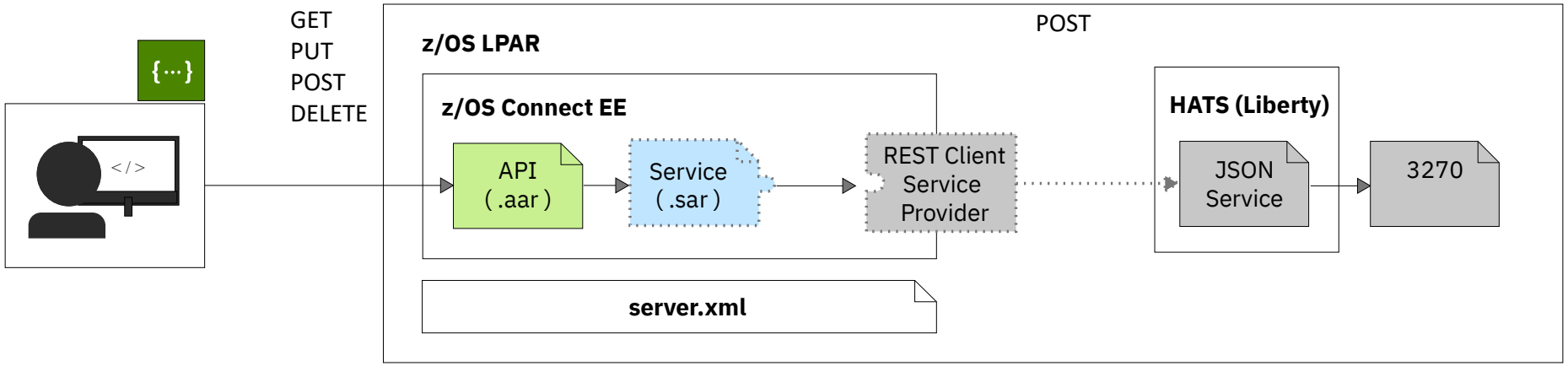


mitchj@us.ibm.com

Connection to HATS

z/OS Connect EE

Topology



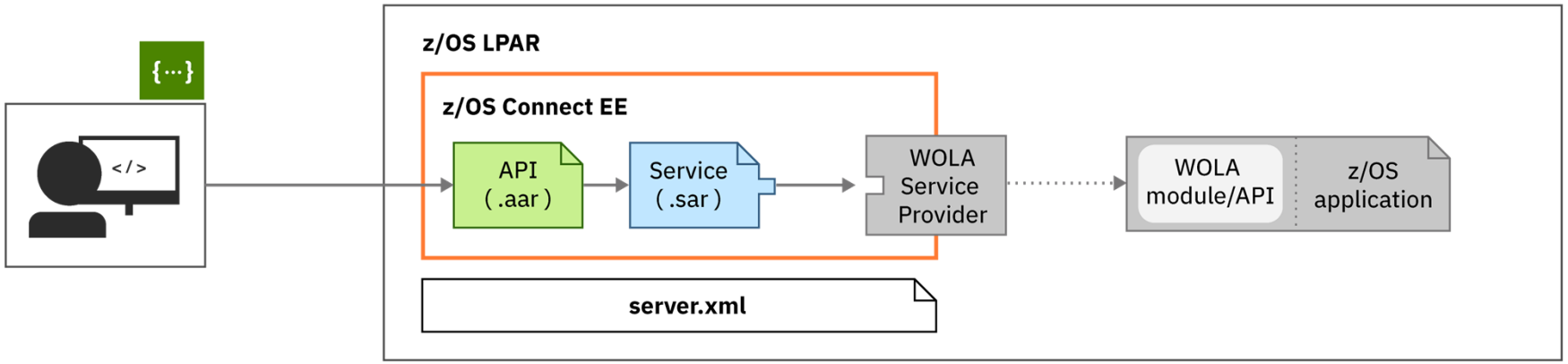
Connection to the HATS REST Service is configured in `server.xml`.

mitchj@us.ibm.com

Connections to a MVS batch application

z/OS Connect EE

Topology



Connection to WOLA is configured in `server.xml`.

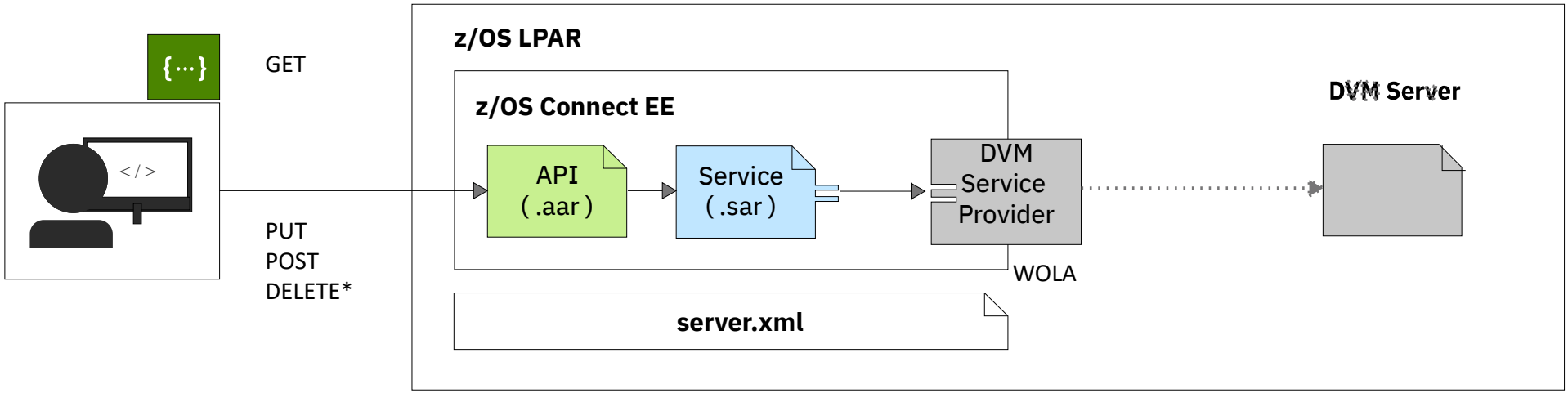
The z/OS application must be WOLA-enabled.

mitchj@us.ibm.com

Connections to DVM

z/OS Connect EE

Topology



The DVM service provider uses WOLA

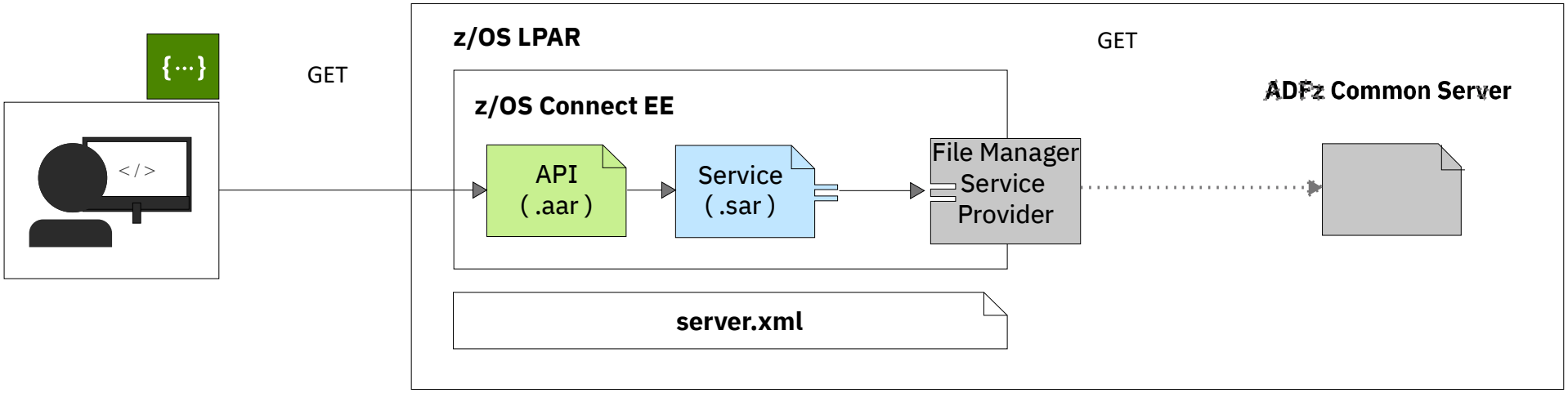
* Requires a resource manager (e.g. RLS, VSAMCICS, etc.

mitchj@us.ibm.com

Connections to File Manager

z/OS Connect EE

Topology



Connection to the Application Delivery Foundation for z (ADFz) common server is over TCP/IP

A File Manager Template is required .

mitchj@us.ibm.com

/miscellaneousTopics

performance, high availability, Liberty

mitchj@us.ibm.com

A Tour of Server Configuration Directories and Files

z/OS Connect EE

A z/OS Connect EE V3.0 server configuration structure looks like this:

```
/var/zosconnect
  /servers
    /zceesrv1
      /logs
         messages.log
      /resources
        /zosconnect
         /apis
         /apiRequesters
         /rules
         /services
       server.xml
       server.env
      /workarea
```

The messages.log file is the key output file for messages about Liberty and the processing taking place in the Liberty server.

