



IBM z/OS Connect Enterprise Edition

Introduction and Overview

Mitch Johnson

mitchj@us.ibm.com

Washington System Center

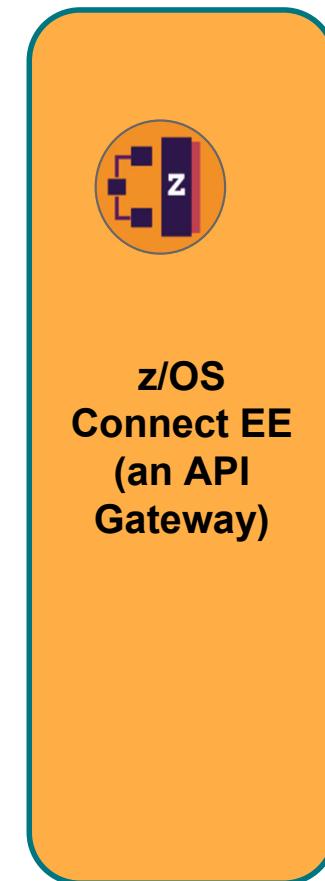


Agenda

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

*For more on security, contact your local IBM rep regarding the schedule of workshop *zCEESEC IBM z/OS Connect Administration/Security Wildfire Workshop*
© 2018, 2022 IBM Corporation

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



 z/OS Connect EE

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

+ HCL and Rocket Software

*Other Vendors or your own implementation

/but_first, what_is_REST?

What makes an API “RESTful”?

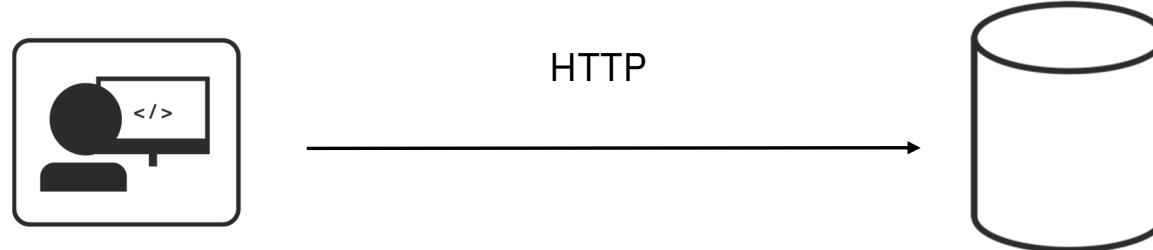
REST is architectural programming style

REST stands for **R**epresentational **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.

Key Principles of REST

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

POST
GET
PUT
DELETE

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

Use Path and Query parameters to refine the request

URI path identifies a resource (or lists of resources)

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

Request/Response Body is used to represent the data object

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



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



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>

Not every REST API is a RESTful API

(How to know if an API is not RESTful)

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

POST http://www.acme.com/customers/**GetCustomerDetails**/12345

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

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

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



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

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

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



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

Strive to design API to use RESTful properties

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

```
GET http://www.acme.com/customers/12345
```

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

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

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



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

3. Use JSON name/value pairs

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



```
RESPONSE HTTP 200 OK
```

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.

How do we describe a REST API?



/swagger/open_api

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

Why use Swagger?

It is more than just an API framework



There are a number of tools available to aid consumption:

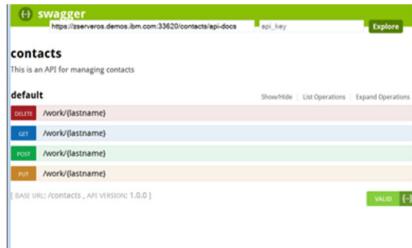
Consume Swagger*

Swagger Codegen create stub code to consume APIs from various languages



Read Swagger⁺

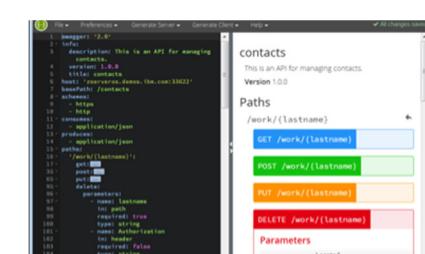
Swagger UI allows API consumers to easily browse and try APIs based on Swagger Doc.



The screenshot shows the Swagger UI interface for a 'contacts' API. It displays a list of operations under the 'default' path: GET /work/{lastname}, GET /work/{lastname}, POST /work/{lastname}, and PUT /work/{lastname}. The UI includes a sidebar with navigation links like 'Explore', 'ShowHide', and 'Expand Operations'. At the bottom, it shows the base URL as 'https://server.demos.ibm.com:33620/contacts/api-docs' and the API version as '1.0.0'.

Write Swagger

Swagger Editor allows API developers to design their swagger documents.



The screenshot shows the Swagger Editor interface for the same 'contacts' API. On the left, the Swagger document is displayed in JSON format. On the right, there are sections for 'Paths' (listing the four contact management endpoints) and 'Parameters' (showing optional parameters for each endpoint). The editor interface includes tabs for 'Preview', 'Generate Schema', 'Generate Client', and 'Help'.

* z/OS Connect API Requester

+z/OS Connect, MQ REST support, Zowe



Why **/zos_connect_ee?**

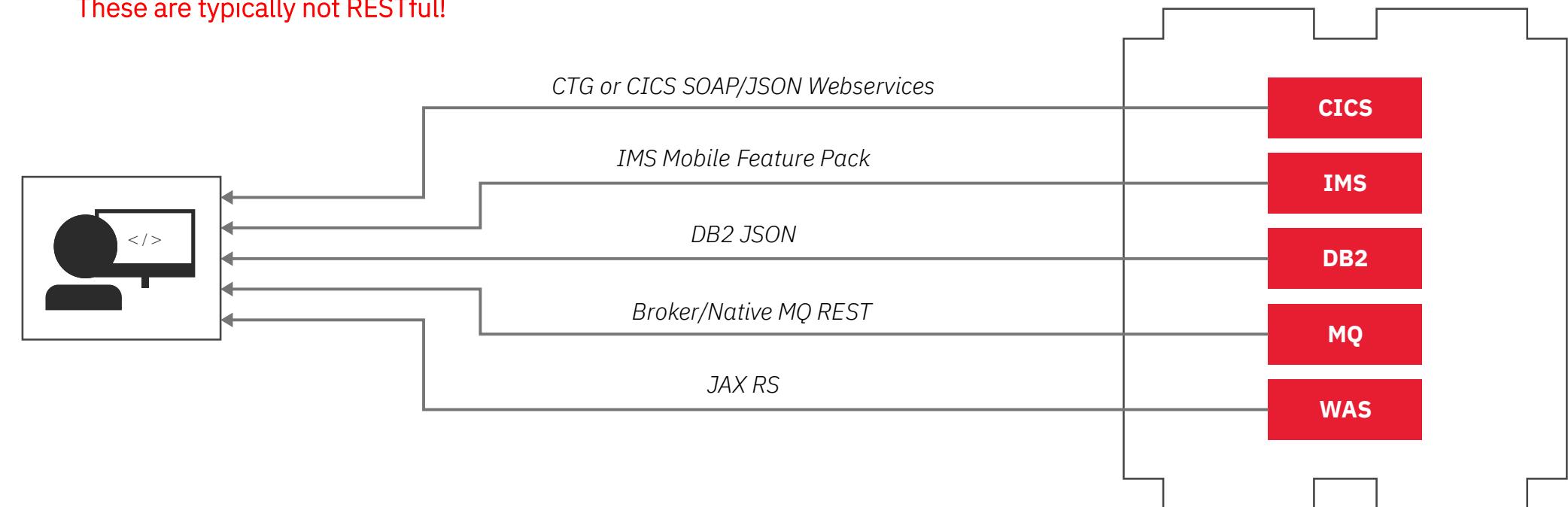
Truly RESTful APIs to and from your mainframe.

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!



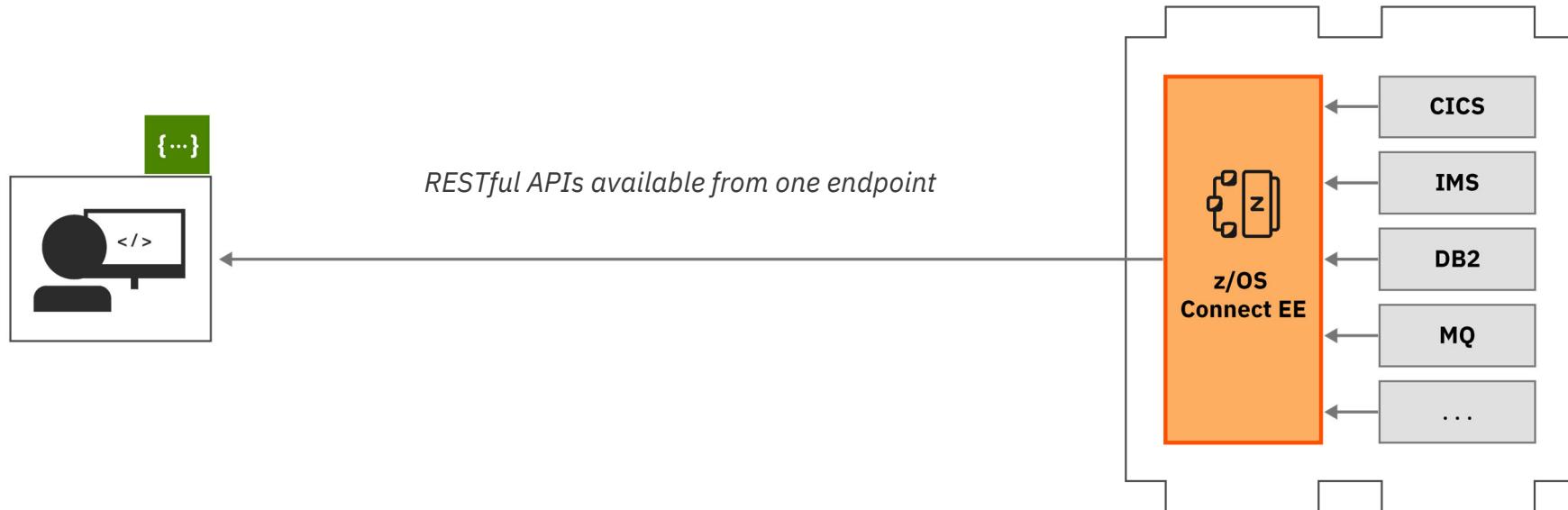


z/OS Connect provides a single-entry point

And exposes z/OS resources without writing any code.

z/OS Connect EE provides

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

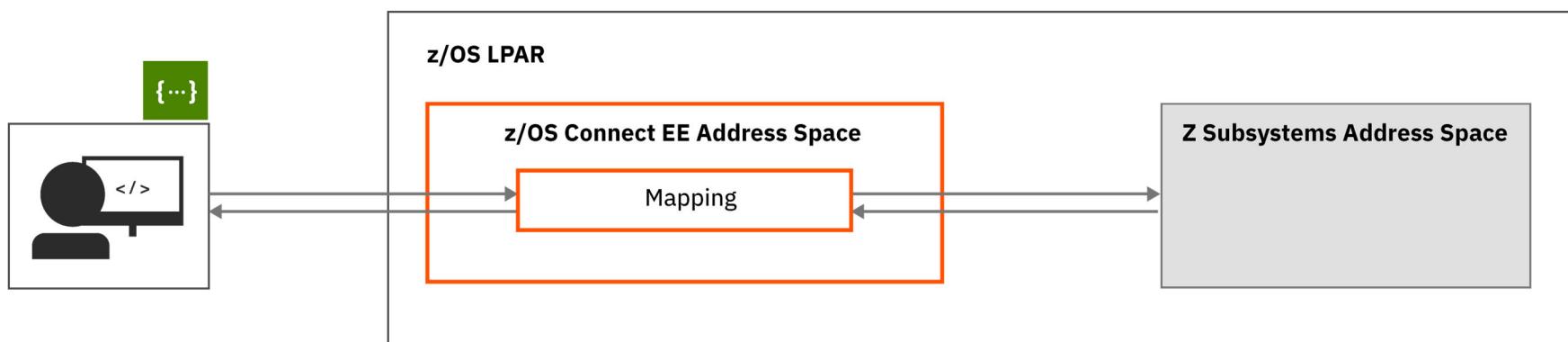




**Other than a RESTful interface,
what else does z/OS Connect provide?**

Let's Start with Data mapping

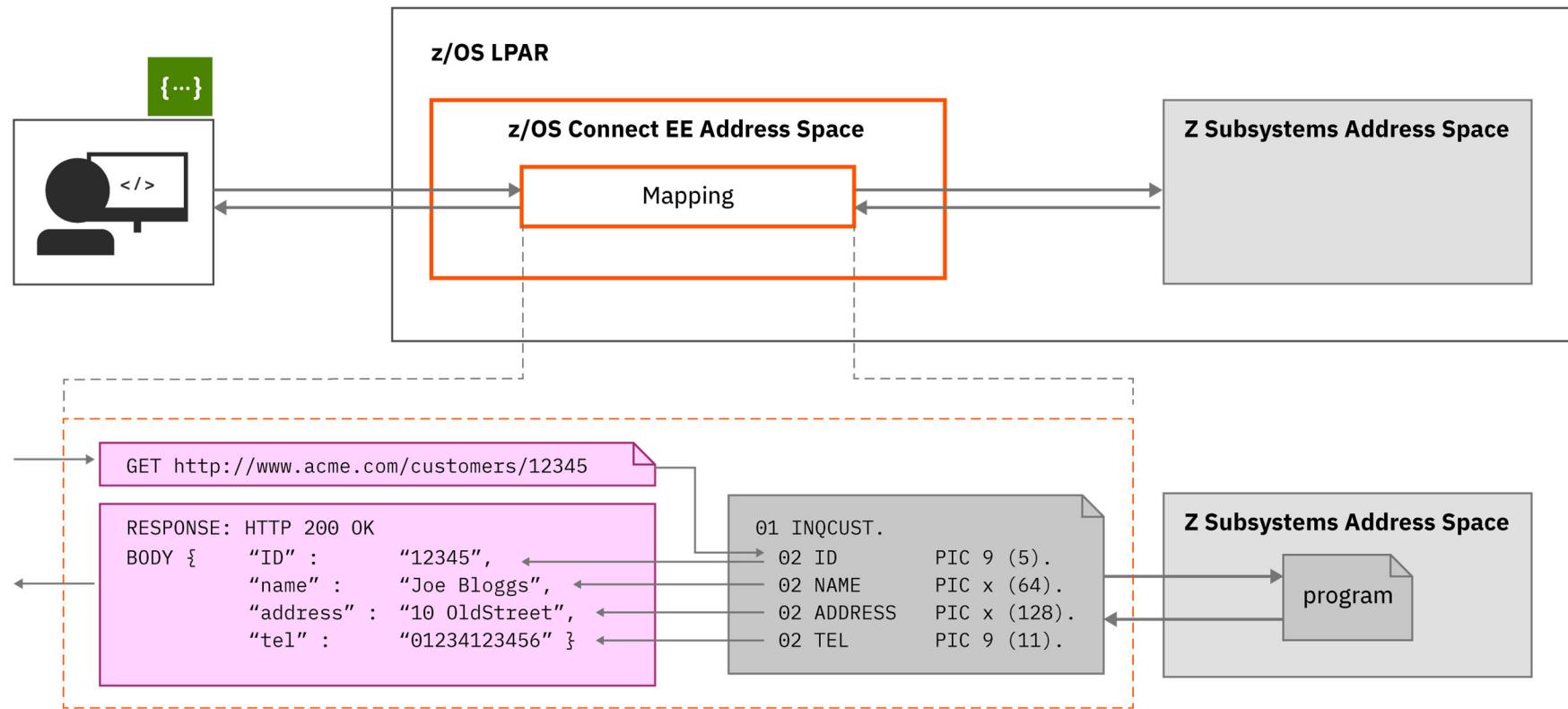
Converting the JSON message to the format the target's subsystem expects*.



* Most z/OS subsystems depend on information in a serial data format and do not normally work with JSON request/response messages. Examples of non-JSON formats are CICS COMMAREAAs 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).

Data mapping Example

A closer look



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



```
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" : 99999999999999.99,  
            "type" : "number",  
            "format" : "decimal"  
        },  
    }  
},
```

```
<?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="0">  
    <message id="1" name="miniloanRequest">  
        <segment id="1" name="COMMAREA" originalName="COMMAREA" included="Y" path="MINILOAN_COMMAREA">  
            <field name="MINILOAN_COMMAREA" originalName="MINILOAN_COMMAREA" included="Y" path="MINILOAN_COMMAREA.NAME">  
                <startPos>1</startPos>  
                <bytes>736</bytes>  
                <maxBytes>0</maxBytes>  
                <applicationDatatype datatype="STRUCT"/>  
                <field name="NAME" originalName="NAME" included="Y" path="MINILOAN_COMMAREA.NAME">  
                    <startPos></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>
```

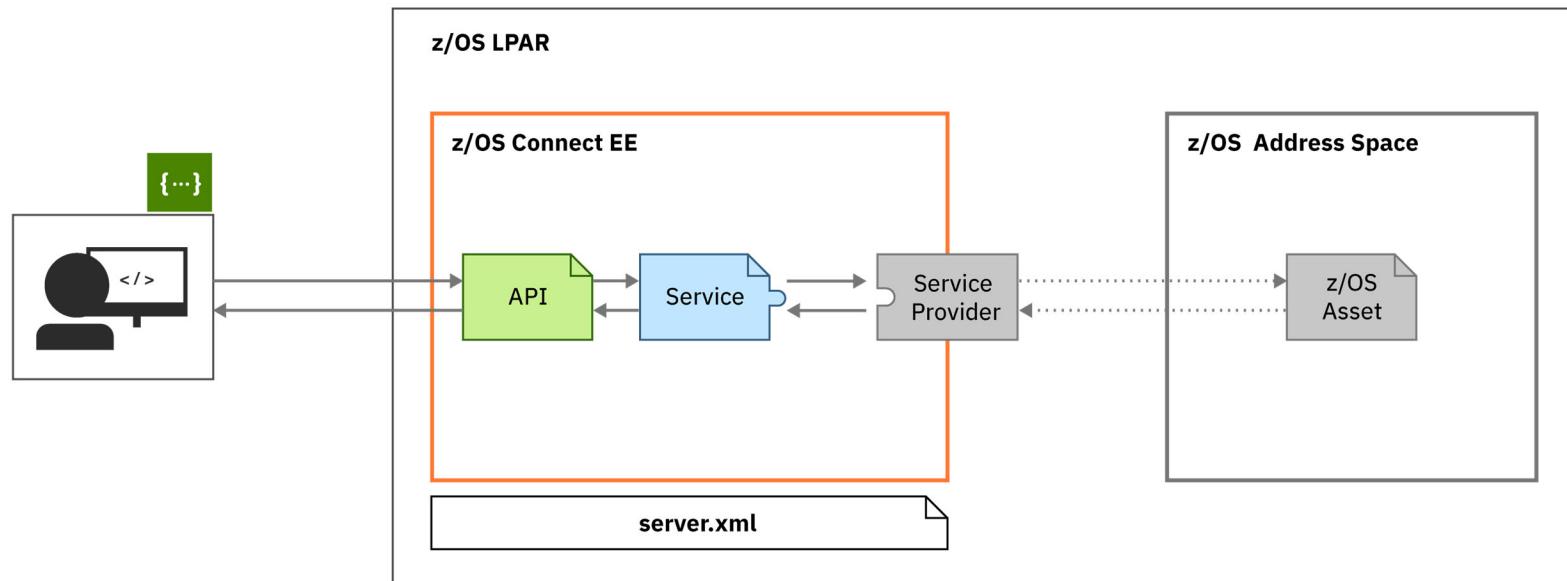
“name”：“Mitch Johnson”，
“creditScore”：“72000”

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

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

Steps to expose a z/OS application

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



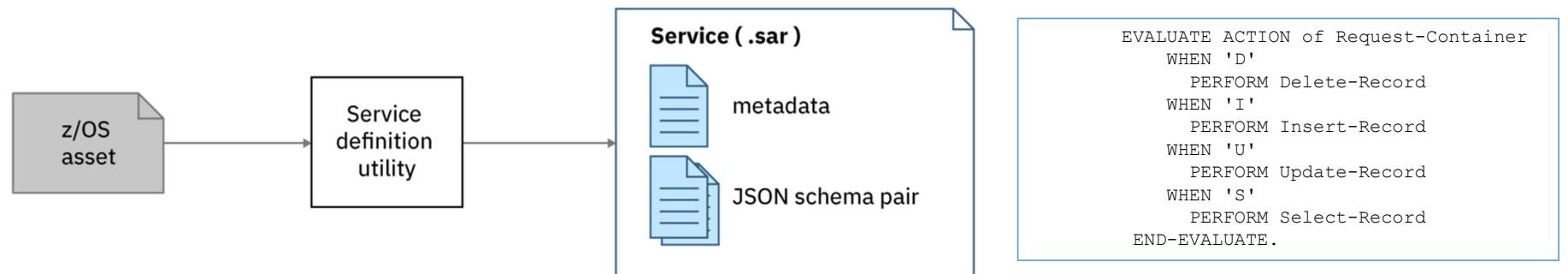
- The API provides the RESTful interface is ready to be consumed by a client and it requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific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

Steps to expose a z/OS application

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

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

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



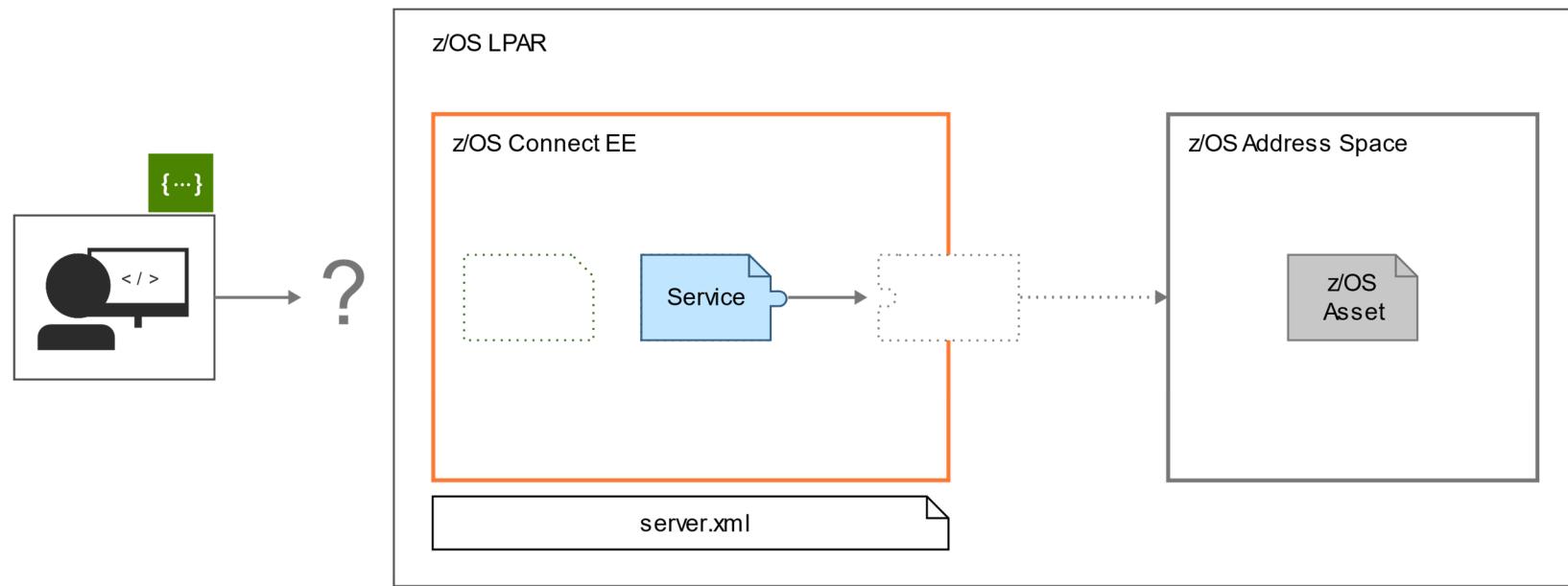
Use a system-appropriate utility to generate a service archive file for the z/OS application

- 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



Steps to expose a z/OS application

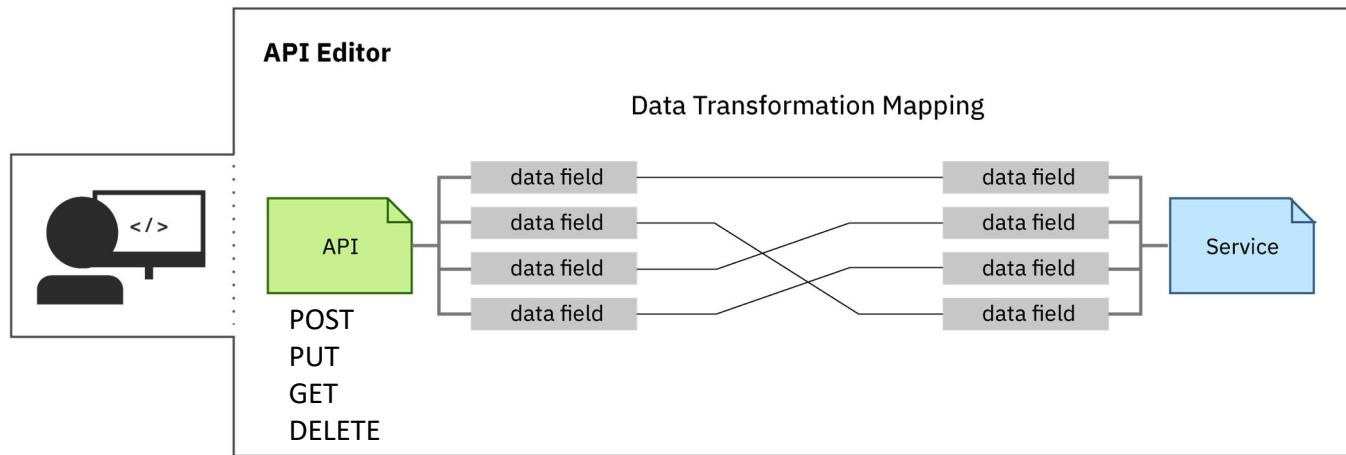
Deploy and export the service



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

Steps to expose a z/OS application

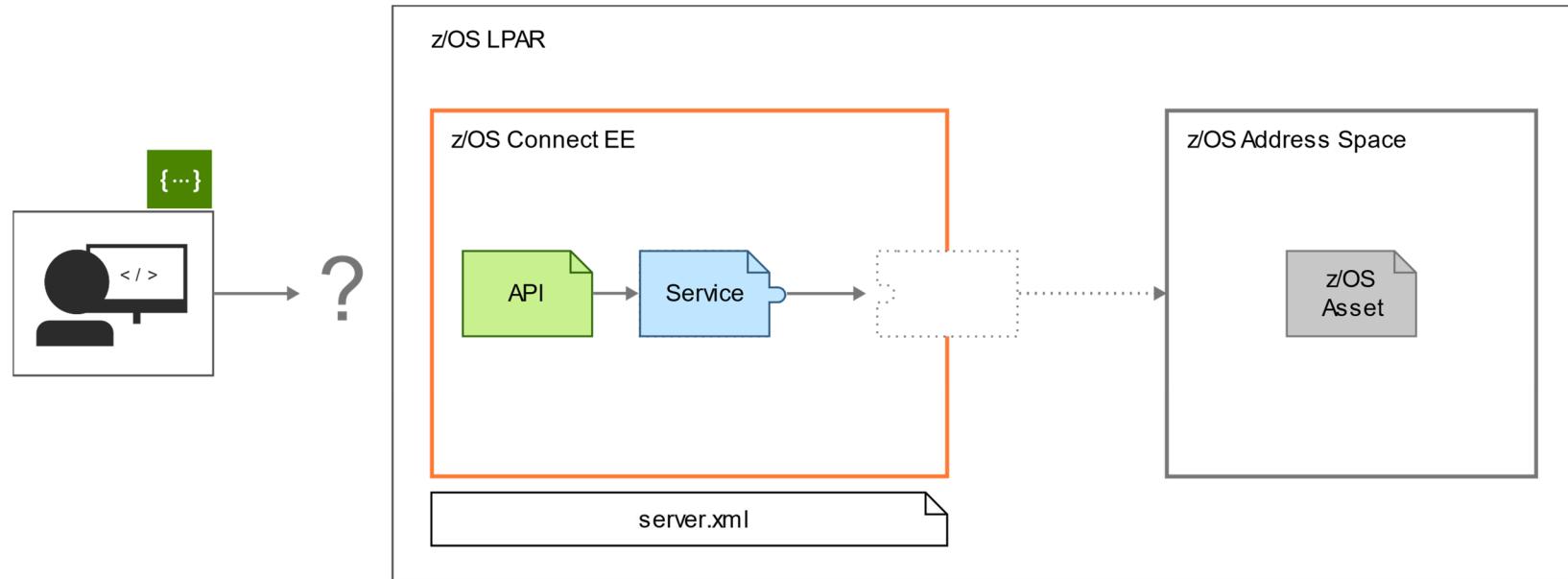
Export the service and then import it to create an API that consumes the service



- 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.

Steps to expose a z/OS application

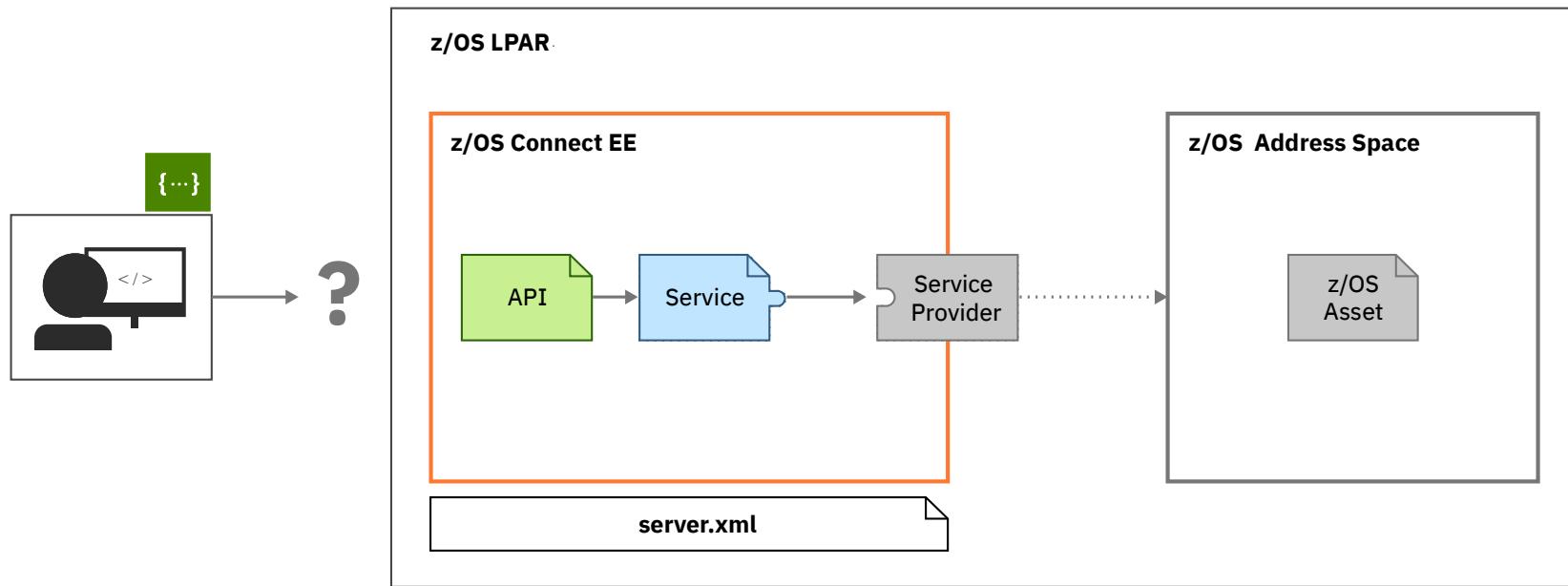
Deploy the API



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

Steps to expose a z/OS application

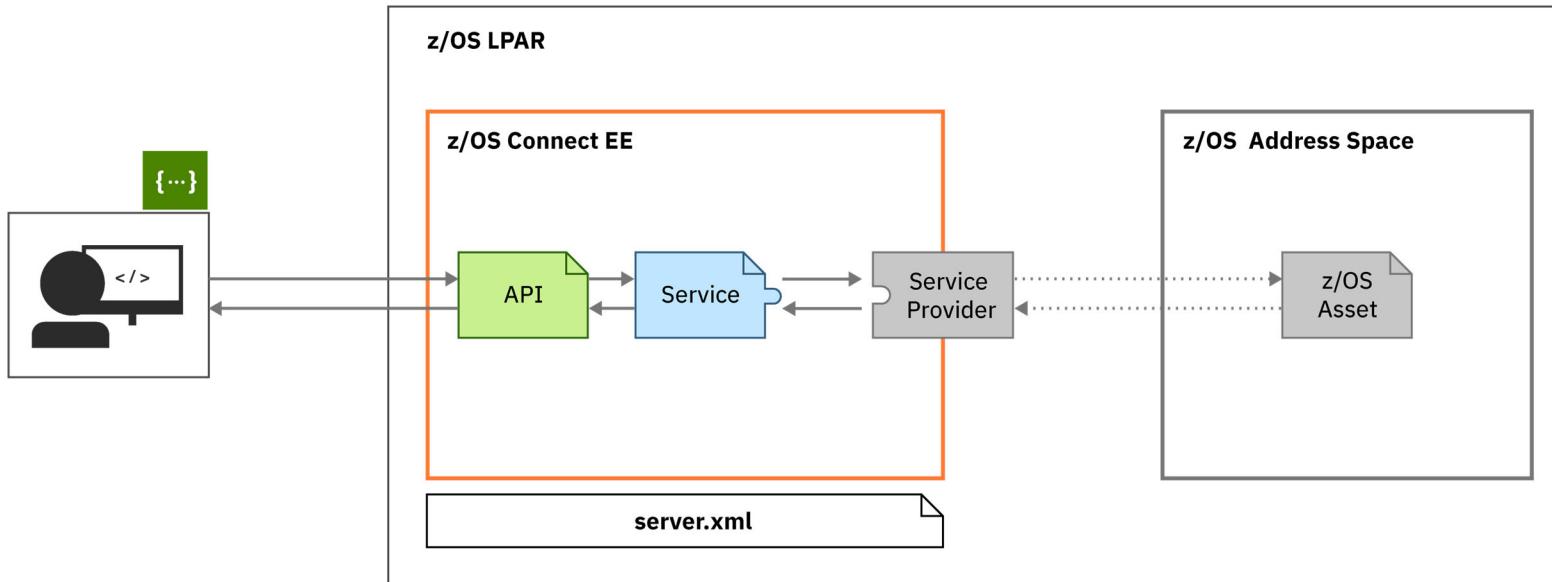
Configure the service provider



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

Steps to expose a z/OS application

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)