The /zosconnect directory is where we will place the deployed APIs, services, and API requester files

The server.xml file is the key configuration file. It is here that z/OS Connect EE V3.0 definitions go which define the essential backend connectivity.

server.xml

```
<server description="zCEE Server">
<include location="${server.config.dir}/includes/safSecurity.xml"/>
<include location="${server.config.dir}/includes/ipicIDProp.xml"/>
<include location="${server.config.dir}/includes/keyringOutboundMutual.xml”/>
<include location="${server.config.dir}/includes/groupAccess.xml"/>
<include location="${server.config.dir}/includes/shared.xml"/>
<include location="${server.config.dir}/includes/apiRequesterHTTPS.xml"/
<include location="${server.config.dir}/includes/imsDatabase.xml"/>
```

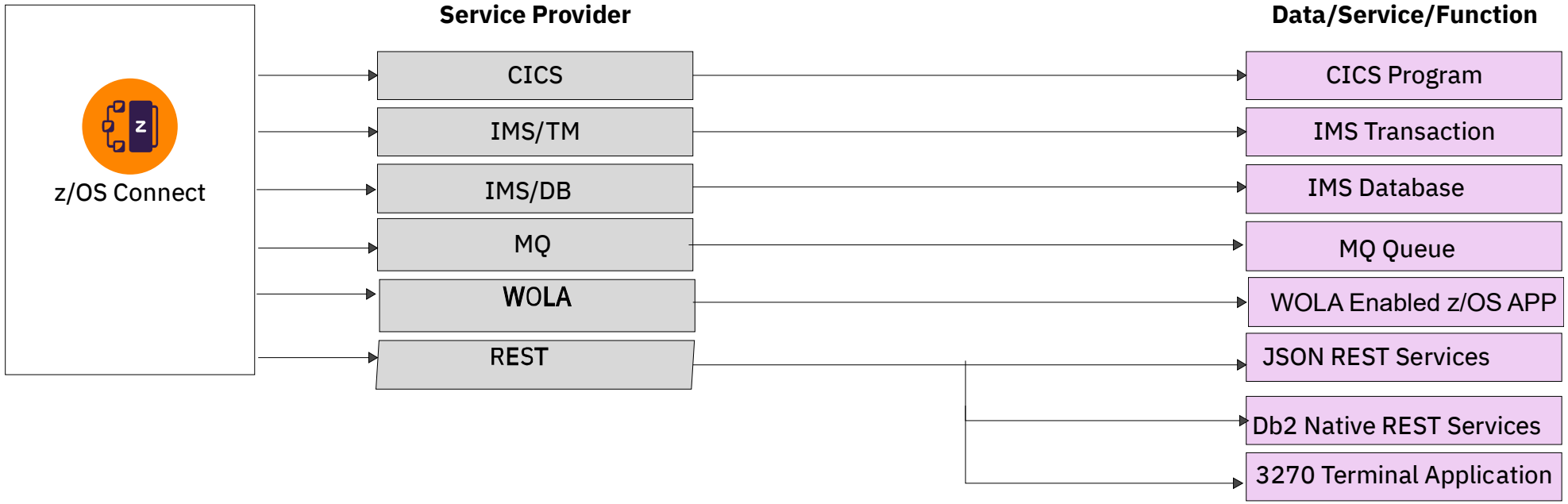
```
-Dcom.ibm.ws.logging.log.directory=/u/johnson/logs
```

mitchj@us.ibm.com

What assets can z/OS Connect EE map to?

z/OS Connect EE

And which service provider could I use?



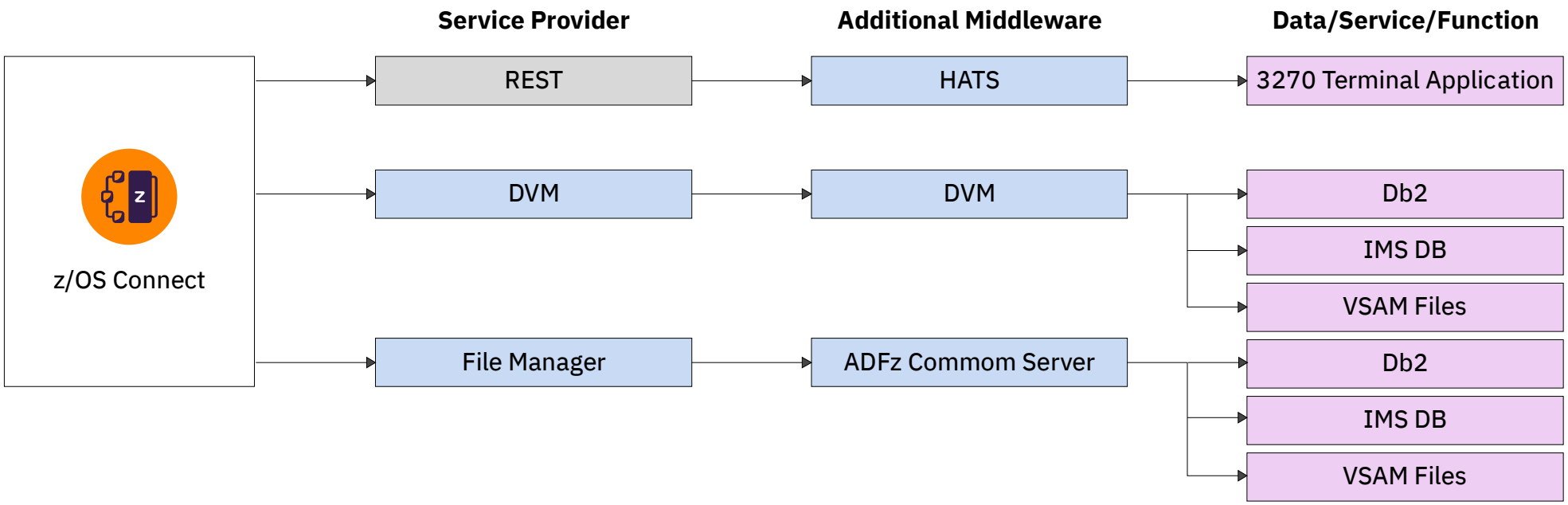
The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.

mitchj@us.ibm.com

Additional Middleware

Additional value from the ecosystem

z/OS Connect EE



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

z/OS Connect EE

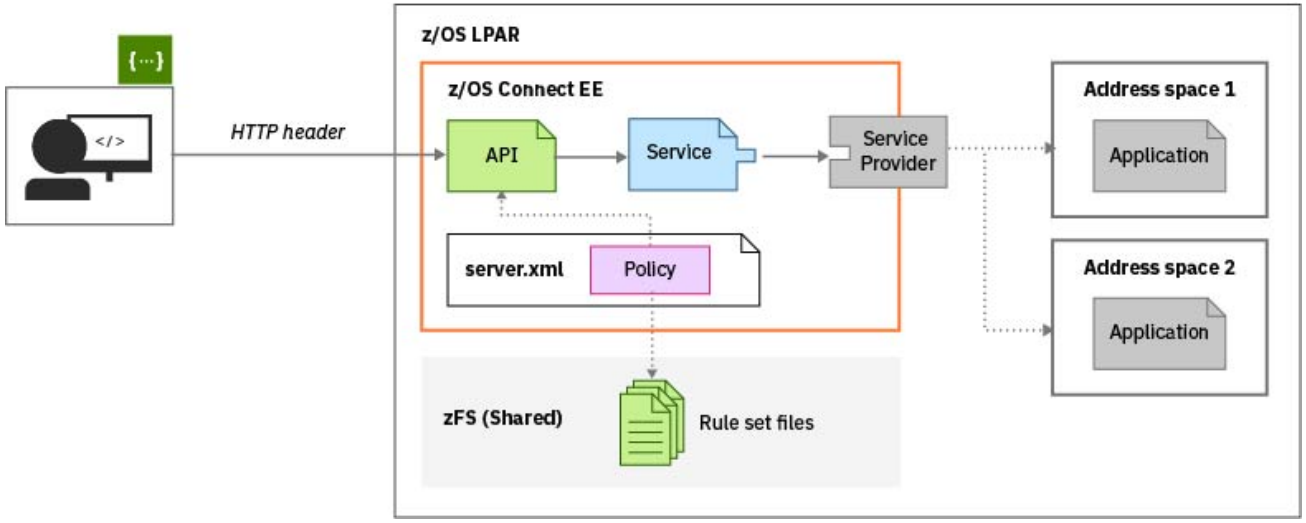
- HTTP header properties can be used to select alternative for IMS (V3.0.4) , CICS (V3.0.10), Db2 (V3.0.36) or MQ (V3.0.39)
- Policies can be configured globally for every API in the server or for individual APIs (V3.0.11)

CICS attributes
- cicsCcsid
- cicsConnectionRef
- cicsTransId

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

Db2 attributes
- db2ConnectionRef
- db2CollectionID

MQ attributes
- mqConnectionFactory
- mqDestination
- mqReplyDestination

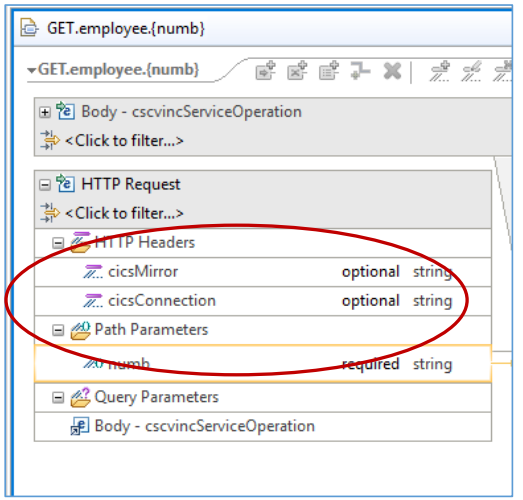


mitchj@us.ibm.com

A sample API Policies for CICS

z/OS Connect EE

```
<ruleset name="CICS rules">
     <rule name="csmi-rule">
         <conditions>
             <header name="cicsMirror" value="CSMI,MIJO"/> *
         </conditions>
         <actions>
             <set property="cicsTransId" value="${cicsMirror}"/>
         </actions>
     </rule>
     <rule name="connection-rule">
         <conditions>
             <header name="cicsConnection"
                     value="cscvinc,cics92,cics93"/>
         </conditions>
         <actions>
             <set property="cicsConnectionRef" value="${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)

mitchj@us.ibm.com

Displaying zCEE messages on the console and/or spool

z/OS Connect EE

server.xml