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



z/OS Connect EE

```

55
56
57
58
59
60
61
62
63
64
65
66
67
68
    // Invoke the REST API using a GET method
    URL url = new URL("https://wg31.washington.ibm.com:9453/cscvinc/employee/" + args[1]);
    System.out.println("URL: " + url);
    HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Content-Type", "application/json");
    byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
    conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
    try {
        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
        }
        BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
        while ((output = bufferReader.readLine()) != null) {
    
```

```

Problems @ Javadoc Declaration Console Coverage
<terminated> ZceeCICSGet [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 7, 2021, 2:54:24 PM)
URL: https://wg31.washington.ibm.com:9453/cscvinc/employee/22222
USERID: CICSUMER
CE1BRESP: 0
CE1BRESP2: 0
name: DR E. GRIFFITHS
employeeNumber: 22222
amount: $0022.00
address: FRANKFURT, GERMANY
phoneNumber: 20034151
date: 26 11 81
Response Message: {"cscvincSelectServiceOperationResponse": {"cscvincContainer": {"response": {"CE1BRESP2": "0", "USERID": "CICSUMER", "CE1BRESP": "0", "name": "DR E. GRIFFITHS", "employeeNumber": "22222", "amount": "$0022.00", "address": "FRANKFURT, GERMANY", "phoneNumber": "20034151", "date": "26 11 81"}}}
```

CICS

```

54
55
56
57
58
59
60
61
62
63
64
65
    // Invoke the REST API using a GET method
    URL url = new URL("https://wg31.washington.ibm.com:9453/mqapi/");
    System.out.println("URL: " + url);
    HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Content-Type", "application/json");
    byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
    conn.addRequestProperty("Authorization", new String(bytesEncoded));
    try {
        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
} catch (Exception e) {
    e.printStackTrace();
}
    
```

```

Problems @ Javadoc Declaration Console Coverage
<terminated> ZceeMqGet [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 7, 2021, 8:53:07 AM)
URL: https://wg31.washington.ibm.com:9453/mqapi/
NAME: TINA J YOUNG
NUMB: 001781
ADDRX: SINDELINGEN,GERMANY
PHONE: 70319990
DATEX: 21 06 77
AMOUNT: $0009.99
|
```

MQ

```

52
53
54
55
56
57
58
59
60
61
62
63
64
65
    // Invoke the REST API using a GET method
    URL url = new URL("https://wg31.washington.ibm.com:9453/db2/employee/" + args[1]);
    System.out.println("URL: " + url);
    HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Content-Type", "application/json");
    byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
    conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
    try {
        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
        }
        BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
        while ((output = bufferReader.readLine()) != null) {
    
```

```

Problems @ Javadoc Declaration Console Coverage
<terminated> ZceeDb2Get [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 7, 2021, 2:56:06 PM)
URL: https://wg31.washington.ibm.com:9453/db2/employee/000010
Employee Number: 000010
First Name : CHRISTINE
Last Name: HAAS
Middle Initial: I
Phone Number: 7030
|
```

Db2

```

53
54
55
56
57
58
59
59
60
61
    URL url = new URL("https://wg31.washington.ibm.com:9453/phonebook/contacts/" + args[1]);
    System.out.println("URL: " + url);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Content-Type", "application/json");
    byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
    conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
    try {
        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
} catch (Exception e) {
    e.printStackTrace();
}
    
```

```

Problems @ Javadoc Declaration Console Coverage
<terminated> ZceeIMSGet [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 17, 2021, 8:48:25 AM)
URL: https://wg31.washington.ibm.com:9453/phonebook/contacts/LAST1
lastName: LAST1
firstName: FIRST1
zipCode: D01/R01
extension: 8-111-1111
message: ENTRY WAS DISPLAYED
HTTP code: 200
|
```

IMS



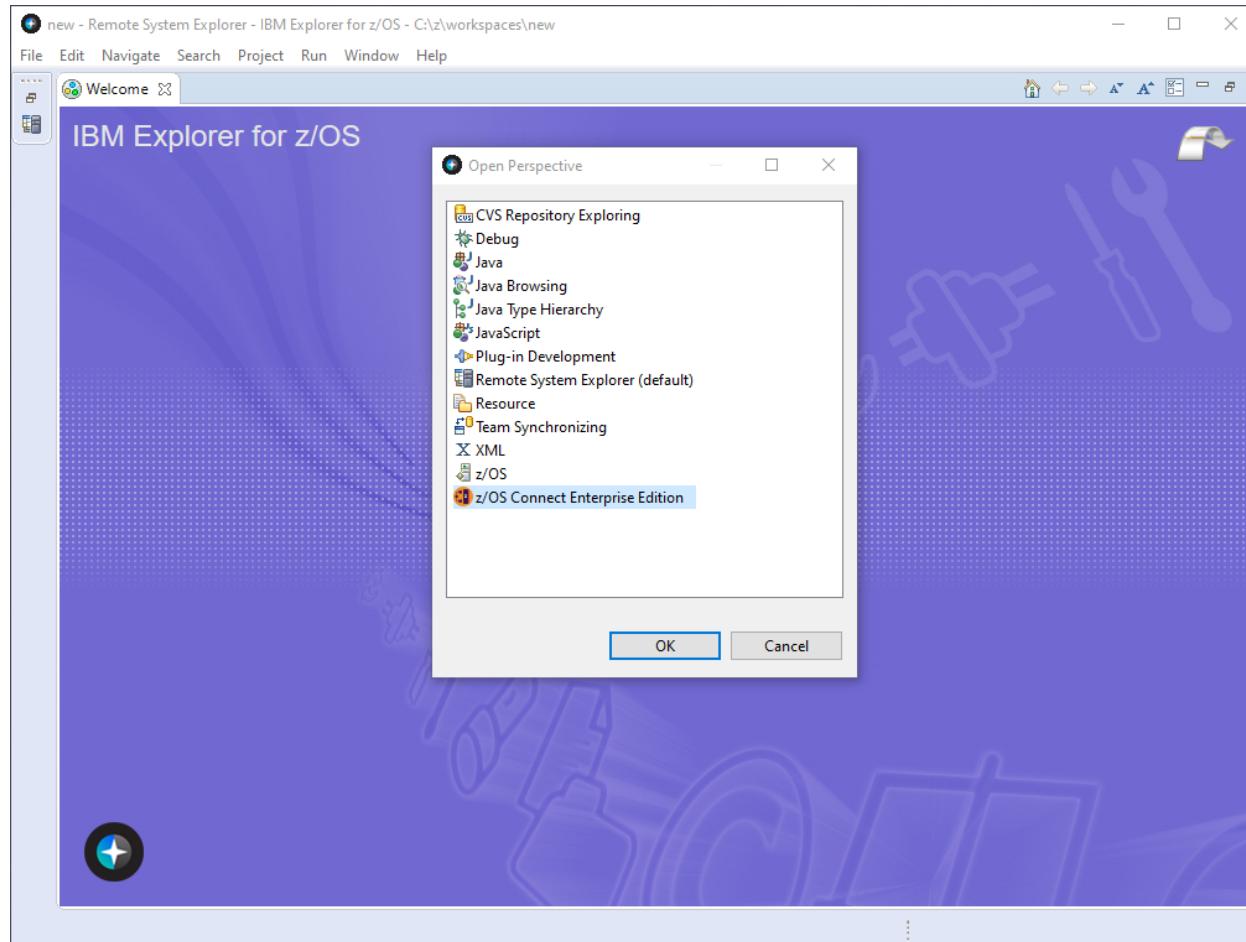
/api_toolkit/services

Simple **service creation.**

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



z/OS Connect EE



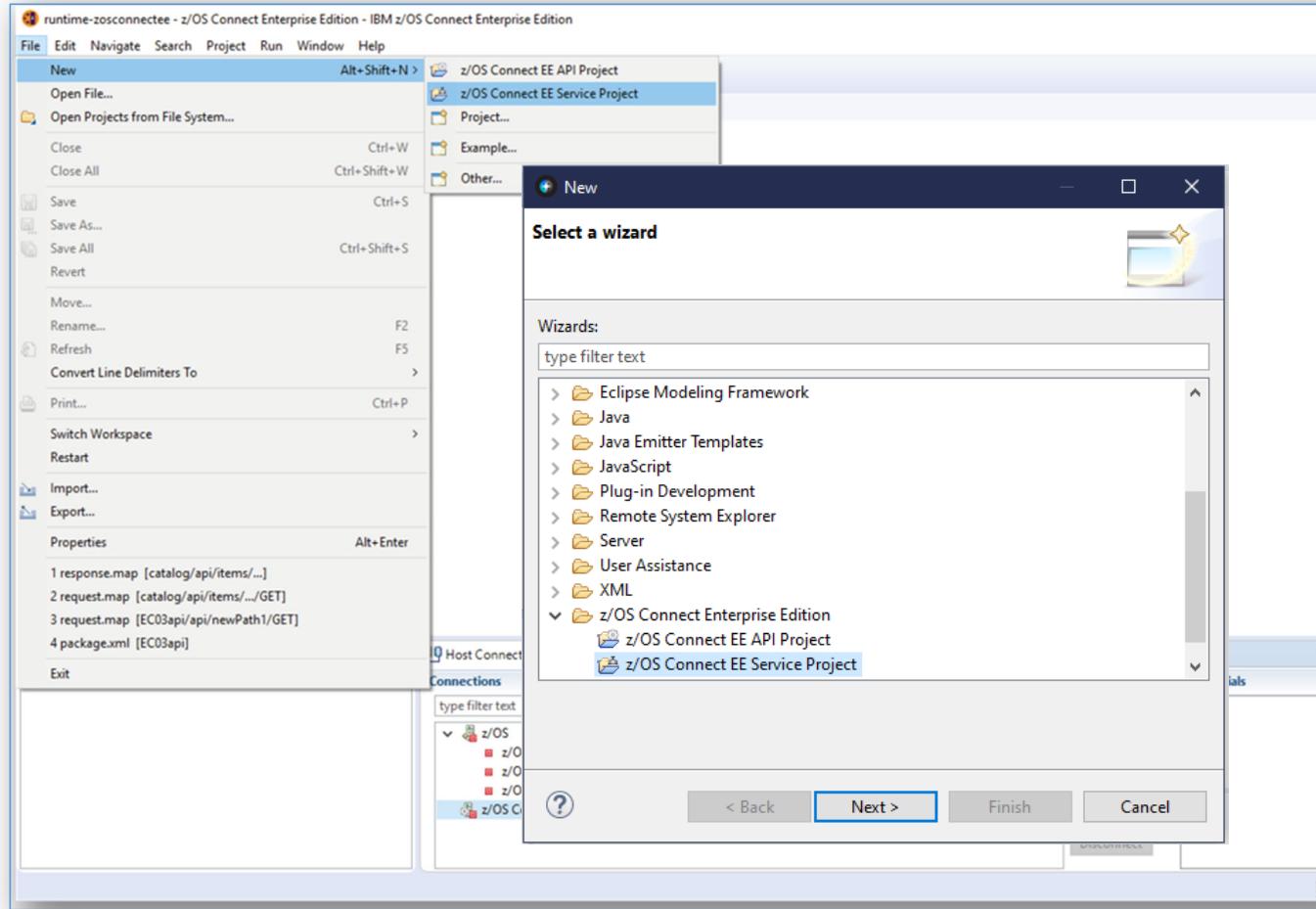
Use the **API toolkit** to create services through Eclipse-based tooling.

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

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



z/OS Connect EE

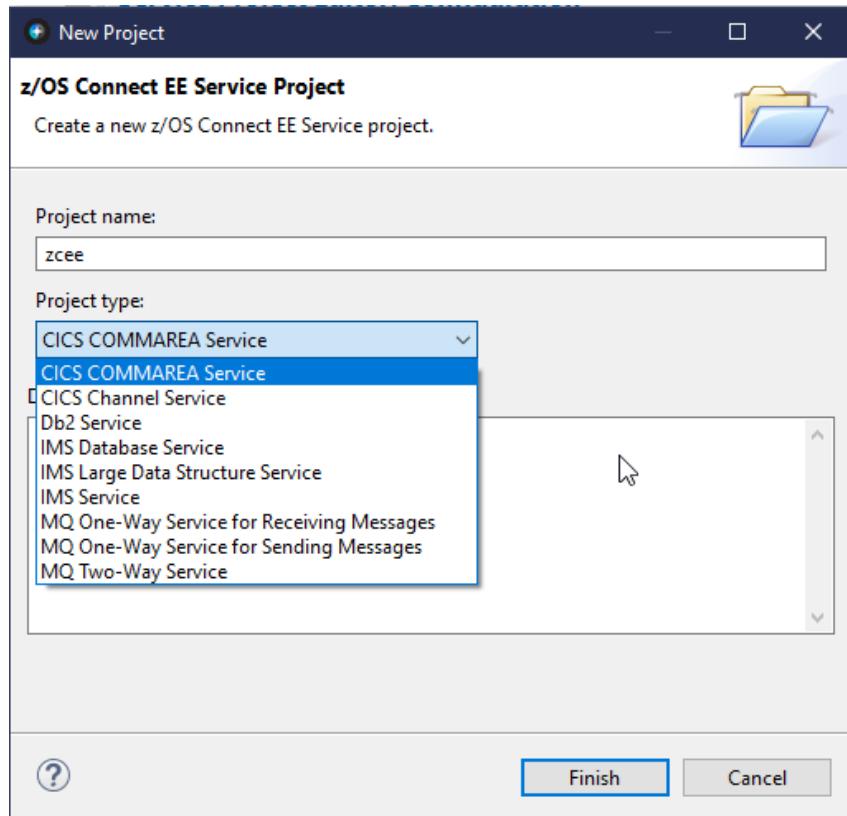


Use the **API toolkit** to create services through Eclipse-based tooling.

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

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

Service creation – a common interface

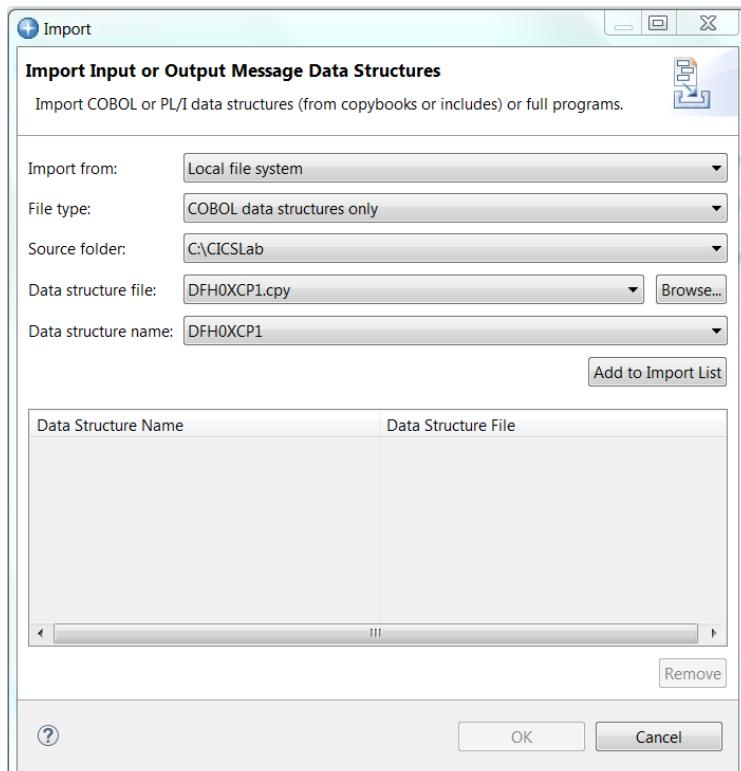


A common interface for service creation, irrespective of back-end subsystem.

- 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

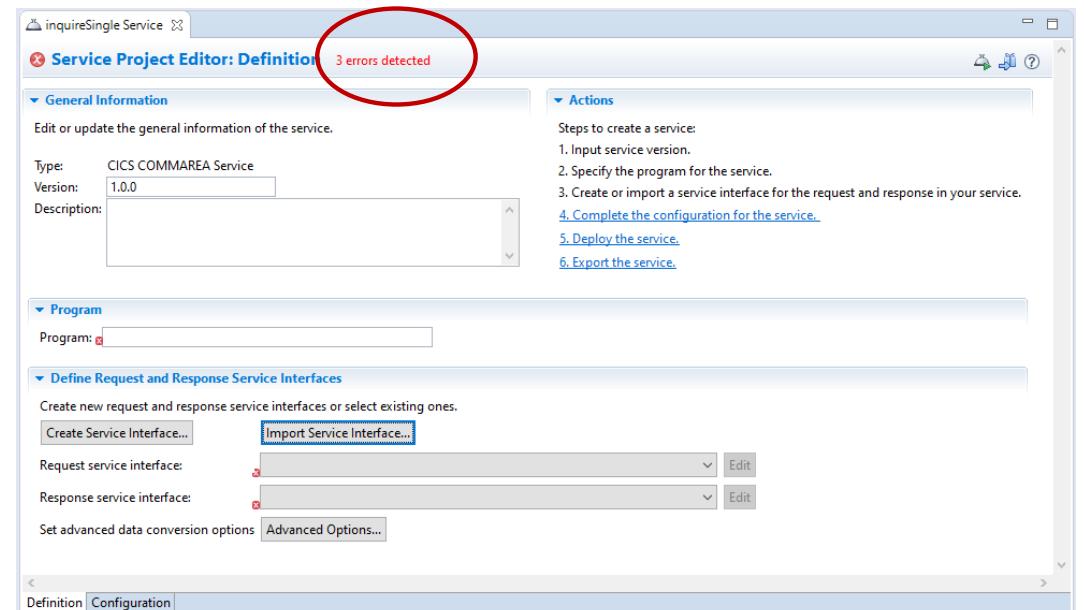
API toolkit – Creating Services for CICS, IMS TM and MQ

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.

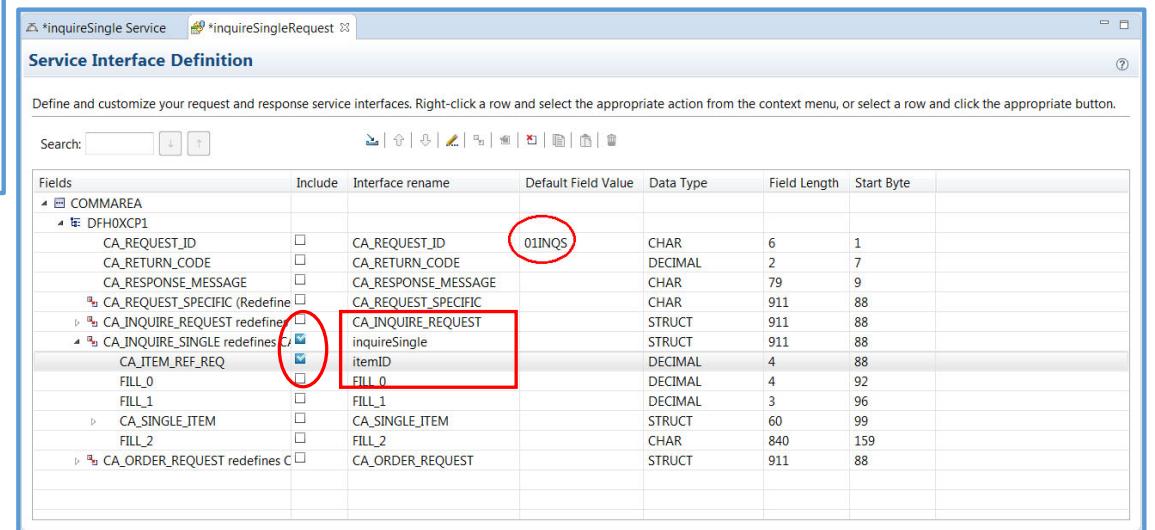


API toolkit – Creating Services for CICS, IMS TM and MQ

Allows editing a request service interface definition

```
-----*
* Check which operation is 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.



The screenshot shows a 'Service Interface Definition' window with two tabs: '*inquireSingle Service' and '*inquireSingleRequest'. The table below lists fields with various actions available for each row.

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFHXCPI						
CA_REQUEST_ID	<input type="checkbox"/>	CA_REQUEST_ID	01INQS	CHAR	6	1
CA_RETURN_CODE	<input type="checkbox"/>	CA_RETURN_CODE		DECIMAL	2	7
CA_RESPONSE_MESSAGE	<input type="checkbox"/>	CA_RESPONSE_MESSAGE		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefine)	<input type="checkbox"/>	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefine	<input checked="" type="checkbox"/>	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines CA_INQUIRE_REQUEST	<input checked="" type="checkbox"/>	inquireSingle		STRUCT	911	88
CA_ITEM_REF_REQ	<input checked="" type="checkbox"/>	itemID		DECIMAL	4	88
FILL_0	<input type="checkbox"/>	FILL_0		DECIMAL	4	92
FILL_1	<input type="checkbox"/>	FILL_1		DECIMAL	3	96
CA_SINGLE_ITEM	<input type="checkbox"/>	CA_SINGLE_ITEM		STRUCT	60	99
FILL_2	<input type="checkbox"/>	FILL_2		CHAR	840	159
CA_ORDER_REQUEST redefines CA_INQUIRE_REQUEST	<input type="checkbox"/>	CA_ORDER_REQUEST		STRUCT	911	88

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.

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

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



API toolkit – Creating Services for CICS

Creating multiple services definitions to the same resource

The screenshot shows the Service Interface Editor with two tabs: "cscvincSelectService Service" and "cscvincSelectRequest". The "cscvincSelectRequest" tab is active. It displays a table of fields with columns: Fields, Include, Interface Rename, Default Field Value, Data Type, and Field Length. A red circle highlights the "Default Field Value" column for the "ACTION" field, which is set to "S".

The screenshot shows the Service Interface Editor with two tabs: "cscvincSelectService Service" and "cscvincSelectRequest". The "cscvincSelectRequest" tab is active. It displays a table of fields with columns: Fields, Include, Interface Rename, Default Field Value, Data Type, and Field Length. A red circle highlights the "Default Field Value" column for the "ACTION" field, which is set to "I".

The screenshot shows the Service Project Editor: Definition for a service named "cscvincSelectService Service". The "Program" field is set to "CSCVINC", highlighted by a red circle. The "Actions" panel on the right lists steps for creating a service, including "Input service version", "Specify the program for the service", and "Complete the configuration for the service".

```
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

© 2018, 2022 IBM Corporation

Accessing a CICS program – Transaction ID Usage



z/OS Connect EE

Service Project Editor: Configuration

Required Configuration
Enter the required configuration for this service.
Coded character set identifier (CCSID): 37
Connection reference: cscvinc

Optional Configuration
Enter the optional configuration for this service.
Transaction ID: MUO
Transaction ID usage: EIB_AND_MIRROR (highlighted by a red circle)
Bidi configuration reference:
Use context containers:
Context containers HTTP headers:
Add another

EIB_ONLY

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: CSMI PROGRAM: DFHMIRS TASK: 0008501 APPLID: CICS53Z DISPLAY: 00
STATUS: PROGRAM INITIATION
EIBTIME = 104730
EIBDATE = 0122050
EIBTRNID = 'CSMI'
EIBTASKN = 8501
EIBTRMID = '/RBX'
EIBCPSON = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0000'
EIBRCODE = X'000000000000
EIBDS = .....
EIBREQID = .....
ENTER: CONTINUE PF1 : UNDEFINED PF2 :
PF4 : SUPPRESS DISPLAYS PF5 :
PF7 : SCROLL BACK PF8 :
PF10: PREVIOUS DISPLAY PF11:
M0 D
Connected to remote server/host wg31a using lu/pool TCP0012
```

EIB_AND_MIRROR

```
WG31 - 3270
File Edit Settings View Communication Actions W
TRANSACTION: MIJO PROGRAM: DFH STATUS: PROGRAM INITIATION
EIBTIME = 109914
EIBDATE = 0122050
EIBTRNID = 'MIJO'
EIBTASKN = 8492
EIBTRMID = '/RBX'
EIBCPSON = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0000'
EIBRCODE = X'000000000000
EIBDS = .....
EIBREQID = .....
ENTER: CONTINUE PF1 : UNDEFINED PF2 :
PF4 : SUPPRESS DISPLAYS PF5 :
PF7 : SCROLL BACK PF8 :
PF10: PREVIOUS DISPLAY PF11:
M0 D
Connected to remote server/host wg31a using lu/pool TCP0012
```

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: DFH MIRRS TASK: 0008476 APPLID: CICS53Z DISPLAY: 00
STATUS: ABOUT TO EXECUTE COMMAND
EXEC CSMI LINK PROGRAM
PROGRAM ('CSCVINC')
SYNCONRETURN
CHANNEL ('Channel')
NOHANDLE
EIBTIME = 181730
EIBDATE = 0122051
EIBTRNID = 'MIJO'
EIBTASKN = 8837
EIBTRMID = '/RBX'
EIBCPSON = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0E02' LINK
EIBRCODE = X'000000000000
EIBDS = .....
EIBREQID = .....
ENTER: CONTINUE PF1 : UNDEFINED PF2 :
PF4 : SUPPRESS DISPLAYS PF5 :
PF7 : SCROLL BACK PF8 :
PF10: PREVIOUS DISPLAY PF11:
M0 D
Connected to remote server/host wg31a using lu/pool TCP0012
```

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008837 APPLID: CICS53Z
STATUS: PROGRAM INITIATION
EIBTIME = 181730
EIBDATE = 0122051
EIBTRNID = 'MIJO'
EIBTASKN = 8837
EIBTRMID = '/RBX'
EIBCPSON = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0E02' LINK
EIBRCODE = X'000000000000
EIBDS = .....
EIBREQID = .....
ENTER: CONTINUE PF1 : UNDEFINED PF2 :
PF4 : SUPPRESS DISPLAYS PF5 :
PF7 : SCROLL BACK PF8 :
PF10: PREVIOUS DISPLAY PF11:
M0 D
Connected to remote server/host wg31a using lu/pool TCP0012
```

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008949 APPLID: CICS53Z
STATUS: PROGRAM INITIATION
EIBTIME = 182613
EIBDATE = 0122051
EIBTRNID = 'MIJO'
EIBTASKN = 8949
EIBTRMID = '/RBX'
EIBCPSON = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0E02' LINK
EIBRCODE = X'000000000000
EIBDS = .....
EIBREQID = .....
ENTER: CONTINUE PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : END EDF
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DIS
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITI
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER T
M0 C
Connected to remote server/host wg31a using lu/pool TCP0012 and port 23
Adobe PDF on Document's.pdf
01/001
Connected to remote server/host wg31a using lu/pool TCP0012 and port 23
Adobe PDF on Document's.pdf
```

- **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

mitchj@us.ibm.com

These attributes also be used in `zosconnect_cicsIpConnection` and `zosconnect_services>service configuration` elements.

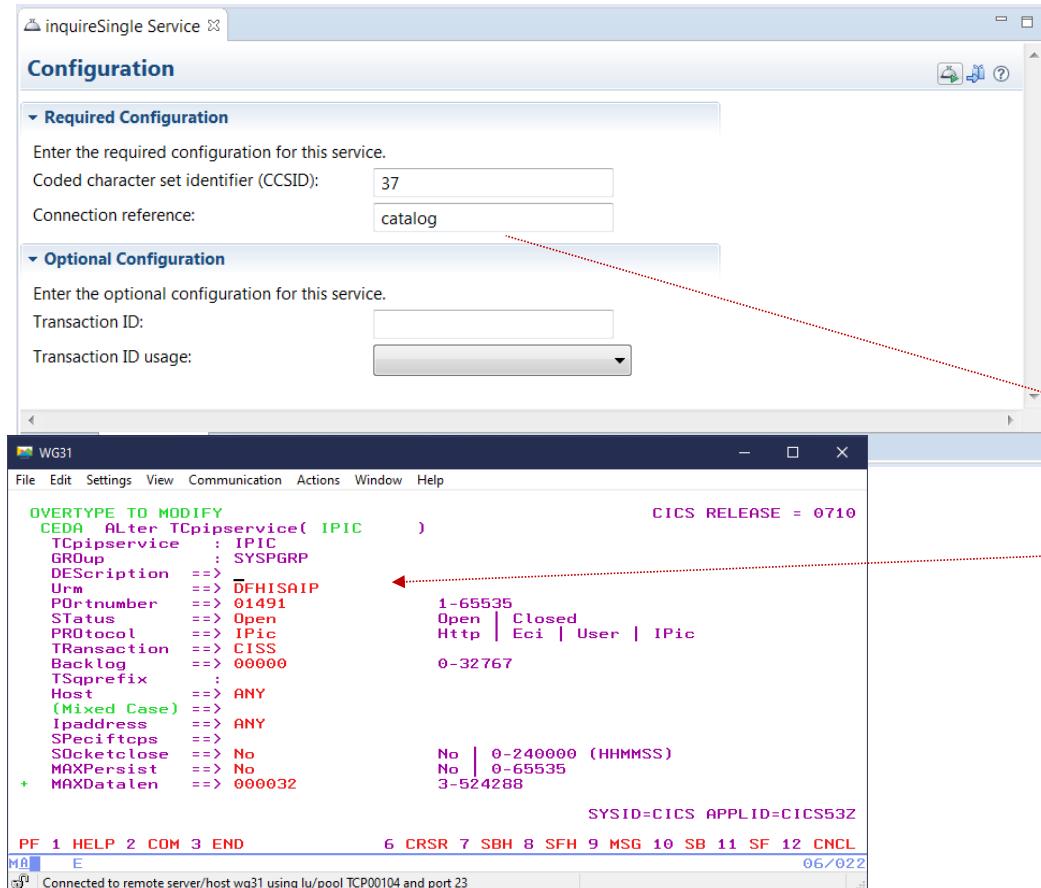
© 2018, 2022 IBM Corporation

Accessing a CICS program using IPIC



z/OS Connect EE

The server.xml file is the key configuration file:



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

```
catalog.xml
```

Design Source

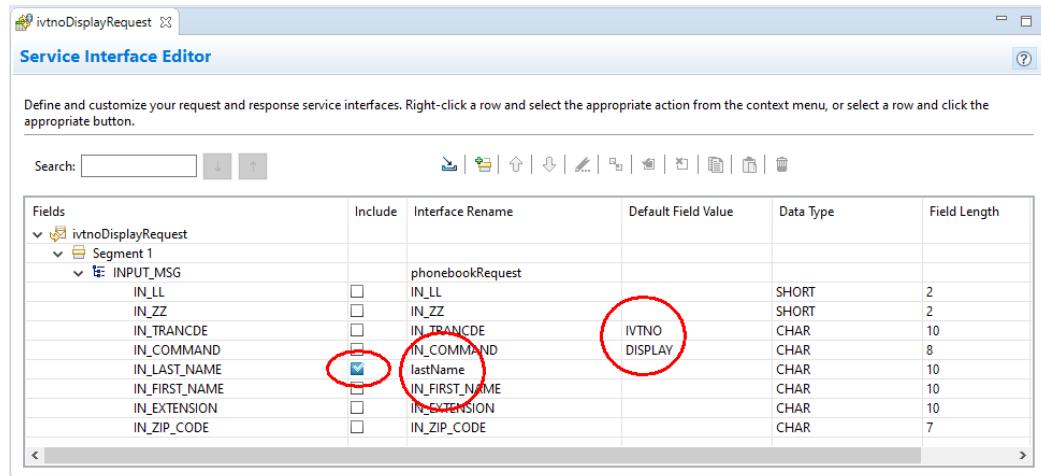
```
1<server description="CICS IPIC - catalog">
2
3<!-- Enable features -->
4<featureManager>
5  <feature>zosconnect:cicsService-1.0</feature>
6</featureManager>
7
8<zosconnect_cicsIpicConnection id="catalog">
9  host="wg31.washington.ibm.com"
10 port="1491"
11 transid="CSMI"
12 transidUsage="EIB_AND_MIRROR"/>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

Define IPIC connection to CICS

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
```

 ivtnoDisplayRequest Service Interface Editor

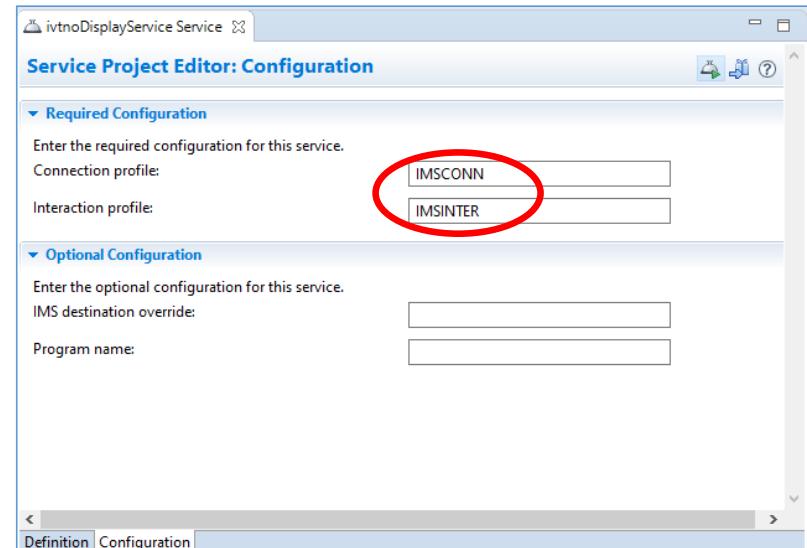
Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and click the appropriate button.