```
<zosLogging wtoMessage=
    "BAQR0657E,BAQR0658E,BAQR0660E,BAQR0686E,BAQR0687E"
    hardCopyMessage=
    "BAQR0657E,BAQR0658E,BAQR0660E,BAQR0686E,BAQR0687E"/>
```

MVS Console

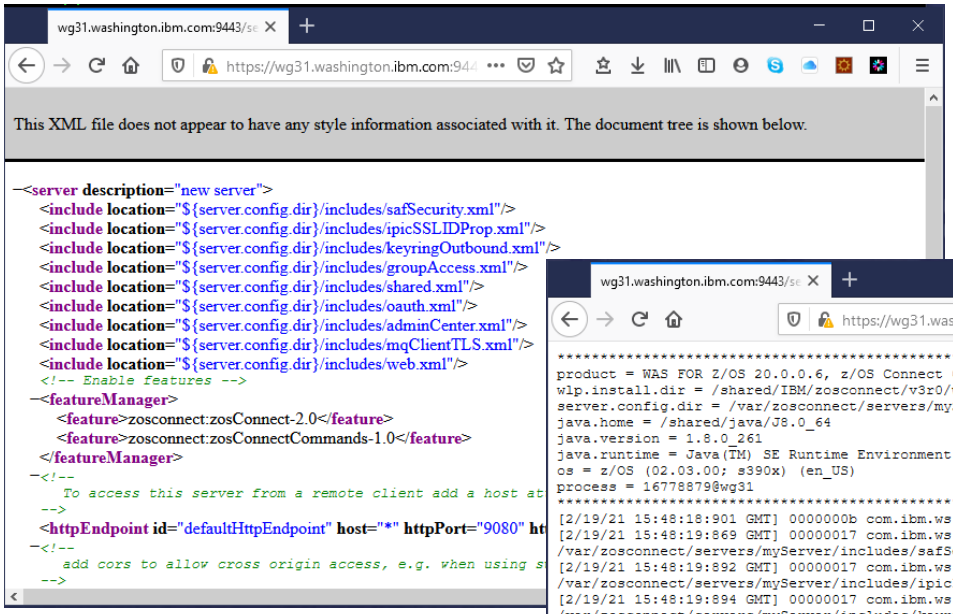
```
18.12.02 STC00137  +BAQR0686E: Program CSCVINC is not available in the CICS region with
   811              connection ID cscvinc; service cscvincService failed.
18.12.02 STC00137  +BAQR0686E: Program CSCVINC is not available in the CICS region with
   812              connection ID cscvinc; service cscvincService failed.
19.07.12 STC00137  +BAQR0657E: Transaction abend MIJO occurred in CICS while using
   745              connection cscvinc and service cscvincService.
```

STDERR

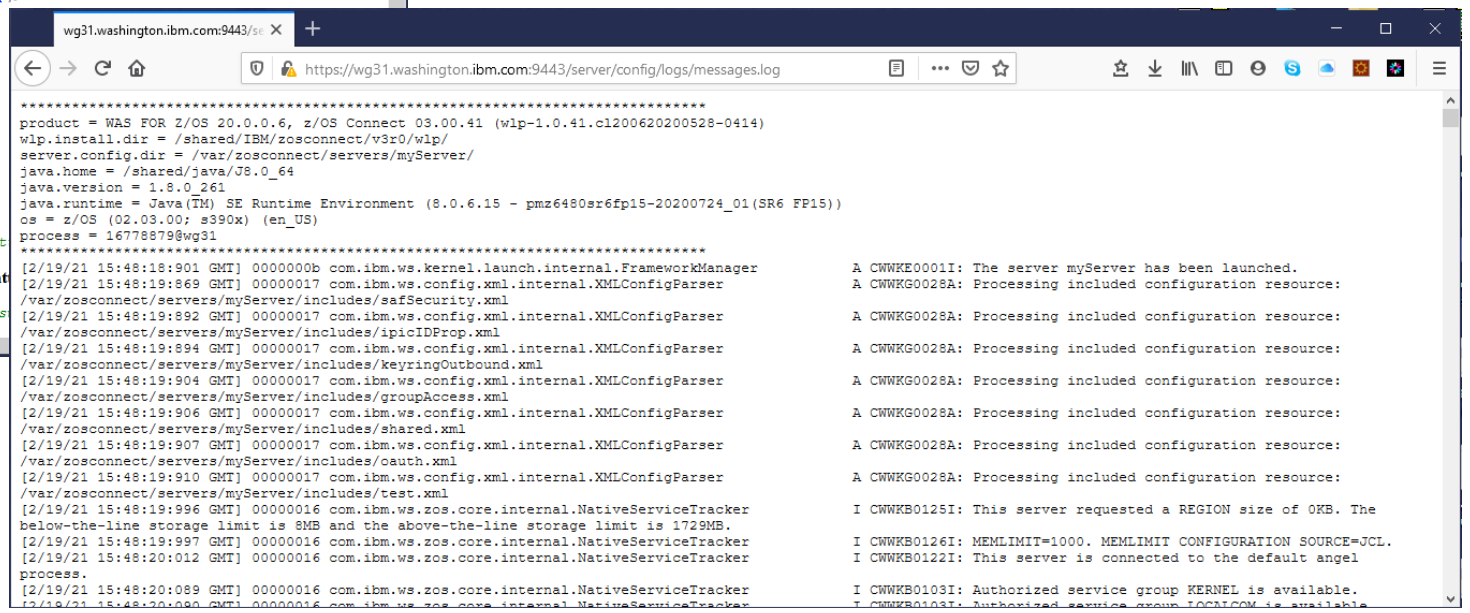
```
ÝERROR   ¨ BAQR0686E: Program CSCVINC is not available in the CICS region with connection cscvinc and service cscvincService.
ÝERROR   ¨ BAQR0686E: Program CSCVINC is not available in the CICS region with connection cscvinc and service cscvincService.
ÝERROR   ¨ BAQR0657E: Transaction abend MIJO occurred in CICS while using CICS connection cscvinc and service cscvincService.
```

mitchj@us.ibm.com

Tech/Tip: Providing access to configuration/log information



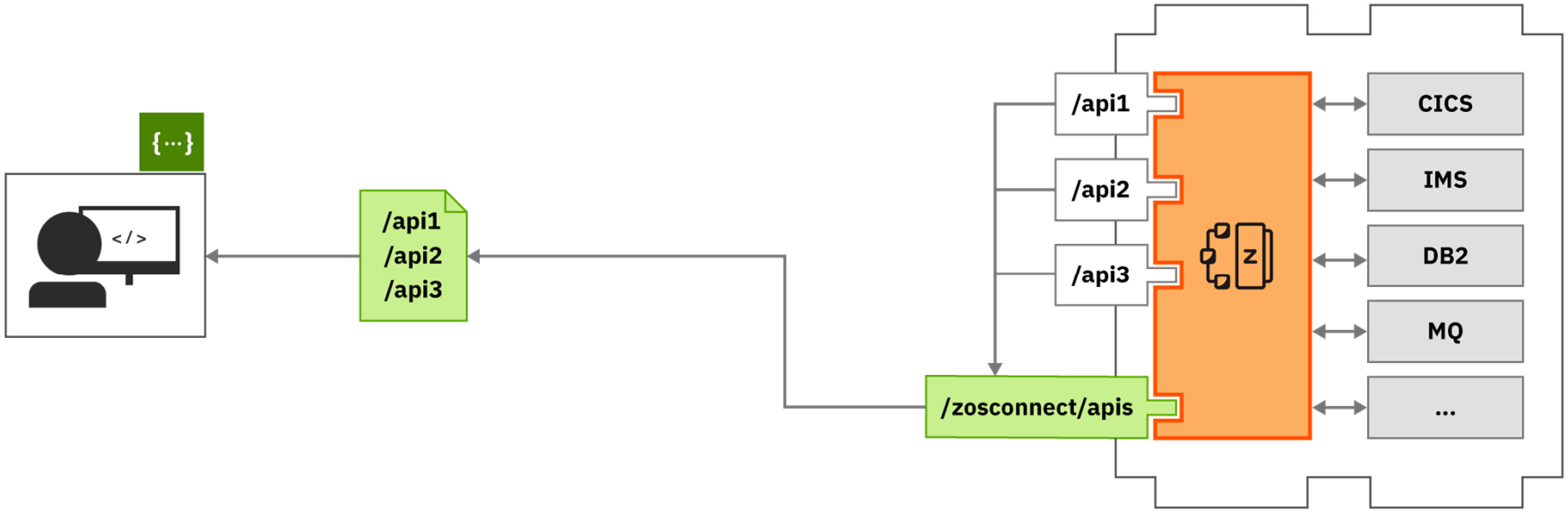

```
<webApplication id="serverConfig-location" name="serverConfig"
  location="${server.config.dir}">
  <web-ext context-root="/server/config"
    enable-file-serving="true" enable-directory-browsing="true">
    <file-serving-attribute name="extendedDocumentRoot"
      value="${server.config.dir}" />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



mitchj@us.ibm.com

API Documentation

z/OS Connect EE



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

mitchj@us.ibm.com

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

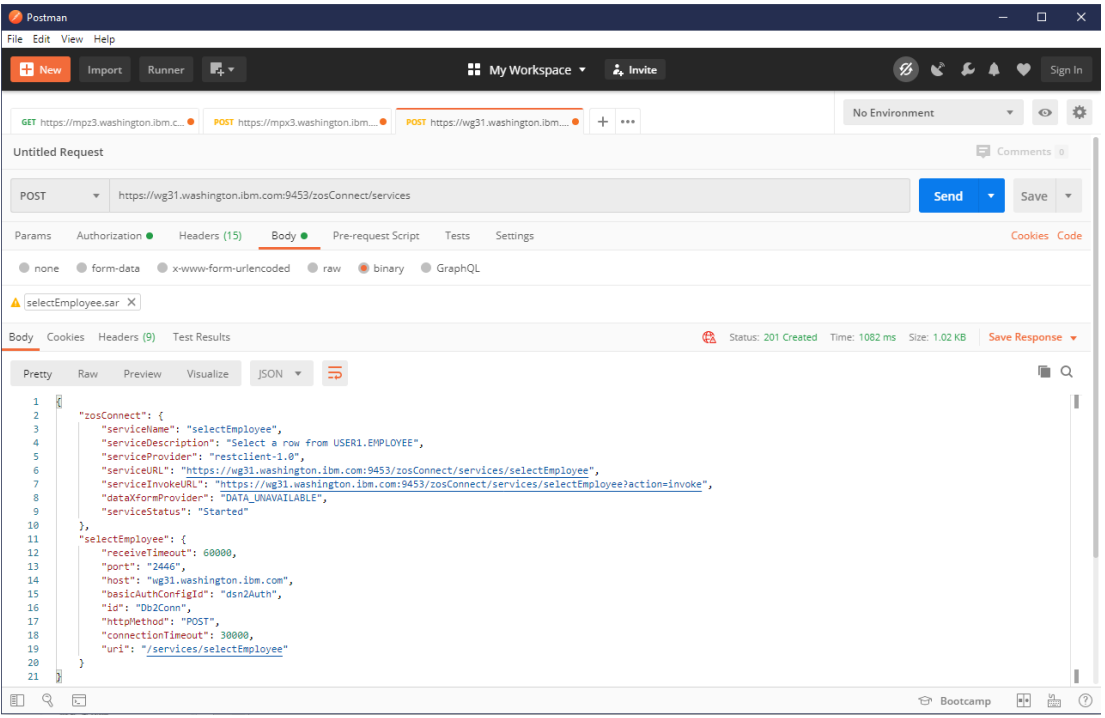
GET	/apiRequesters	Returns a list of all the deployed z/OS Connect API Requesters
POST	/apiRequesters	Deploys a new API Requester into z/OS Connect and invoke an API Requester call
DELETE	/apiRequesters/{apiRequesterName}	Undeploys an API Requester from z/OS Connect
GET	/apiRequesters/{apiRequesterName}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