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
ivtnoDisplayRequest					
Segment 1					
INPUT_MSG		phonebookRequest			
IN_LL	<input type="checkbox"/>	IN_LL		SHORT	2
IN_ZZ	<input type="checkbox"/>	IN_ZZ		SHORT	2
IN_TRANCODE	<input type="checkbox"/>	IN_TRANCODE		CHAR	10
IN_COMMAND	<input checked="" type="checkbox"/>	IN_COMMAND	IVTNO DISPLAY	CHAR	8
IN_LAST_NAME	<input type="checkbox"/>	lastName		CHAR	10
IN_FIRST_NAME	<input type="checkbox"/>	IN_FIRST_NAME		CHAR	10
IN_EXTENSION	<input type="checkbox"/>	IN_EXTENSION		CHAR	10
IN_ZIP_CODE	<input type="checkbox"/>	IN_ZIP_CODE		CHAR	7

mitchj@us.ibm.com

The service developer creates distinct services for each function.

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



IMS/TM Meta Data

© 2018, 2022 IBM Corporation

IMS Connections and Interactions (server XML)



z/OS Connect EE

ivtnoService Service Configuration

Required Configuration

Enter the required configuration for this service.

Connection profile: **IMSCONN**

Interaction profile: **IMSINTER**

Optional Configuration

Enter the optional configuration for this service.

IMS destination override:

Program name:

Overview Configuration

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, T MEMBER=OTMAMEM)
IMSPLEX= (MEMBER=IMS14HWS, T MEMBER=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>

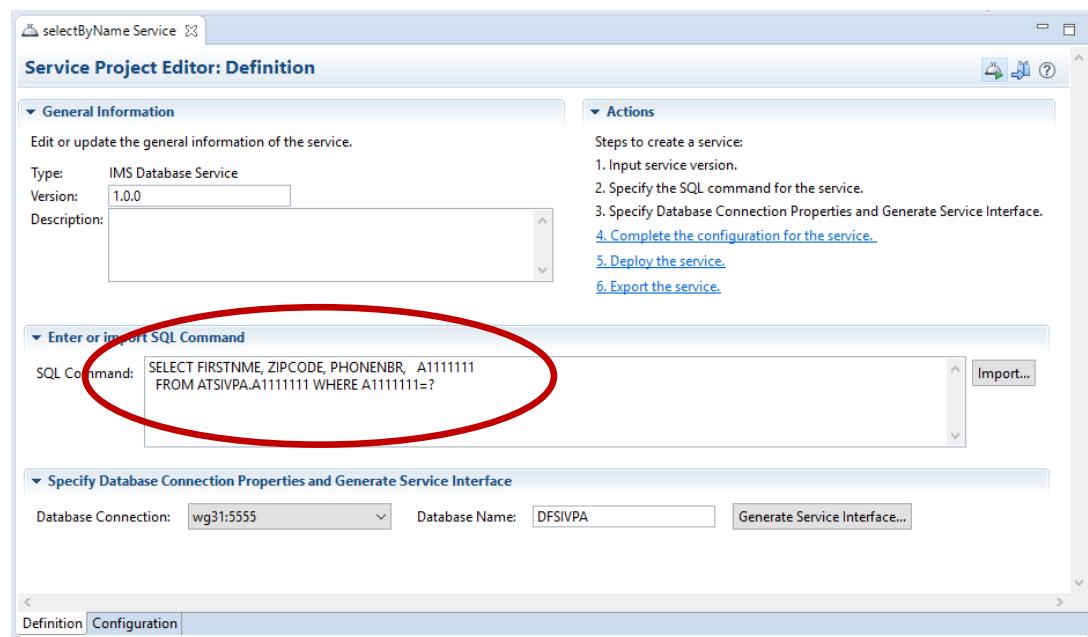
<authData id="Connection1_Auth" password="encryptedPassword1" user="userName1"/>
</server>
```

Interaction

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

API toolkit – Creating Services for IMS DB

Creating a service project from the IMS Catalog

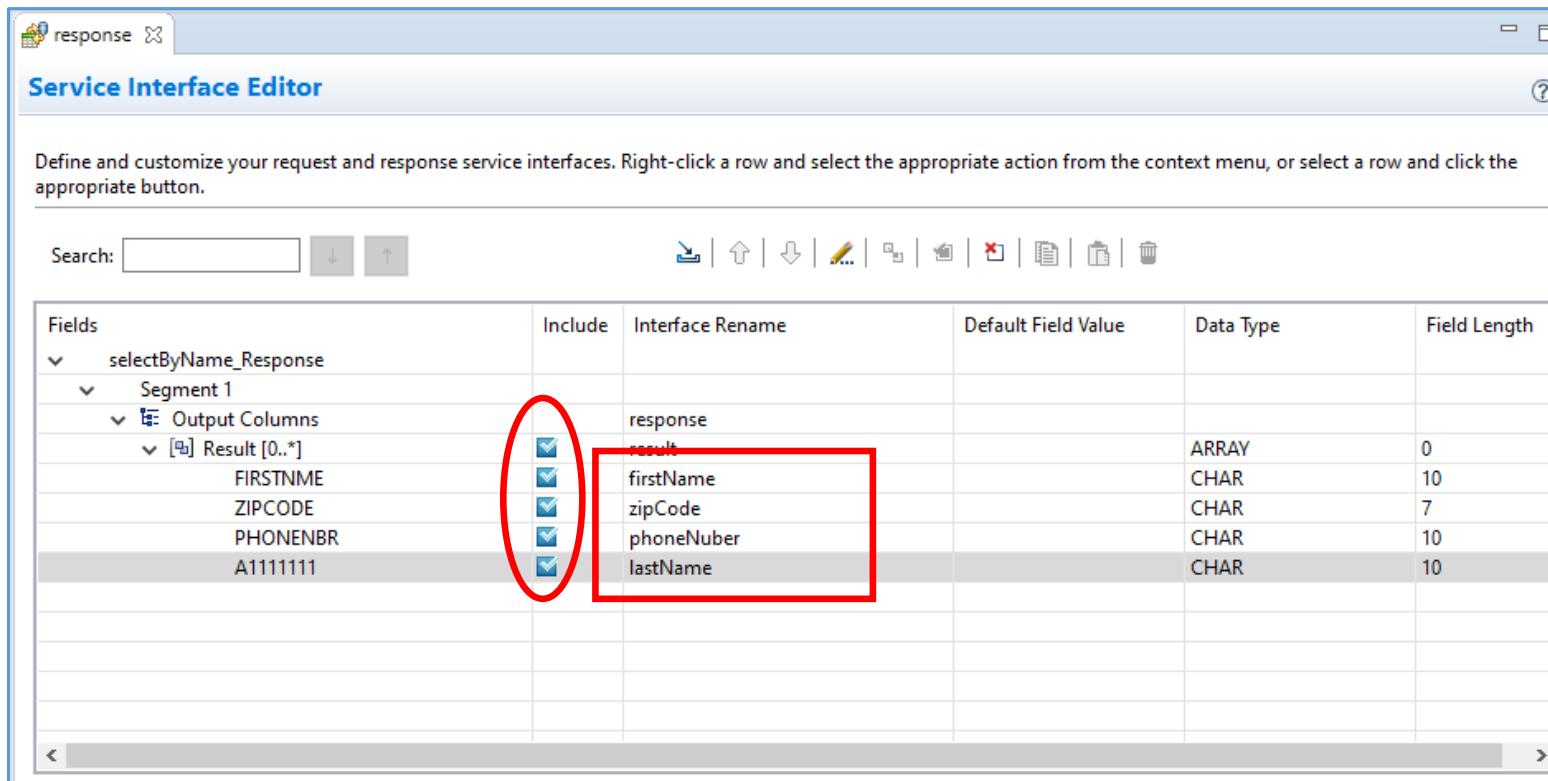


Use the IMS Catalog to assist with developing and testing SQL SELECT commands used for accessing IMS databases.

```
*-----*
* 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=DFSVVD1,DEVICE=3380,SIZE=2048
SEGM    NAME=A1111111,PARENT=0,BYTES=40,RULES=(LLV,LAST),
        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
```

API toolkit – Creating Services for IMS DB

The Toolkit allows editing a service interface definitions*



The screenshot shows the Service Interface Editor window. The title bar says "Service Interface Editor". The main area has a heading "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." Below this is a search bar and a toolbar with icons for copy, paste, up, down, edit, etc. The main table has columns: Fields, Include, Interface Rename, Default Field Value, Data Type, and Field Length. The table data is as follows:

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
selectByName_Response					
Segment 1					
Output Columns					
Result [0..*]					
FIRSTNAME	<input checked="" type="checkbox"/>	response		ARRAY	0
ZIPCODE	<input checked="" type="checkbox"/>	result		CHAR	10
PHONENR	<input checked="" type="checkbox"/>	firstName		CHAR	7
A1111111	<input checked="" type="checkbox"/>	zipCode		CHAR	10
	<input checked="" type="checkbox"/>	phoneNuber		CHAR	10
	<input checked="" type="checkbox"/>	lastName		CHAR	10

*Using a slightly different process

IMS Connection Factory in the server XML



z/OS Connect EE

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection profile: DFSIVPACConn

ConnectionFactory

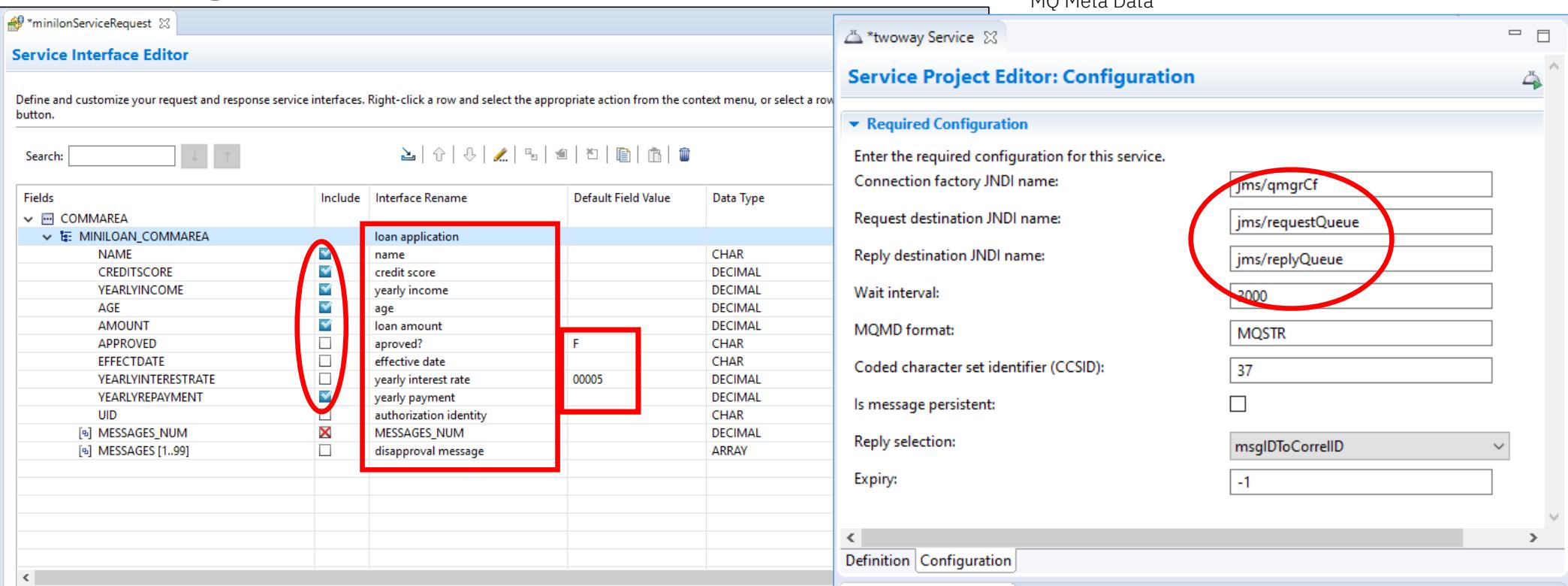
```
<connectionFactory id="DFSIVPACConn">
<properties.imsudbJLocal
  databaseName="DFSIVPA"
  datastoreName="IVP1"
  datastoreServer="wg31.washington.ibm.com"
  driverType="4"
  portNumber="5555"
  user="USER1"
  password="password"
  flattenTables="True"/>
</connectionFactory>
```

IMS Connect HWSCFG

```
HWS=(ID=IMS14HWS,XIBAREA=100,RACE=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=(ODBMAUTOCCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000), ODBMTMOT=6000)
```

API toolkit – Creating Services for MQ

Creating a “POST” service interface definition



The image shows two windows side-by-side:

- Service Interface Editor:** This window displays a table of fields for a service interface named "minilonServiceRequest". The table has columns: Fields, Include, Interface Rename, Default Field Value, and Data Type. A red box highlights the "Interface Rename" column for the "loan application" row, which contains the values "name", "credit score", "yearly income", "age", "loan amount", "aproved?", "effective date", "yearly interest rate", "yearly payment", "authorization identity", "MESSAGES_NUM", and "disapproval message". A red circle highlights the "Include" checkbox for the "loan application" row.
- Service Project Editor: Configuration:** This window shows configuration settings for a service named "twoway Service". It includes sections for Required Configuration, Connection factory JNDI name (jms/qmgrCf), Request destination JNDI name (jms/requestQueue), Reply destination JNDI name (jms/replyQueue), Wait interval (3000), MQMD format (MQSTR), Coded character set identifier (CCSID) (37), Is message persistent (unchecked), Reply selection (msgIDToCorrelID), and Expiry (-1). A red circle highlights the "Request destination JNDI name" field.

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.

Using JMS to access MQ (One-Way)



z/OS Connect EE

mqGetService Service

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection factory JNDI name:
jms/qmgrCf

Destination JNDI name:
jms/default

Coded character set identifier (CCSID):
37

Optional Configuration

Enter the optional configuration for this service.

Wait interval:

Message selector:

Definition Configuration

mqClient.xml

Read only Close

Design Source

```
<server description="MQ Service Provider">
<featureManager>
    <feature>zosconnect:mqService-1.0</feature>
</featureManager>
<variable name="wmqJmsClient.rar.location"
    value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
<wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
<zosconnect_services>
    <service name="mqPutService">
        <property name="useCallerPrincipal" value="false"/>
    </service>
</zosconnect_services>
<connectionManager id="ConMgr1" maxPoolSize="5"/>
<jmsConnectionFactory id="qmgrCF" jndiName="jms/qmgrCF">
    <connectionManagerRef>ConMgr1</connectionManagerRef>
    <properties.wmqJMS transportType="CLIENT"
        queueManager="ZMQ1"
        channel="LIBERTY.DEF.SVRCONN"
        hostname="wg31.washington.ibm.com"
        port="1422" />
</jmsConnectionFactory>
<jmsQueue id="q1" jndiName="jms/default">
    <properties.wmqJMS
        baseQueueName="ZCEE.DEFAULT.MQZCEE.QUEUE"
        CCSID="37"/>
</jmsQueue>
</server>
```

Using JMS to access MQ (Two-Way)

*twoWay Service X

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection factory JNDI name:

Request destination JNDI name:

Reply destination JNDI name:

Wait interval:

MQMD format:

Coded character set identifier (CCSID):

Is message persistent:

Reply selection:

Expiry:

Definition Configuration

mq.xml

Design Source