Deploying Service Archive options

z/OS Connect EE

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



Command:

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

Results:


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

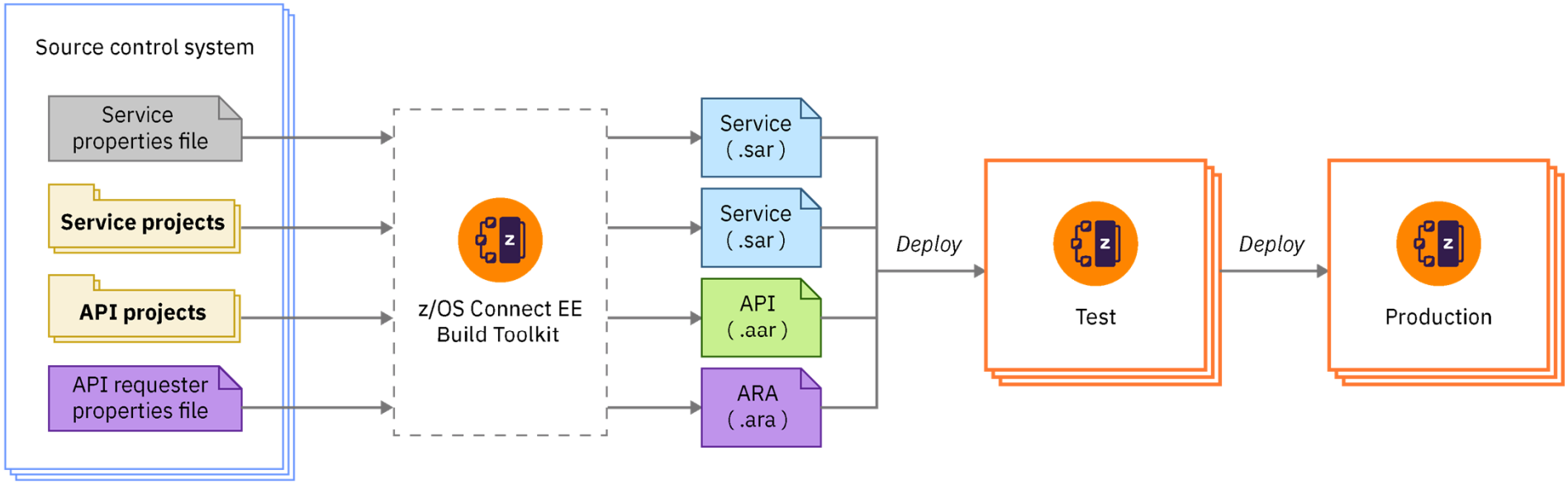

mitchj@us.ibm.com

DevOps using z/OS Connect EE

z/OS Connect EE

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

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



mitchj@us.ibm.com

ibm.biz/zosconnect-devops

Deploying Service Archive – z/OS options

z/OS Connect EE

```
//*************************************************************
//*  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 stop API cscvinc
//*************************************************************
//LIST EXEC CURL
//SYSTSIN  DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; $CURL/bin/curl -X PUT  +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9443/zosConnect/apis/cscvinc?status=sto+
pped
//*************************************************************
//*  STEP CURL - use cURL to delete the API cscvinc
//*************************************************************
//DELETE EXEC CURL
//SYSTSIN  DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; $CURL/bin/curl -X DELETE +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9443/zosConnect/apis/cscvinc
//*************************************************************
//*  STEP CURL - use cURL to deploy the API cscvinc
//*************************************************************
//DEPLOY EXEC CURL
//SYSTSIN  DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; $CURL/bin/curl -X POST +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
--data-binary @/u/johnson/cscvinc.aar +
--header "Content-Type: application/zip" +
https://wg31.washington.ibm.com:9443/zosConnect/apis
```

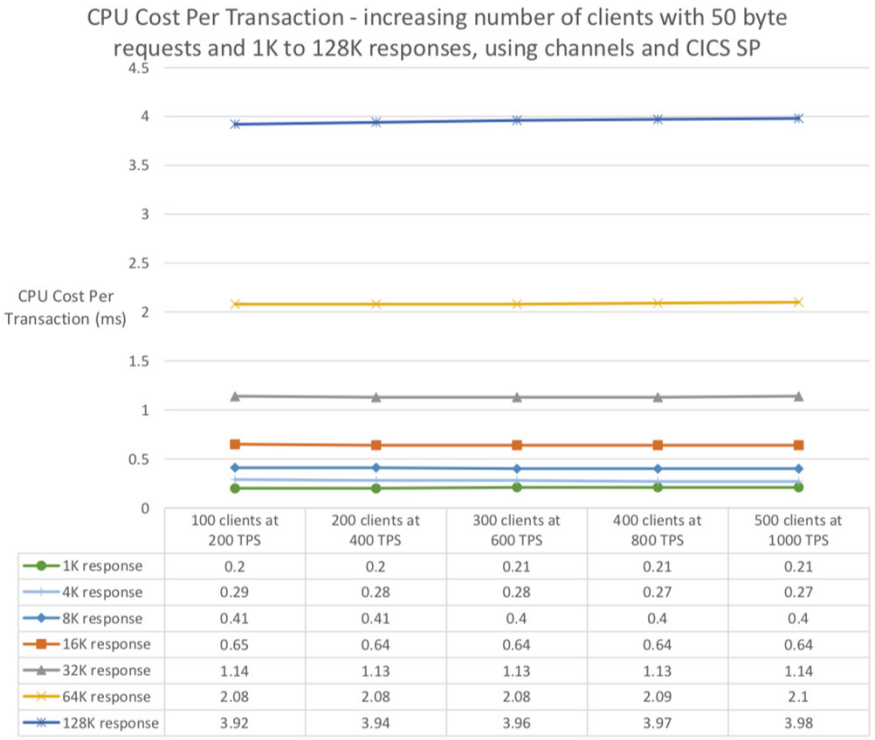
https://www.rocketsoftware.com/platforms/ibm-z/curl-for-zos

mitchj@us.ibm.com

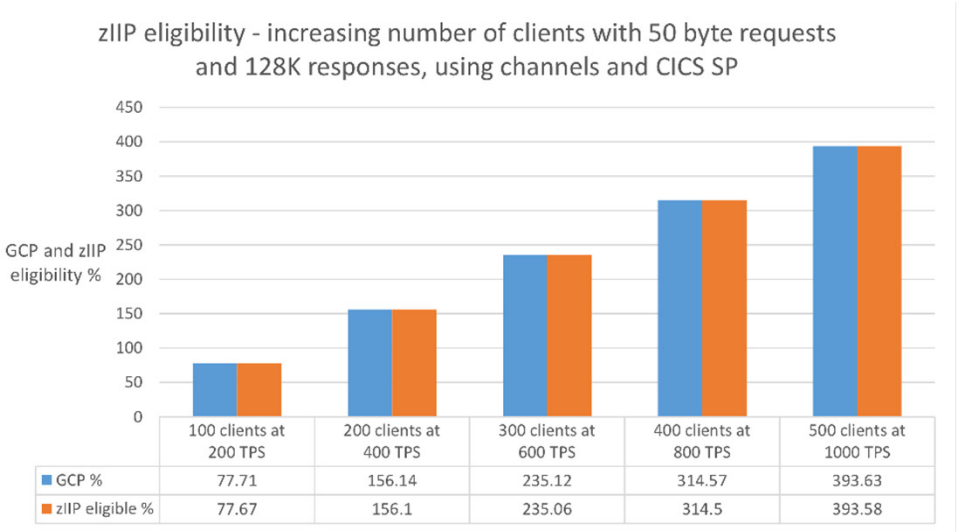
Performance: API Provider

High Speed, High Throughput, Low Cost

z/OS Connect EE



	100 clients at 200 TPS	200 clients at 400 TPS	300 clients at 600 TPS	400 clients at 800 TPS	500 clients at 1000 TPS
1K response	0.2	0.2	0.21	0.21	0.21
4K response	0.29	0.28	0.28	0.27	0.27
8K response	0.41	0.41	0.4	0.4	0.4
16K response	0.65	0.64	0.64	0.64	0.64
32K response	1.14	1.13	1.13	1.13	1.14
64K response	2.08	2.08	2.08	2.09	2.1
128K response	3.92	3.94	3.96	3.97	3.98



	100 clients at 200 TPS	200 clients at 400 TPS	300 clients at 600 TPS	400 clients at 800 TPS	500 clients at 1000 TPS
GCP %	77.71	156.14	235.12	314.57	393.63
zIIP eligible %	77.67	156.1	235.06	314.5	393.58

z/OS Connect EE is a Java-based product:
Over **99%** of its MIPs are **eligible for ZIIP offload**.

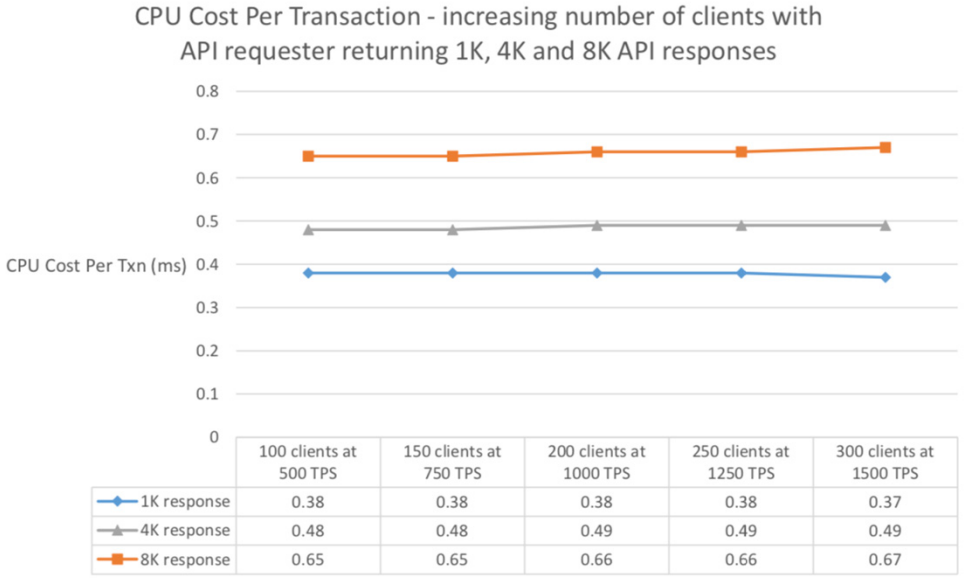
mitchj@us.ibm.com

ibm.biz/zosconnect-performance-report

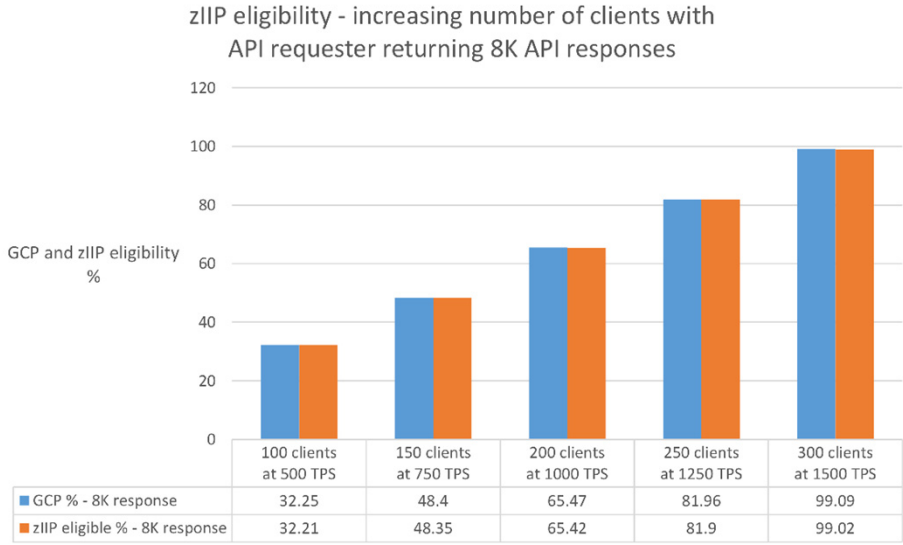
Performance: API Requester

High Speed, High Throughput, Low Cost

z/OS Connect EE



	100 clients at 500 TPS	150 clients at 750 TPS	200 clients at 1000 TPS	250 clients at 1250 TPS	300 clients at 1500 TPS
1K response	0.38	0.38	0.38	0.38	0.37
4K response	0.48	0.48	0.49	0.49	0.49
8K response	0.65	0.65	0.66	0.66	0.67



	100 clients at 500 TPS	150 clients at 750 TPS	200 clients at 1000 TPS	250 clients at 1250 TPS	300 clients at 1500 TPS
GCP % - 8K response	32.25	48.4	65.47	81.96	99.09
zIIP eligible % - 8K response	32.21	48.35	65.42	81.9	99.02

z/OS Connect EE is a Java-based product:
Over **99%** of its MIPs are **eligible for ZIIP offload**.

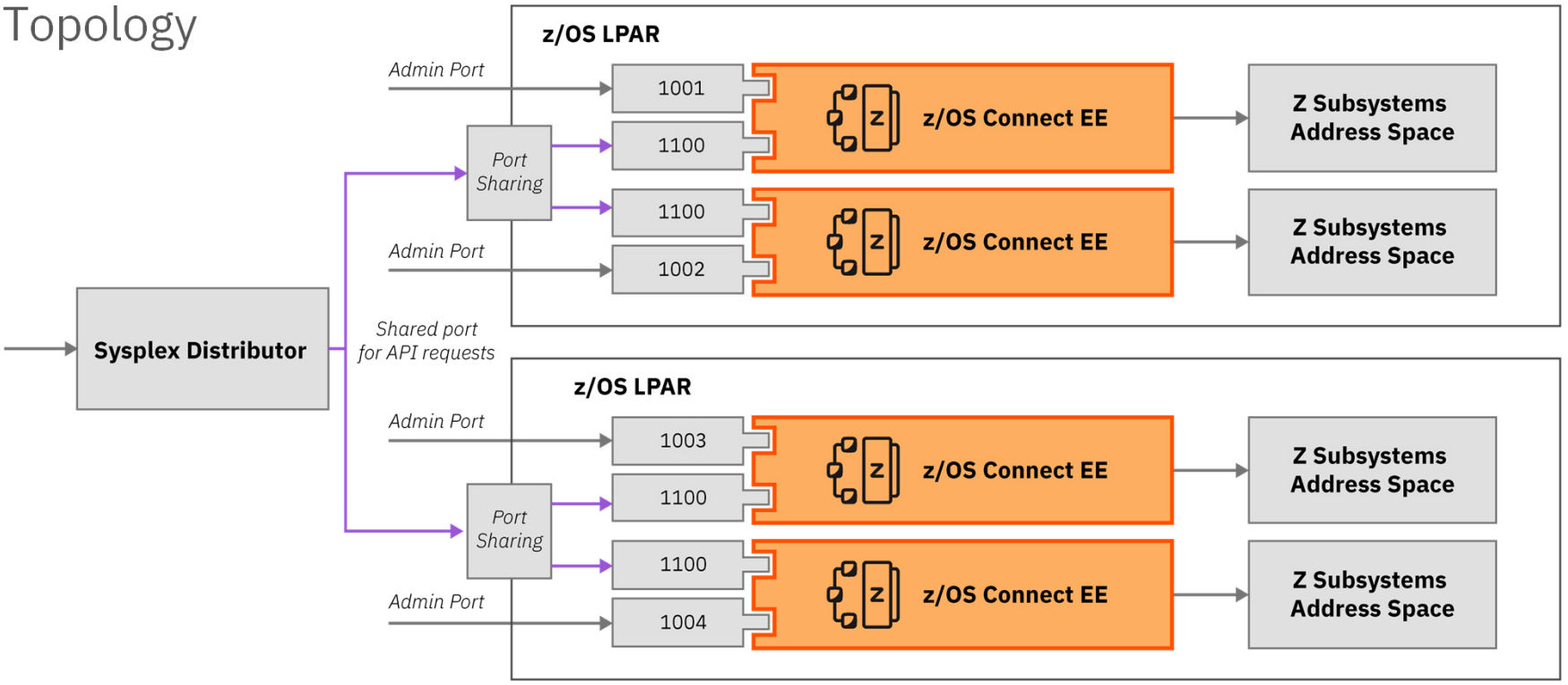
mitchj@us.ibm.com

ibm.biz/zosconnect-performance-report

High Availability

z/OS Connect EE

Topology



ibm.biz/zosconnect-ha-concepts

ibm.biz/zosconnect-scenarios

mitchj@us.ibm.com

/security

How is security implement?

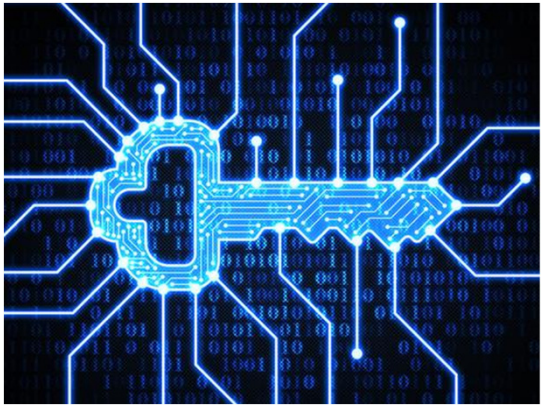
mitchj@us.ibm.com

General considerations for securing REST APIs

- Know who is invoking the API (**Authentication)**
  - Basic Authentication
  - TLS (mutual authentication)
  - Third Party Tokens

- Ensure that the data has not been altered in transit (**Data Integrity)** and ensure confidentiality of data in transit (**Encryption)**
  - TLS – (encrypted and signed messages)

- Control access (**Authorization)**
  - End user access to z/OS Connect
  - Access to APIs, etc.



mitchj@us.ibm.com

We need to understand the challenges

- Providing secure access between middleware components that use disparate security technologies e.g., registries like LDAP, SAF, TLS etc. in order to propagate security credentials from a client all the way to the targeted resource.

  ➢This is a driver for implementing open security models like OAuth and OpenID Connect and standard tokens like JWT

- Integrating security involves different products including z/OS Connect, WebSphere Liberty Profile on z/OS with CICS, IMS, Db2, MQ,… probably for the first time in your environment.