```

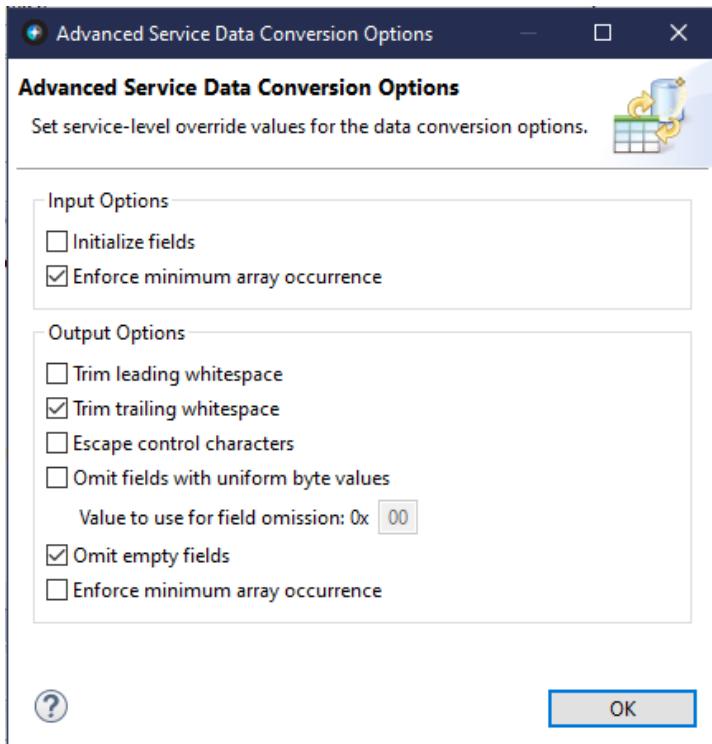
2 <featureManager>
3   <feature>zosconnect:mqService-1.0</feature>
4 </featureManager>
5
6 <variable name="wmqJmsClient.rar.location"
7   value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
8 <wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
9
10 <connectionManager id="ConMgr1" maxPoolSize="5"/>
11
12 <jmsConnectionFactory id="qmgrCF" jndiName="jms/qmgrCf"
13   connectionManagerRef="ConMgr1">
14   <properties.wmqJms transportType="BINDINGS"
15     queueManager="QMZ1" />
16 </jmsConnectionFactory>
17
18 <jmsConnectionFactory id="qmgrCF2" jndiName="jms/qmgrCF2"
19   connectionManagerRef="ConMgr1">
20   <properties.wmqJms transportType="CLIENT"
21     queueManager="ZMQ1"
22     channel="LIBERTY.DEF.SVRCONN"
23     hostName="wg31.washington.ibm.com"
24     port="1422" />
25 </jmsConnectionFactory>
26
27 <jmsQueue id="q1_jndiName="jms/default">
28   <properties.wmqJms
29     baseQueueName="ZCONN2.DEFAULT.MQZEE.QUEUE"
30     CCSID="37"/>
31 </jmsQueue>
32
33 <jmsQueue id="requestQueue" jndiName="jms/request">
34   <properties.wmqJms
35     baseQueueName="ZCONN2.TRIGGER.REQUEST"
36     targetClient="MQ"
37     CCSID="37"/>
38 </jmsQueue>
39
40 <jmsQueue id="replyQueue" jndiName="jms/replyQueue">
41   <properties.wmqJms
42     baseQueueName="ZCONN2.TRIGGER.RESPONSE"
43     targetClient="MQ"
44     CCSID="37"/>
45 </jmsQueue>
46
47

```

API toolkit – Advanced Data Conversion Options



z/OS Connect EE



Request Messages:

- Initialize fields
- Enforce minimum array occurrence

Response Messages:

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

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", FIRSTNAME 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')
```

 Import Db2 service from service manager

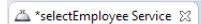
Db2 service manager connection:  wg31:2446

Type to search...

Service Name	Version	Collection ID	Description
selectEmployee		SYSIBMSERVICE	Select an employee from table USER1.EMPLOYEE
deleteEmployee		zCEEService	Delete an employee from table USER1.EMPLOYEE
displayEmployee		zCEEService	Display an employee in table USER1.EMPLOYEE
insertEmployee		zCEEService	Insert an employee into table USER1.EMPLOYEE
selectByDepartments		zCEEService	Select employees by departments
selectByRole		zCEEService	Select an employee based on job and department
selectEmployee	V1	zCEEService	Select an employee from table USER1.EMPLOYEE
selectEmployee	V2	zCEEService	Select an employee from table USER1.EMPLOYEE
updateEmployee		zCEEService	Update an employee in table USER1.EMPLOYEE

Definition Configuration

Import Cancel

 *selectEmployee Service

Service Project Editor: Definition

General Information

Edit or update the general information of the service.

Type: Db2 Service
Version: 1.0.0
Description:

Actions

Steps to create a service:

1. Input service version.
2. Import JSON schemas from a Db2 service manager or your local machine.
3. Complete the configuration for the service.
4. Deploy the service.
5. Export the service.

Define Db2 service

Import a Db2 native REST service from a Db2 service manager. Alternatively, enter your Db2 service details and import the JSON schemas from your local machine.

[Import from Db2 service manager...](#)

Collection Id: SYSIBMSERVICE
Db2 native REST service name: selectByRole
Db2 native REST service version: V1
Request JSON schema: request-schema.json [Import from local machine...](#)
Response JSON schema: response-schema.json [Import from local machine...](#)

The service developer retrieves details about the Db2 REST services

Note there is no service interface editor available

Accessing a Db2 REST service resource



z/OS Connect EE

The screenshot shows the Service Project Editor for a project named "selectEmployee Service". The "Configuration" tab is selected. In the "Required Configuration" section, there is a "Connection reference:" field containing "db2conn". A red arrow points from this field to the "db2pass.xml" file in the "Source" tab.

The "db2pass.xml" file contains the following XML code:

```
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
```

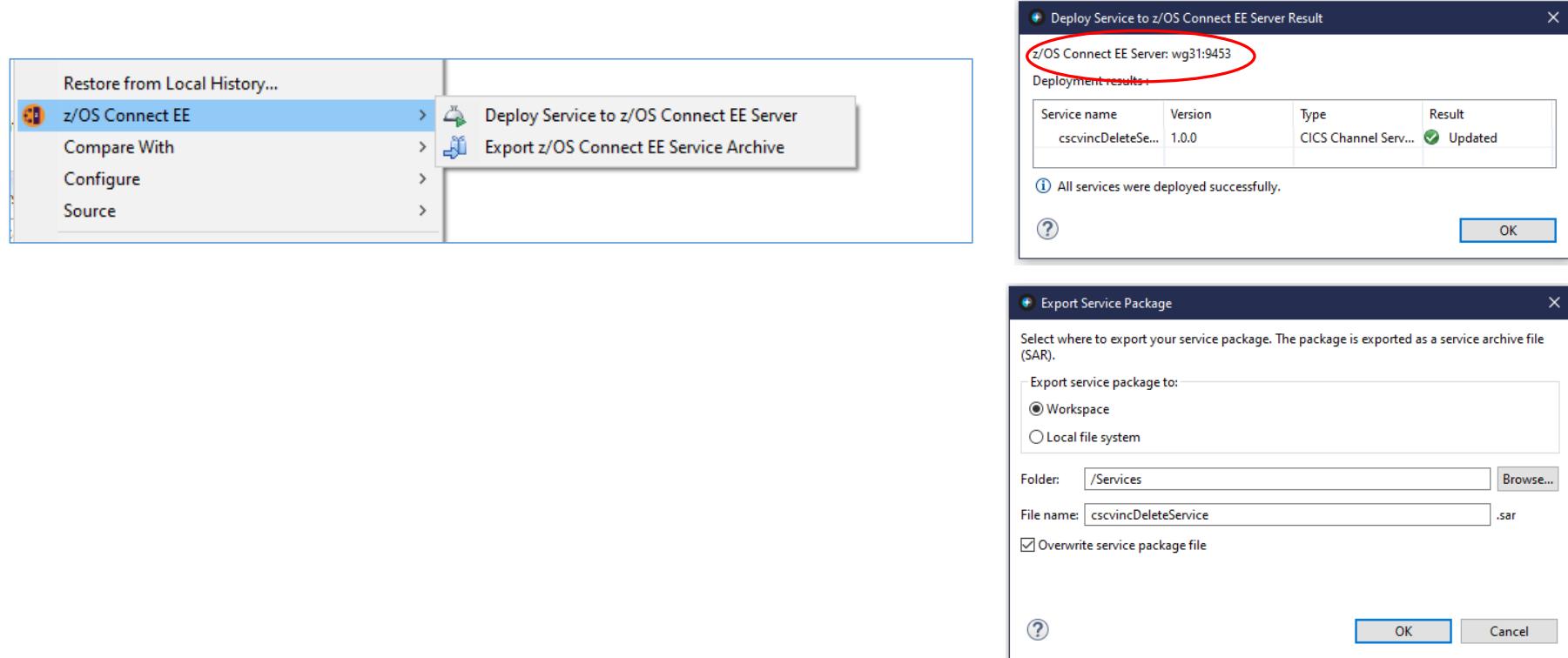
On the left side of the editor, there is a log output window showing the following messages:

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

API toolkit – Services Editor

Server connection and Services deployment

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



The screenshot shows the API toolkit interface with the "z/OS Connect EE" server selected in the "Host Connections" view. A context menu is open, showing options: "Deploy Service to z/OS Connect EE Server" and "Export z/OS Connect EE Service Archive".

Deploy Service to z/OS Connect EE Server Result

z/OS Connect EE Server: wg31:9453

Deployment results:

Service name	Version	Type	Result
cscvincDeleteSe...	1.0.0	CICS Channel Serv...	Updated

All services were deployed successfully.

OK

Export Service Package

Select where to export your service package. The package is exported as a service archive file (SAR).

Export service package to:

Workspace

Local file system

Folder: /Services

File name: cscvincDeleteService.sar

Overwrite service package file

OK Cancel



/api_toolkit/services

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

Use HATS to capture screen flows and input/output fields

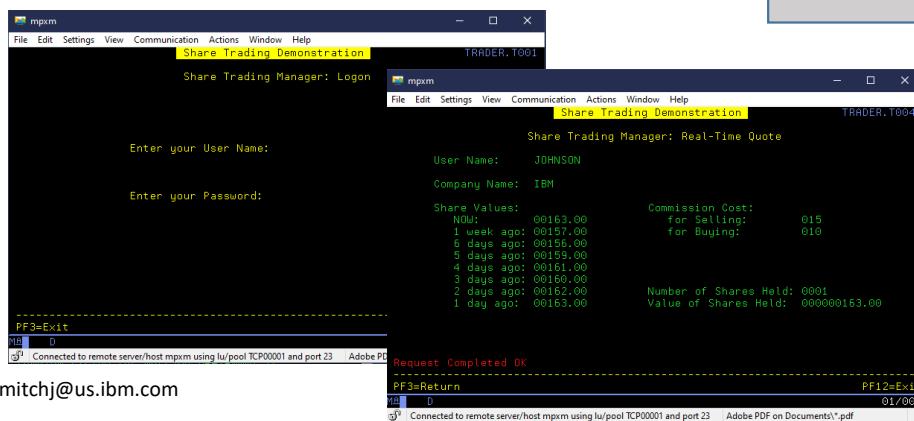
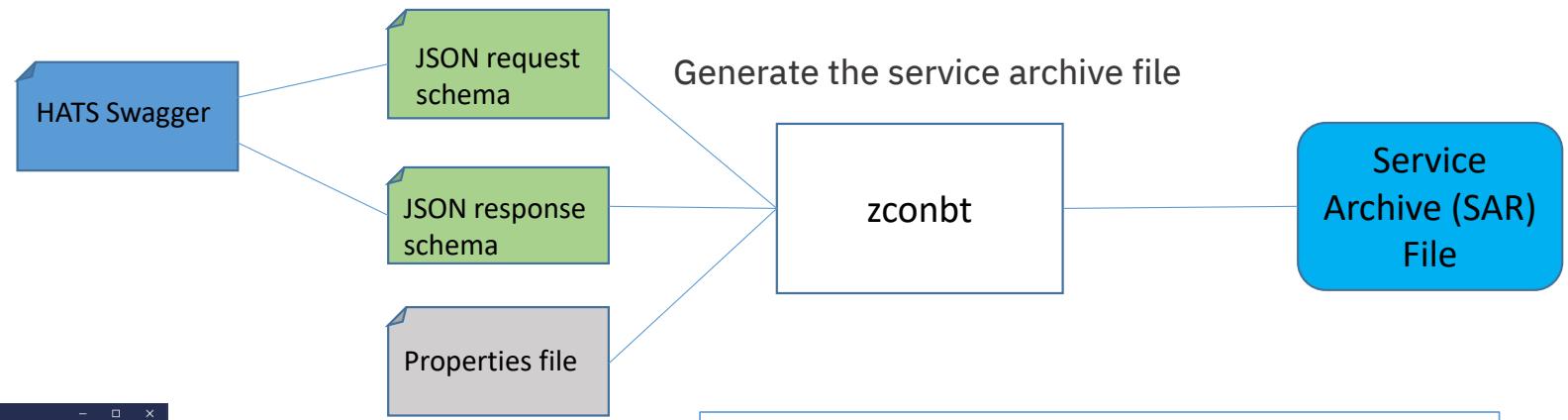


The screenshot shows the Rational Application Developer for WebSphere Software interface with the following components:

- Left Panel:** Project Explorer showing a tree structure for a "Trader" project with various macros like "buyShares", "connect", "disconnect", "getCompany", "getQuoteValue", and "sellShares".
- Middle Panel:** Editor window displaying the Java code for `BuyShares.java`. The code handles methods such as setPassword, setCompany, setNumberofShares, setMessage, and getUserName.
- Right Panel:** A "Host Terminal" window showing a terminal session with the Washington Systems Center login screen. The session includes commands like "allowTracing()", "Ras.traceEntry()", and "Ras.trace(class, method)".
- Bottom Right Window:** A "Create a Screen Customization - Screen6" dialog box. It displays "Screen Recognition Criteria" for identifying the terminal screen. The "Screen" dropdown is set to "buyShares/Screen6". The configuration includes:
 - Select a screen source:** Radio button selected for "Use the host terminal screen".
 - String:** "USER1" is highlighted in the "String" field.
 - String position:** Radio button selected for "Anywhere on the screen".
 - Region:** "Within a rectangular region" is selected. The "Start row" is 7, "Start column" is 33, "End row" is 7, and "End column" is 38.
 - Attributes:** Checkboxes for "Case sensitive", "Optional", and "Invert" are available but not checked.

Command line(zconbt) – REST Services

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
  
```

HATS server XML configuration



z/OS Connect EE