  ➢ Security for of these components are all documented in different places

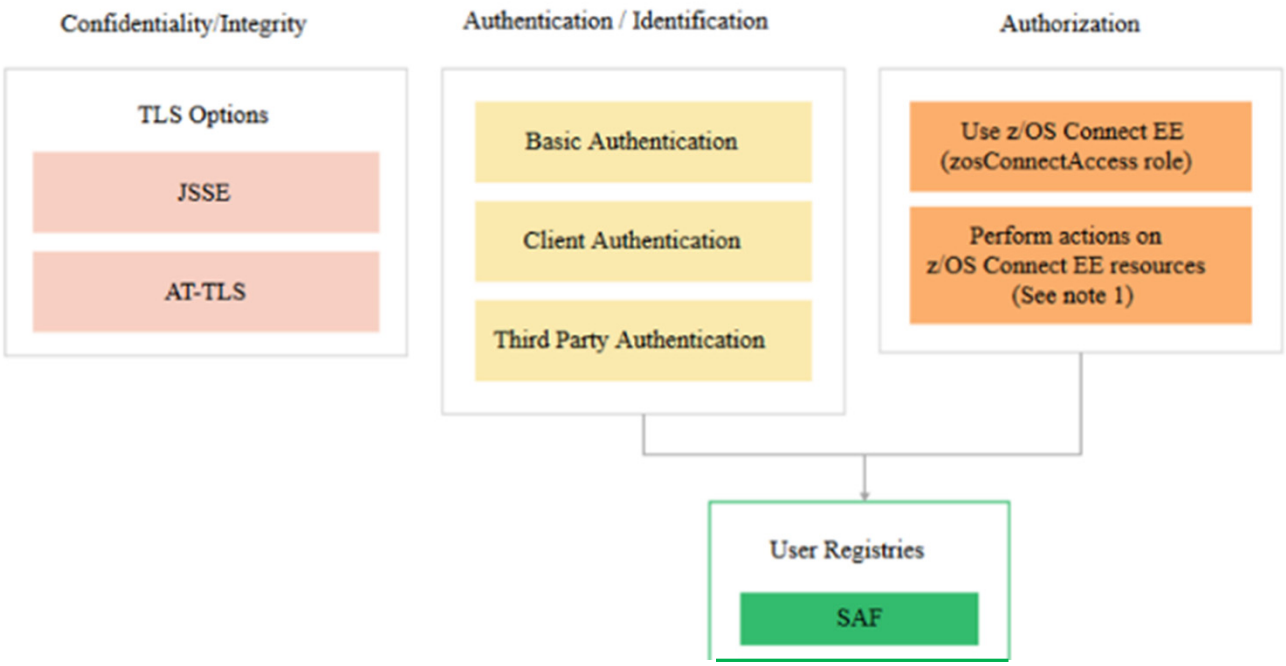
- Considering that security is often at odds with **performance**, the more secure techniques often mean more processing overhead, especially if not configured optimally

  ➢Remember security is usually not a choice but a requirement.

mitchj@us.ibm.com

mitchj@us.ibm.com

z/OS Connect EE security options



The actions which can be controlled by authorization (see Note 1 in the diagram above) are: deploying, querying, updating, starting, stopping and deleting of APIs, services and API requesters.

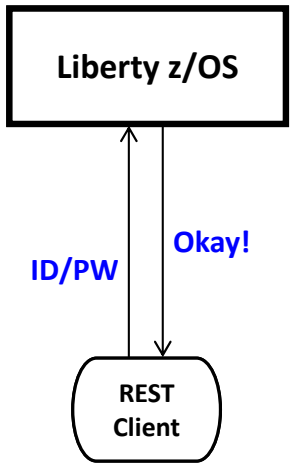
mitchj@us.ibm.com

API Provider Authentication

z/OS Connect EE

Several different ways this can be accomplished:

Basic Authentication



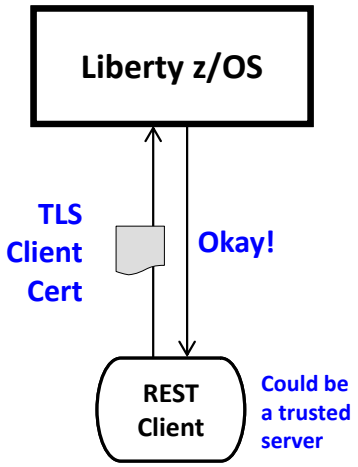
Server prompts for ID/PW

Client supplies ID/PW or ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

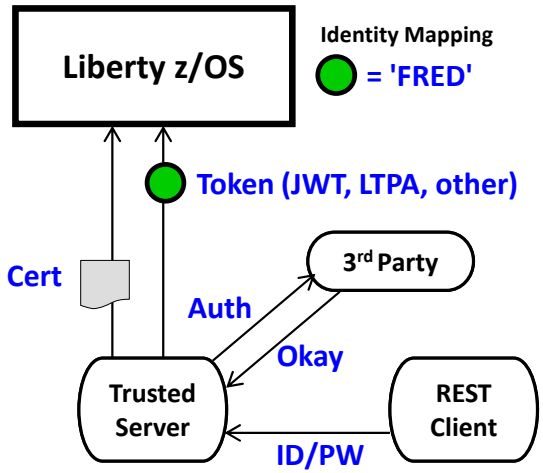
Client supplies certificate

Server validates cert and maps to an identity

Registry options:

- LDAP
- SAF

Third Party Authentication



Client authenticates to 3rd party sever

Client receives a trusted 3rd party token

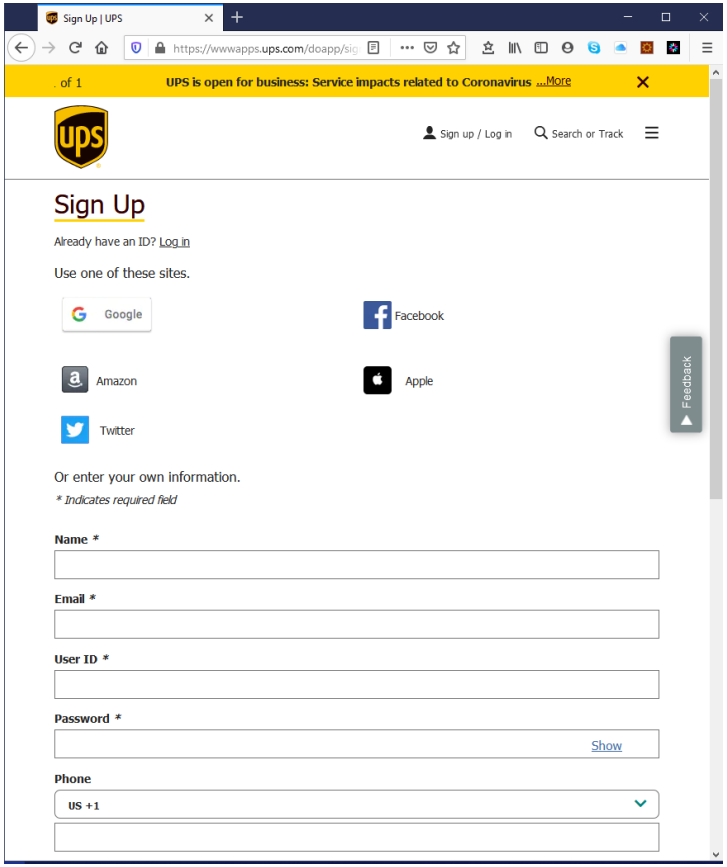
Token flows to Liberty z/OS and is mapped to an identity

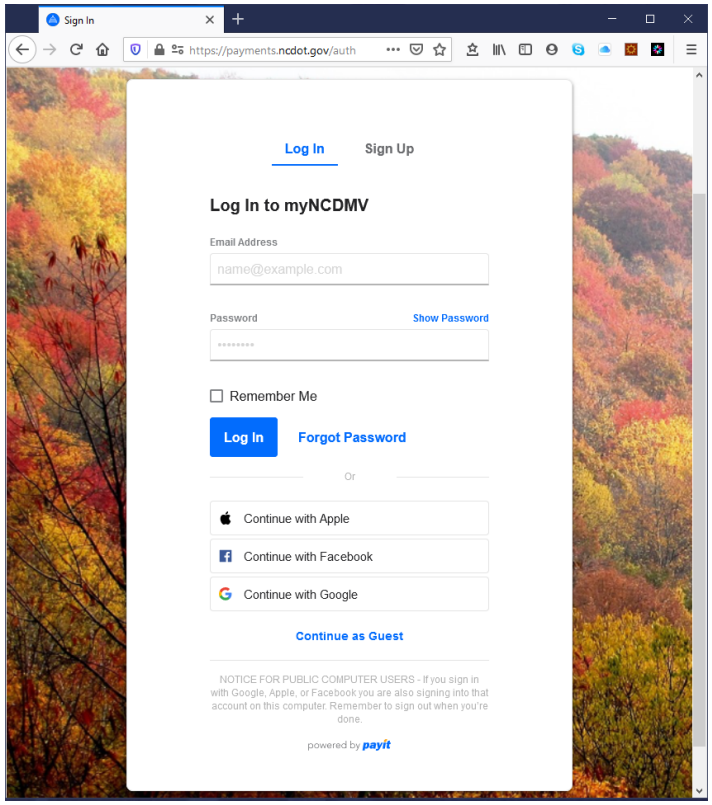
Registry options:

- LDAP
- SAF

mitchj@us.ibm.com

Third Party Authentication Examples

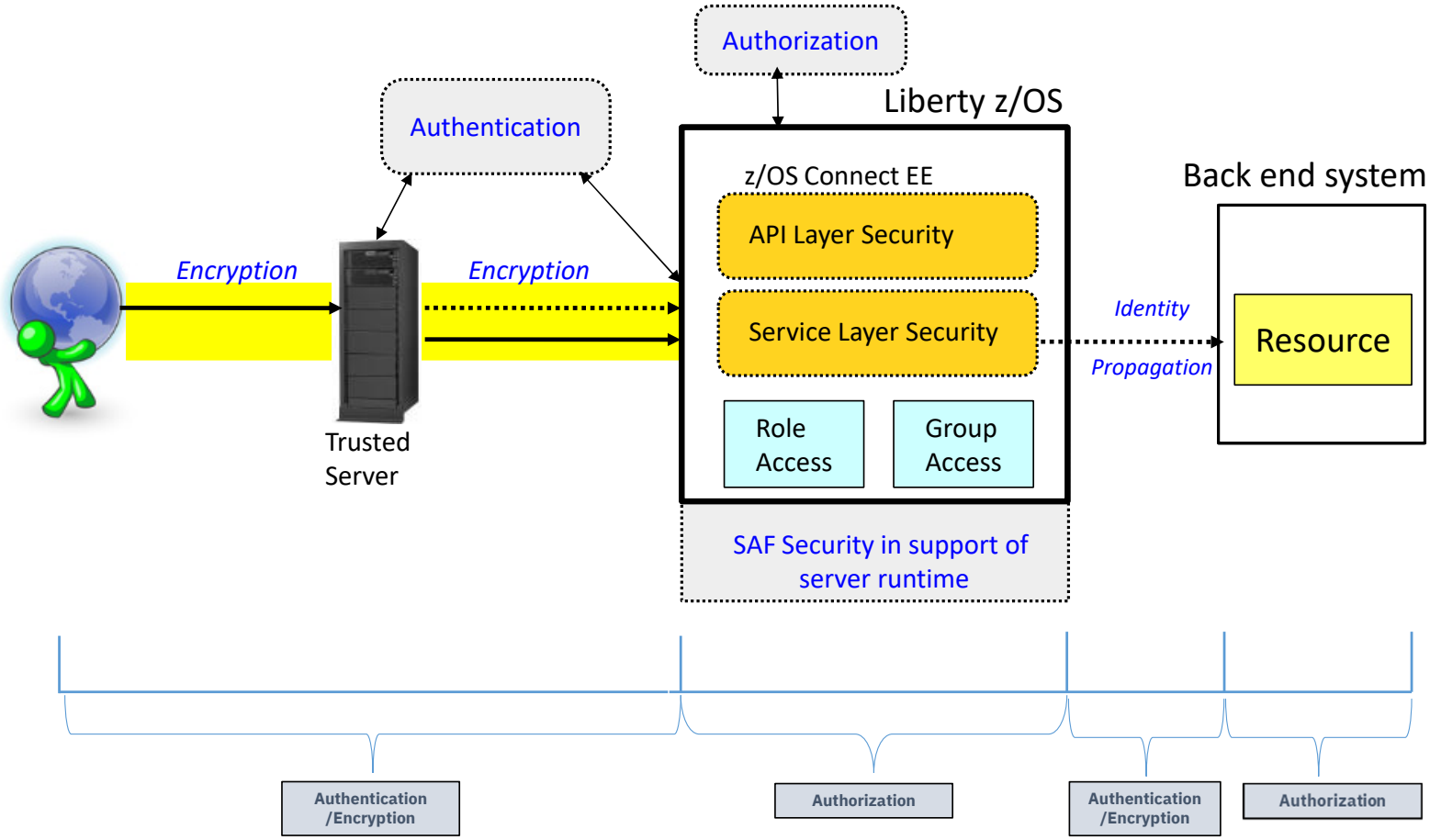




mitchj@us.ibm.com

z/OS Connect EE API provider security overview

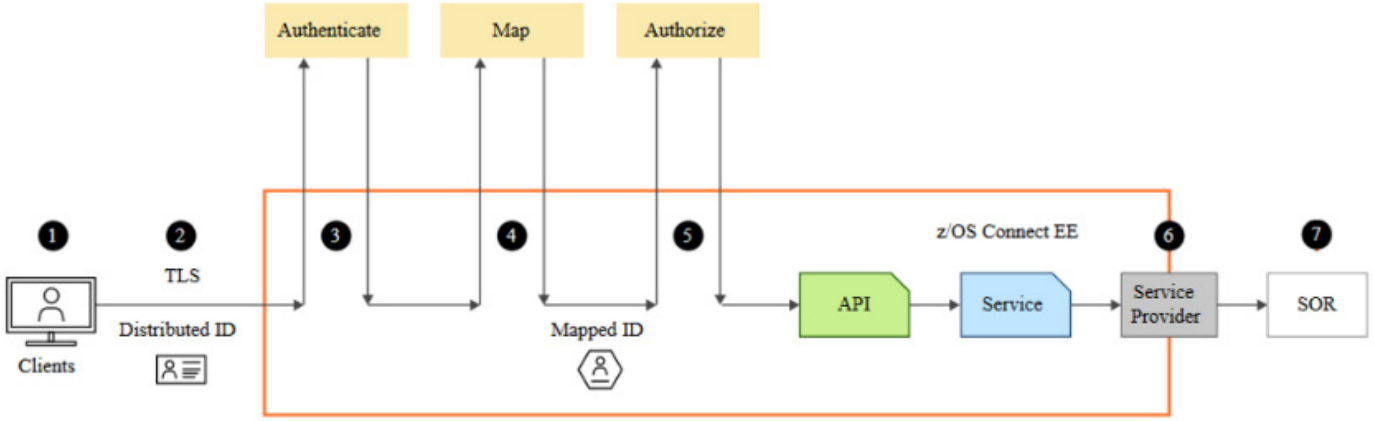
z/OS Connect EE



mitchj@us.ibm.com

Typical z/OS Connect EE API Provider security flow

z/OS Connect EE

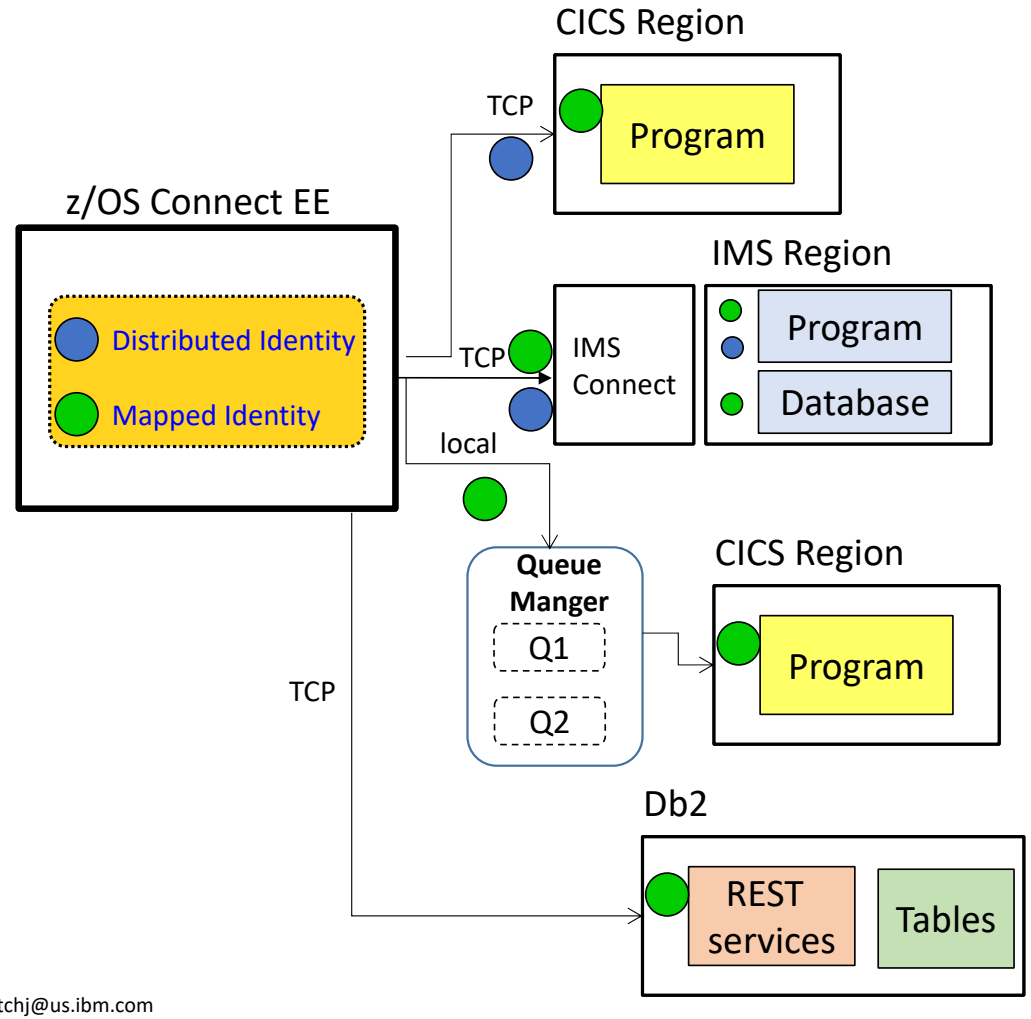


1. The credentials provided by the client
2. Secure the connection to the z/OS Connect EE server
3. Authenticate the client. This can be within the z/OS Connect EE server or by requesting verification from a third-party server
4. Map the authenticated identity to a user ID in the user registry
5. Authorize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

mitchj@us.ibm.com

Flowing an identity to the back end

z/OS Connect EE



The CICS SP propagates the distributed id to CICS (or sends the SAF id if it has been used for authentication)

The IMS TM and DB SPs asserts the mapped identity to IMS but also sends the distributed id for audit purposes (V3.0.33) added support for Passtickets.

The MQ SP asserts the mapped identity to the queue manager and onto CICS

The REST client SP can request a Passticket and send with mapped ID to Db2 (V3.0.15)

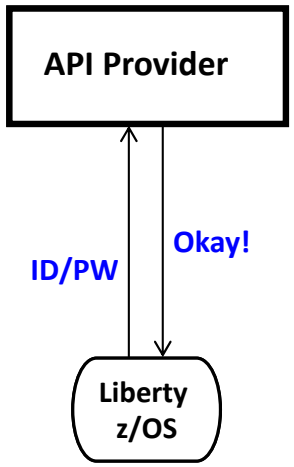
mitchj@us.ibm.com

API Requester Authentication

z/OS Connect EE

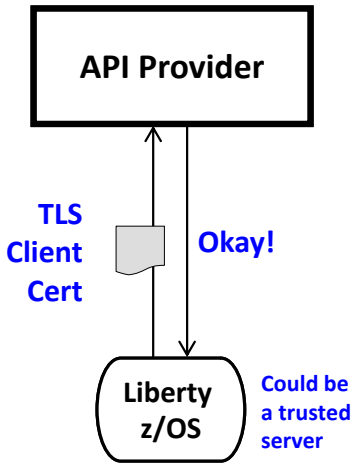
Several different ways this can be accomplished:

Basic Authentication



zCEE supplies ID/PW or ID/Passticket

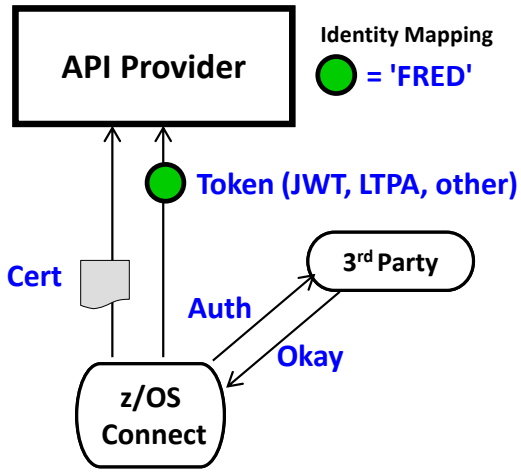
Client Certificate



Server prompts for certificate

zCEE supplies certificate

Third Party Authentication



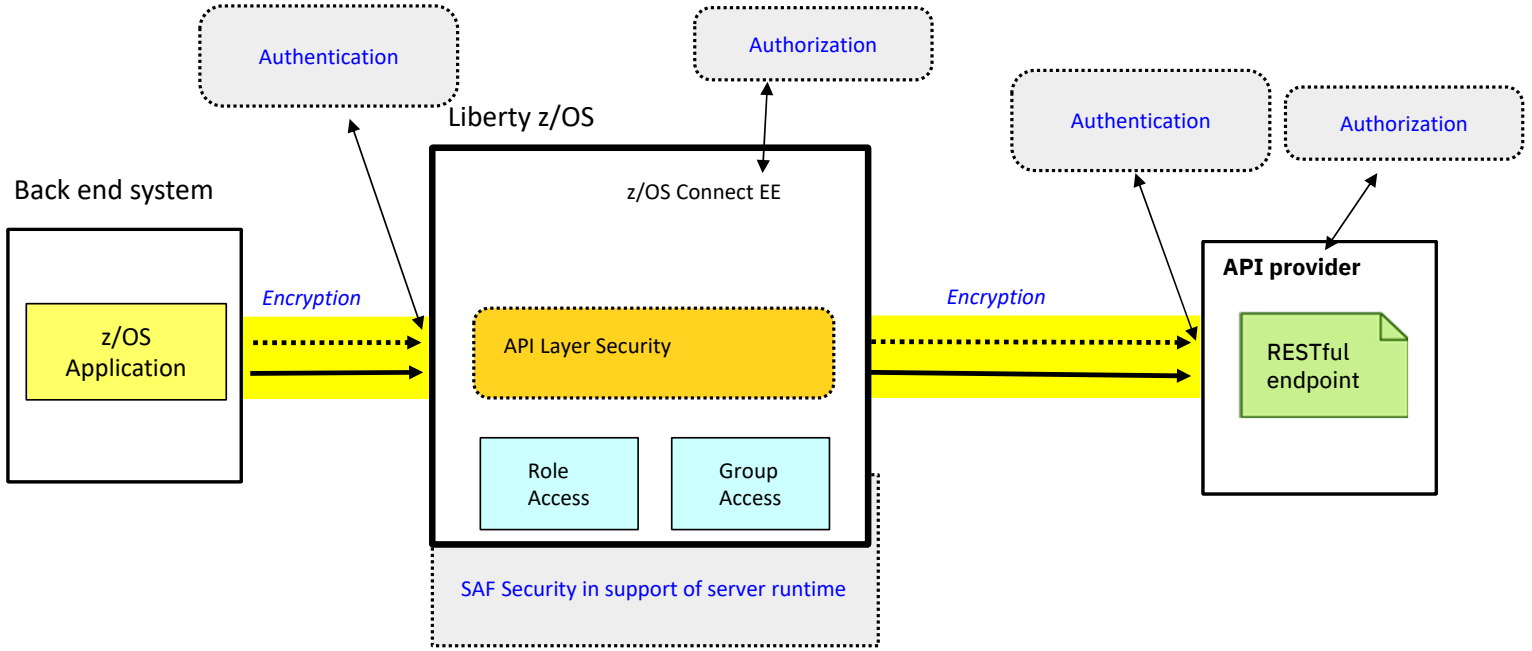
zCEE authenticates to 3rd party sever

zCEE receives a trusted 3rd party token

Token flows to API Provider

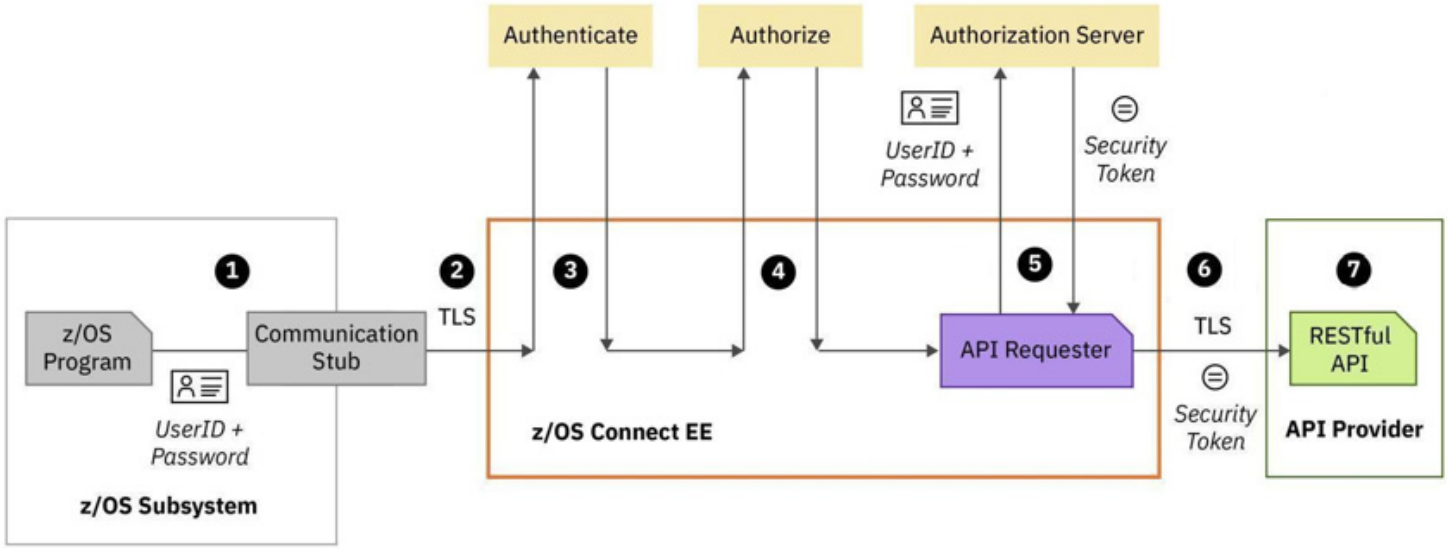
mitchj@us.ibm.com

API requester security – overview



mitchj@us.ibm.com

Typical z/OS Connect EE API Requester security flow



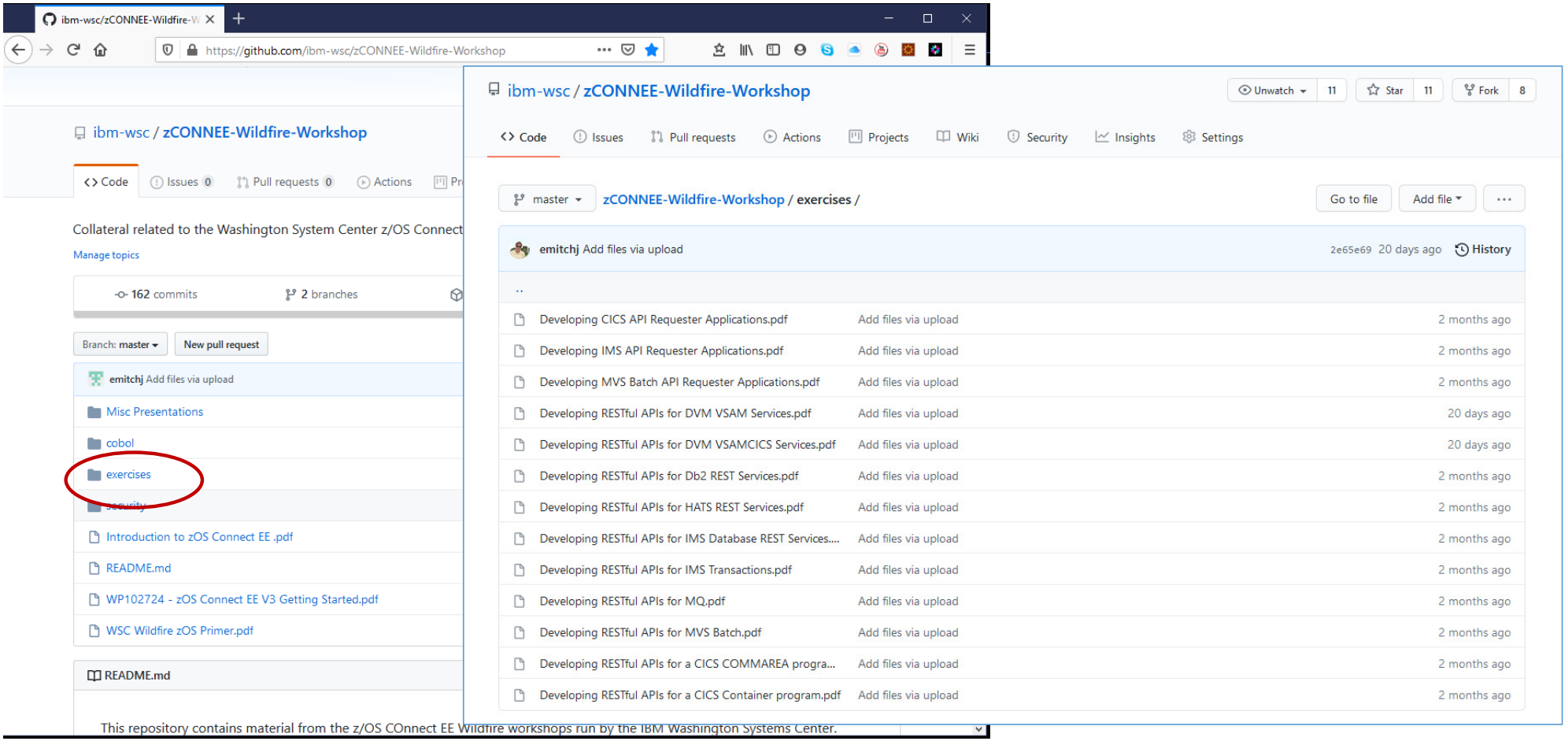
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS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

mitchj@us.ibm.com

z/OS Connect Wildfire Github Site

http://tinyurl.com/y28fsezs

z/OS Connect EE



mitchj@us.ibm.com

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

/questions?thanks=true

Thank you for listening.

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

mitchj@us.ibm.com