```
getCompany.properties - Notepad
File Edit Format View Help
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
```

Server Config

hats.xml

Read only Close

Design Source

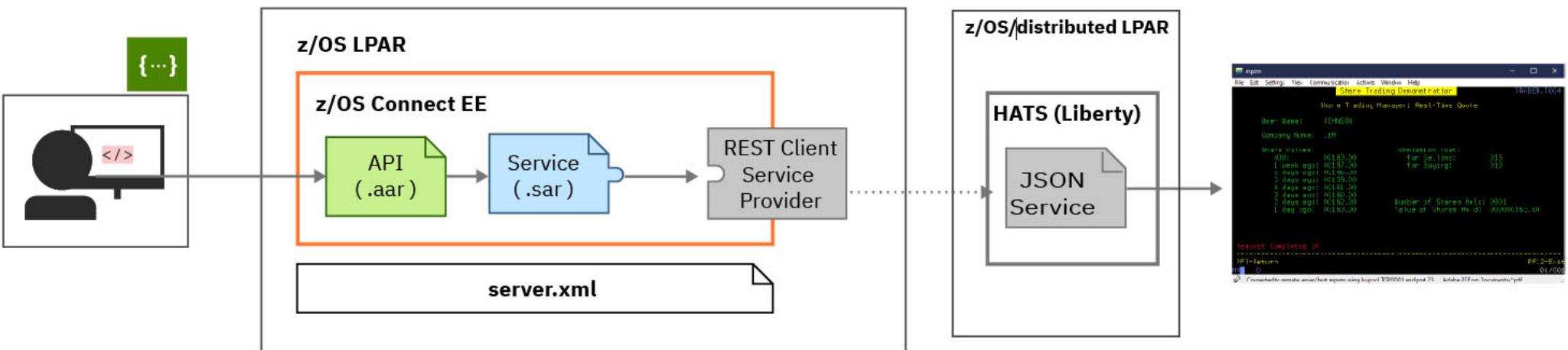
```
<server description="HATS">
  <zosconnect_zosConnectServiceRestClientConnection id="HatsConn"
    host="wg31.washington.ibm.com"
    port="29080" />
</server>
```

HATS Liberty server.xml

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



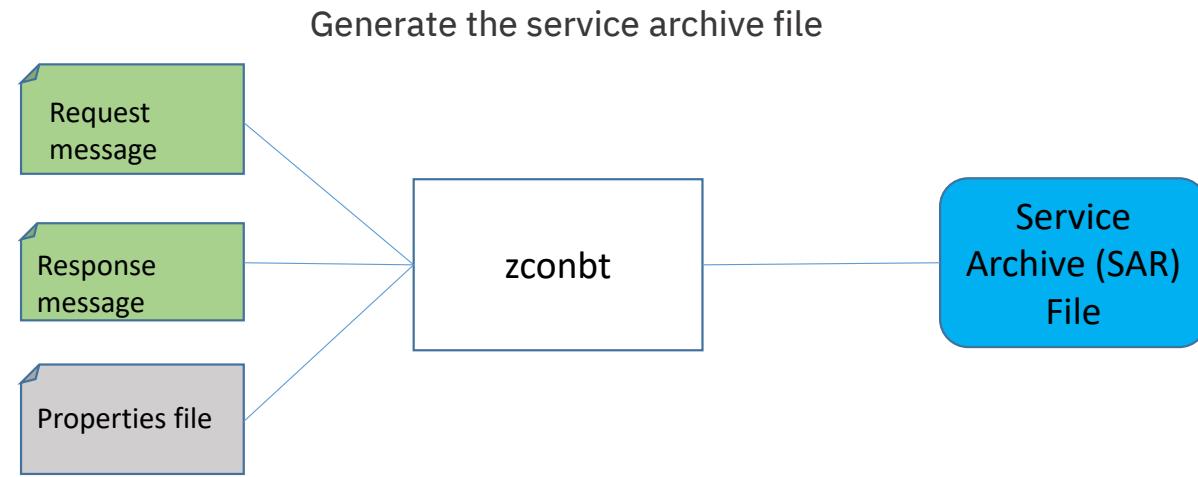
Using HATS to access a 3270 application



Command line(zconbt) – MVS Batch

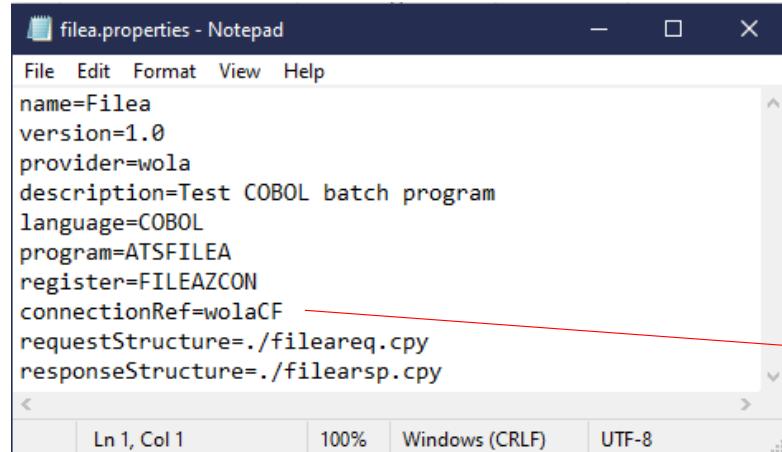
For batch WOLA services use the z/OS Connect Build toolkit (zconbt)

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

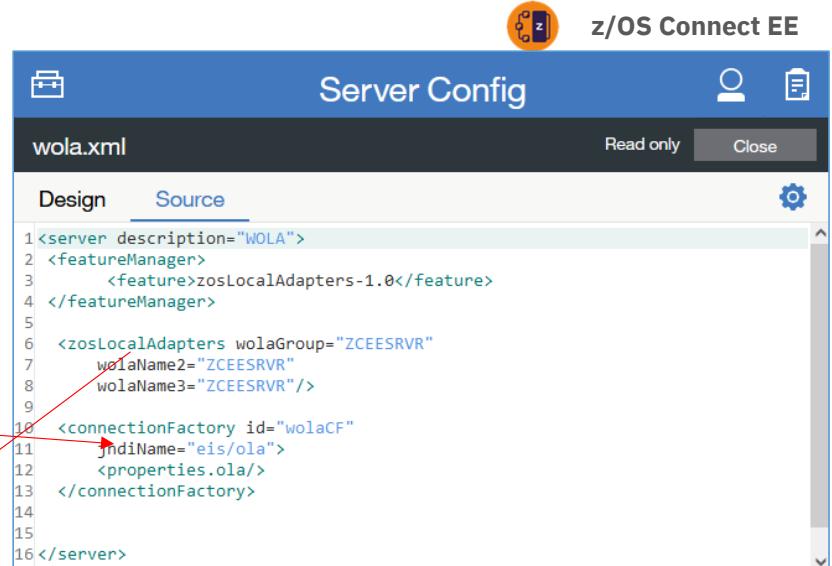


WebSphere Optimized Local Adapter – a protocol for cross memory communications between address spaces

MVS batch server XML



```
filea.properties - Notepad
File Edit Format View Help
name=Filea
version=1.0
provider=wola
description=Test COBOL batch program
language=COBOL
program=ATSFIL
register=FILEAZCON
connectionRef=wolaCF
requestStructure=./fileareq.cpy
responseStructure=./filearsp.cpy
```



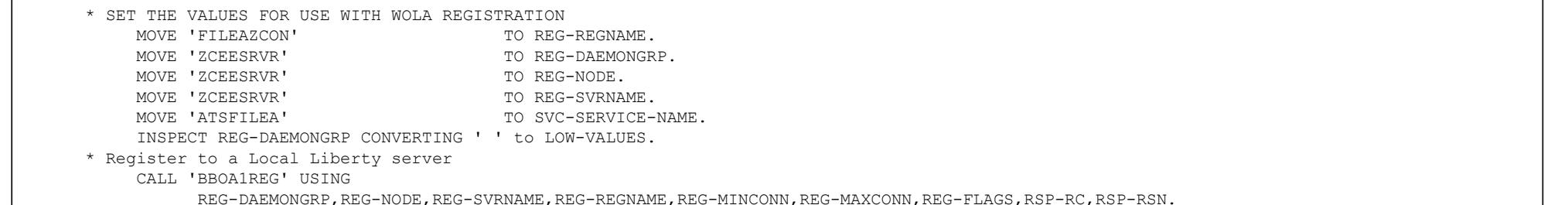
z/OS Connect EE

Server Config

wola.xml

Design Source

```
<server description="WOLA">
  <featureManager>
    <feature>zosLocalAdapters-1.0</feature>
  </featureManager>
  <zosLocalAdapters wolaGroup="ZCEESRVR">
    wolaName2="ZCEESRVR"
    wolaName3="ZCEESRVR"/>
  <connectionFactory id="wolaCF">
    jndiName="eis/ola">
      <properties.ola/>
    </connectionFactory>
  </server>
```

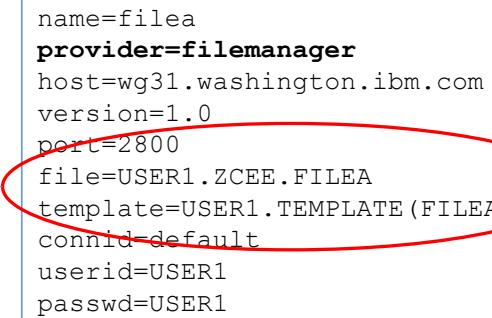
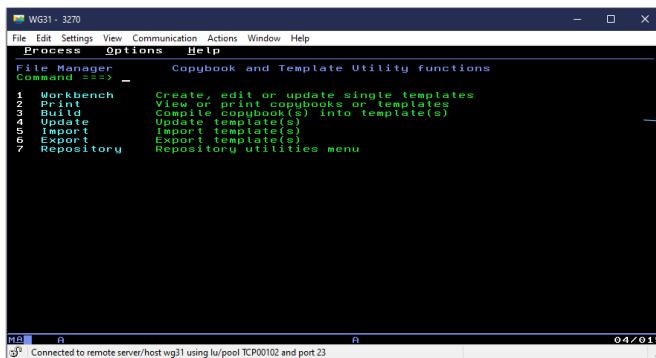
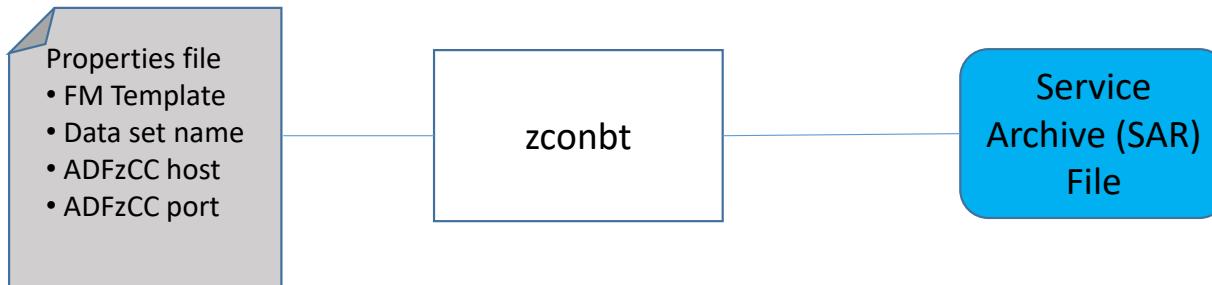


```
* SET THE VALUES FOR USE WITH WOLA REGISTRATION
MOVE 'FILEAZCON'          TO REG-REGNAME.
MOVE 'ZCEESRVR'            TO REG-DAEMONGRP.
MOVE 'ZCEESRVR'            TO REG-NODE.
MOVE 'ZCEESRVR'            TO REG-SVRNAME.
MOVE 'ATSFIL'              TO SVC-SERVICE-NAME.
INSPECT REG-DAEMONGRP CONVERTING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

Command line(zconbt) – File Manager

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
```

File Manager server XML



z/OS Connect EE

```
filea.properties - Notepad
File Edit Format View Help
name=filea
provider=filemanager
host=wg31.washington.ibm.com
version=1.0
port=2800
file=USER1.ZCEE.FILEA
template=USER1.ZCEE.TEMPLATE(FILEA)
connid=default
userid=USER1
passwd=USER1
```

Server Config

filemgr.xml

Read only Close

Design Source

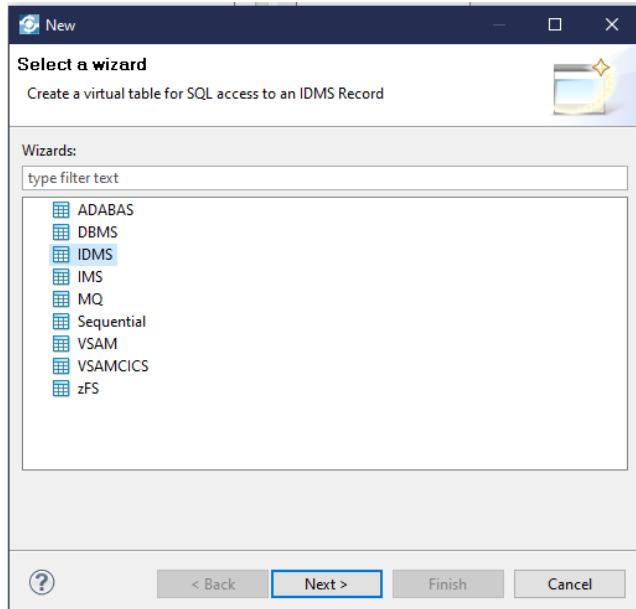
```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
  <!-- Enable features -->
  <featureManager>
    <feature>filemanager:fmProvider-2.0</feature>
  </featureManager>
  <FileManager_Connection id="default">
    <runport>2800</runport>
    <max_timeout>1800</max_timeout>
  </FileManager_Connection>
</server>
```

SYS1.PROCLIB(IPVSRV1)

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

DVM Studio

For DVM use the DVM Studio



The screenshot shows the DVM Studio interface. The title bar says 'DV Data - Data Virtualization Manager/SQL Samples/Generated.sql - IBM Data Virtualization Manager for z/OS'. The top menu includes File, Edit, Navigate, Search, Project, SQL, Run, Window, Help. The toolbar has various icons. The left sidebar has sections for Data (Navigator, JDBC), Data Sources (Edit SQL), and Services (Web Services). Under Services, 'Web Services' is expanded, showing options like /REST/, INSERT, UPD, Set Tree Filter, Target System, WSC, and Admin. A context menu is open over the 'INSERT' option, with 'Generate SAR File(s)' highlighted and circled in red. The central pane displays a SQL script named 'Generated.sql' with several INSERT and UPDATE statements. The bottom right pane shows a table with data from the query results, with a 'Generate SAR File(s)' button also circled in red.

WS_DESCRIPTION	WS_DEPARTMENT	WS_C
Mitch Johnson	10	002.
Mitch Johnson	10	002.
	10	002.

DVM server XML



z/OS Connect EE

Server Config

dvs.xml

Read only Close

Design Source

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<!-- Enable features -->
<featureManager>
<feature>usr:dvsProvider</feature>
<feature>zosLocalAdapters-1.0</feature>
</featureManager>
<!-- Adapter Details with WOLA Group Name (ZCEEDVM) -->
<zosLocalAdapters wolaName3="NAME3"
wolaName2="NAME2"
wolaGroup="ZCEEDVM"/>
<!-- DVS Service Details with Register Name (ZCEEDVM) -->
<zosconnect_zosConnectService invokeURI="/dvs"
serviceDescription=""
serviceRef="dvsService"
serviceName="dvsService"
id="zosConnectDvsService"/>
<usr_dvsService invokeURI="/dvs"
serviceName="DVSS1"
registerName="ZCEEDVM"
connectionFactoryRef="wolaCF"
id="dvsService"/>
<connectionFactory jndiName="eis/ola" id="wolaCF">
<properties.ola/>
</connectionFactory>
<zosconnect_zosConnectService serviceRef="svc1"
serviceAsyncRequestTimeout="600s"
serviceName="dvs1" id="sdef1"/>
<zosconnect_localAdaptersConnectService
connectionWaitTimeout="7200"
connectionFactoryRef="wolaCF"
serviceName="DVSS1"
registerName="ZCEEDVM"
id="svc1"/>
</server>
```

DVS . AVZS . SAVZEXEC (AVZSIN00)

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

z/OS Resources accessible from DVM using SQL*



z/OS Connect EE

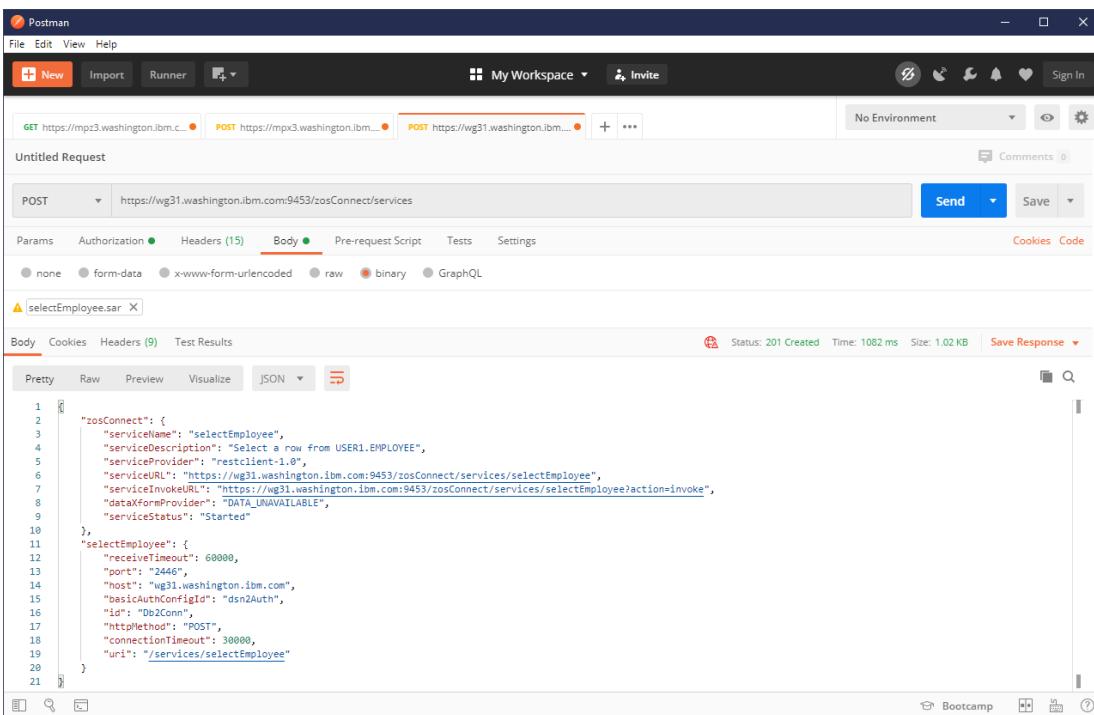
Data Source	SELECT	INSERT	UPDATE	DELETE	CALL Statement
CA IDMS	Yes	No	No	No	N/A
CICS COMMAREA	N/A	N/A	N/A	N/A	Yes
DB2 for z/OS	Yes	Yes	Yes	Yes	Yes
Db2 Direct	Yes	No	No	No	N/A
Db2 other platforms	Yes	Yes	Yes	Yes	Yes
IMS DBCTL	Yes	Yes	Yes	Yes	N/A
IMS Direct	Yes	No	No	No	N/A
IMS OTMA	N/A	N/A	N/A	N/A	Yes
MQ	Yes	No	No	No	N/A
Natural	N/A	N/A	N/A	N/A	Yes
SMF	Yes	No	No	No	N/A
Sequential data set	Yes	Yes	No	No	N/A
VSAM data set*	Yes	Yes	Yes	Yes	N/A
z/OS Syslog	Yes	No	No	No	N/A
OMVS files	Yes	No	No	No	Yes

*Administrating IBM DVM Manager, SC27-9303

© 2018, 2022 IBM Corporation

Deploying Service Archive files (those not developed in Eclipse)

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



The screenshot shows the Postman interface with a POST request to <https://wg31.washington.ibm.com:9453/zosConnect/services>. The 'Body' tab is selected, and the file 'selectEmployee.sar' is attached. The response status is 201 Created.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
    "zosConnect": {
      "serviceName": "selectEmployee",
      "serviceDescription": "Select a row from USER1.EMPLOYEE",
      "serviceProvider": "restclient-1.0",
      "serviceURL": "https://wg31.washington.ibm.com:9453/zosConnect/services/selectEmployee",
      "dataFormProvider": "DATA_UNAVAILABLE",
      "serviceStatus": "Started"
    },
    "selectEmployee": {
      "receiveTimeout": 60000,
      "port": "2446",
      "host": "wg31.washington.ibm.com",
      "basicAuthConfigId": "dsn2Auth",
      "id": "Db2Conn",
      "httpMethod": "POST",
      "connectionTimeout": 30000,
      "uri": "/services/selectEmployee"
    }
  }
}
  
```

Command:

```

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

Results:

```

{
  "zosConnect": {
    "serviceName": "selectEmployee",
    "serviceDescription": "Select a row from USER1.EMPLOYEE",
    "serviceProvider": "IBM_ZOS_CONNECT_SERVICE_REST_CLIENT-1.0",
    "serviceURL": "https://mpxm:9453/zosConnect/services/selectEmployee",
    "dataFormProvider": "DATA_UNAVAILABLE",
    "serviceStatus": "Started"
  },
  "selectEmployee": {
    "receiveTimeout": 0,
    "port": null,
    "host": null,
    "httpMethod": "POST",
    "connectionTimeout": 0,
    "uri": "/services/selectEmployee"
  }
}
  
```



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 they are created

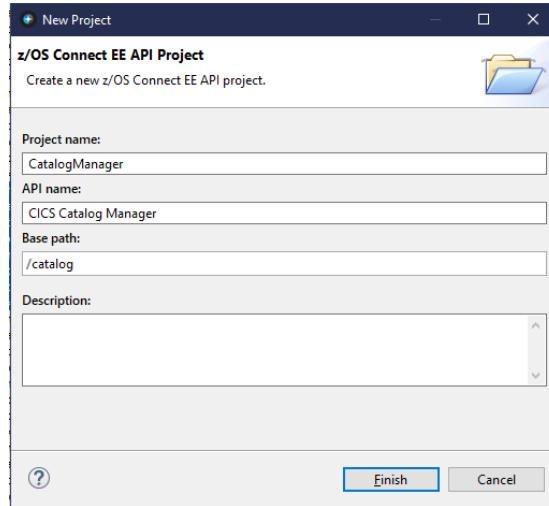


/api_toolkit/api_editor

Quick and easy **API mapping**.



API toolkit – API Editor



Describe your API

Name: CICS Catalog Manager Description:

Base path: /catalog

Version: 1.0.0

Contact Information

Path

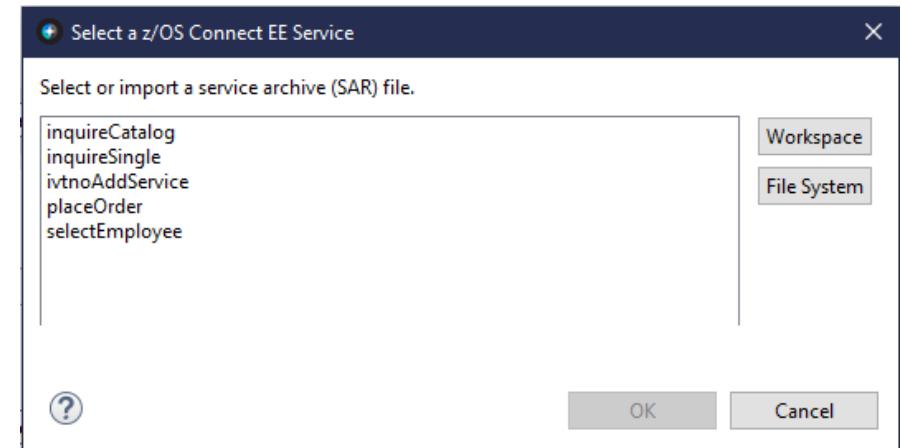
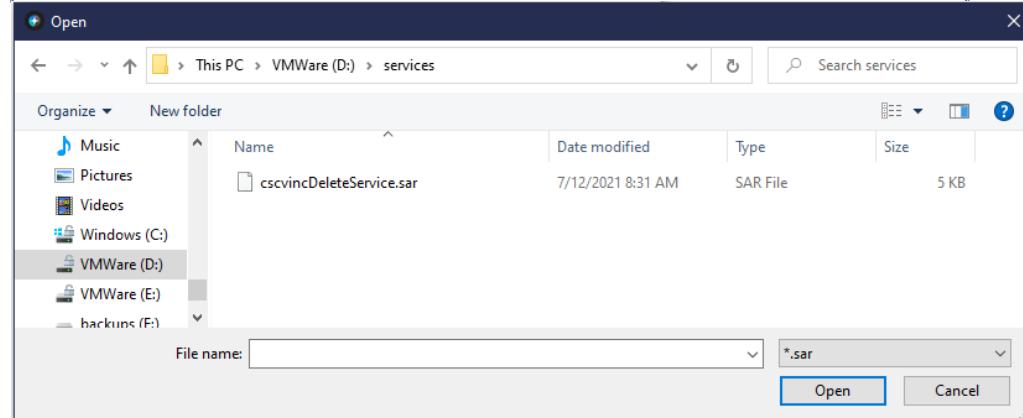
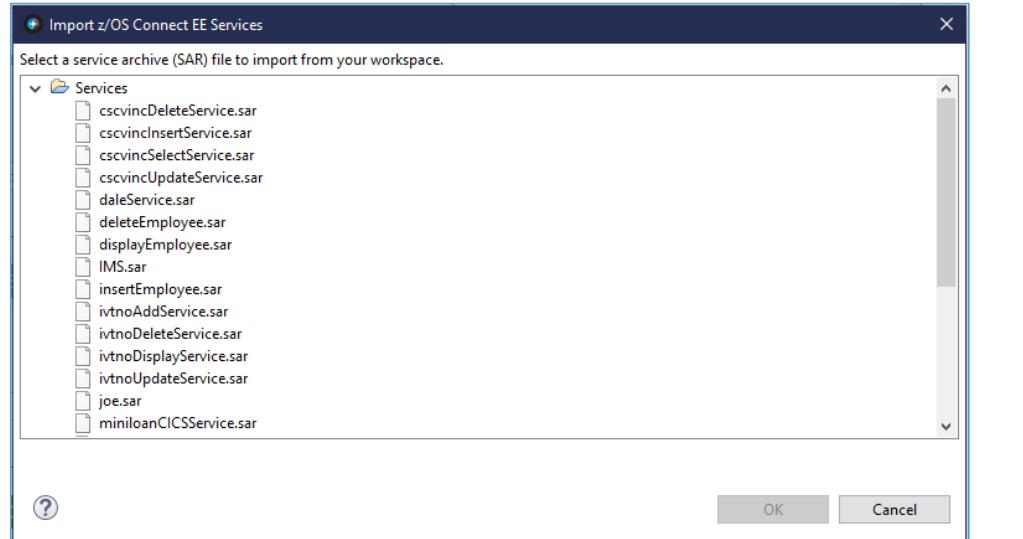
/newPath1

Methods (4)

Method	Service...	Mapping...	Up	Down	X
POST					
GET					
PUT					
DELETE					



Importing the service archives files





API toolkit – API Editor

The screenshot shows the API toolkit API Editor interface. It displays three API definitions:

- catalog**: Path: /catalog, Methods: GET inquireCatalog, PUT ivtnoAddService.
- order**: Path: /order, Methods: POST placeOrder, PUT selectEmployee.
- item**: Path: /item/{itemID}, Methods: GET inquireSingle.

A large red oval highlights the **item** API definition.

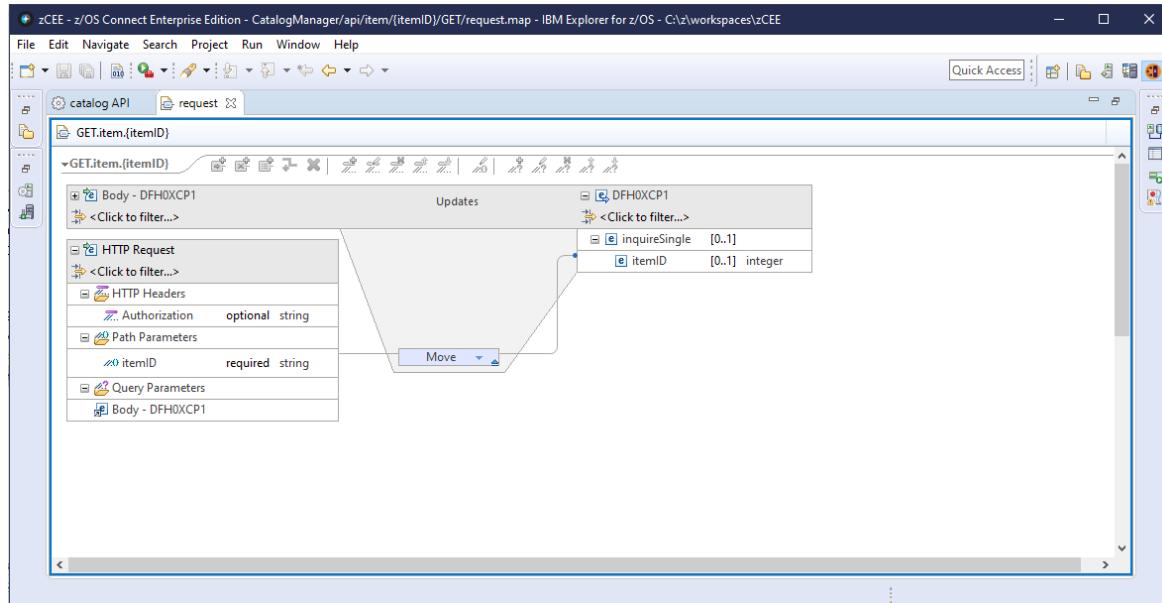
The **API toolkit** is designed to encourage RESTful API design.

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.

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.

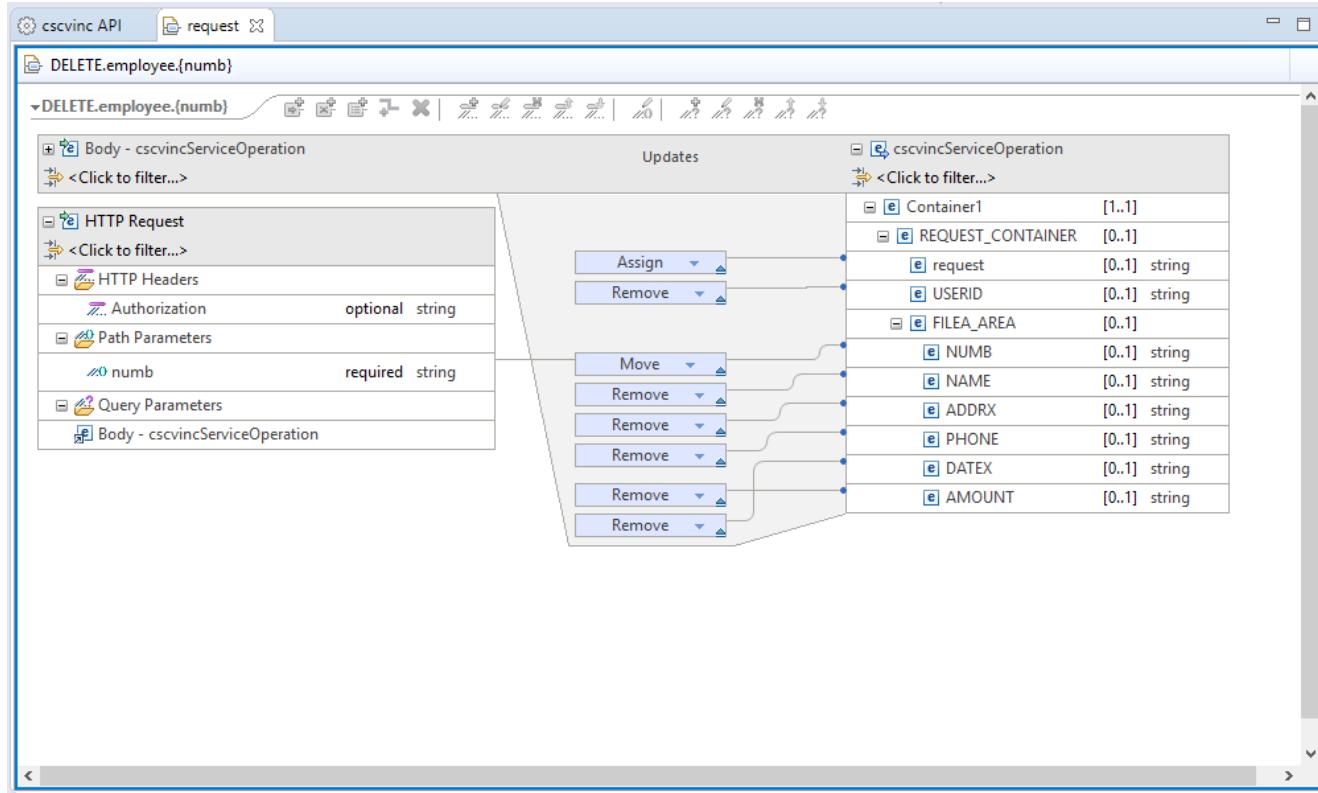
Map path and query parameters to native data structures.

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).

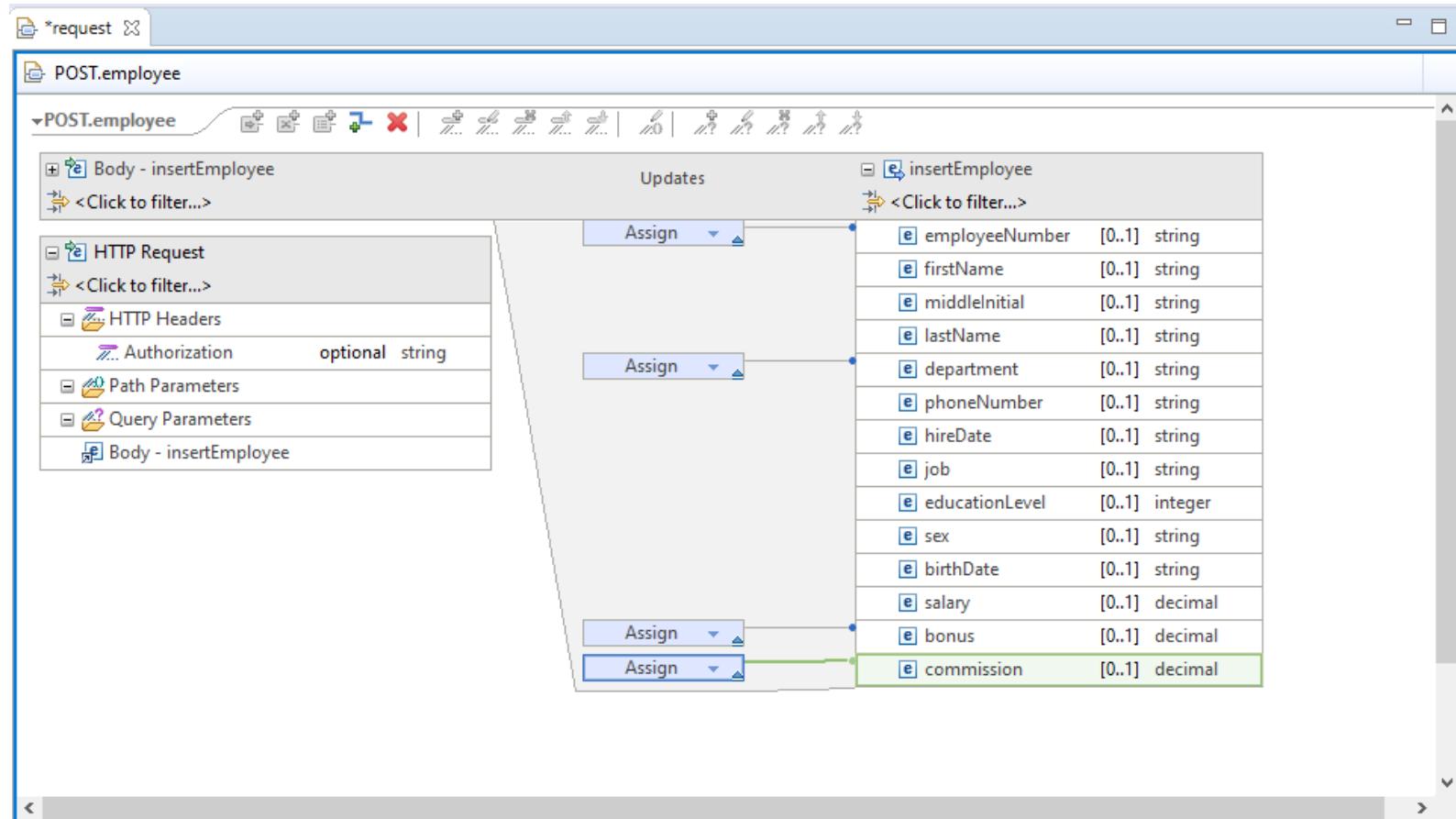
API toolkit – API Editor

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



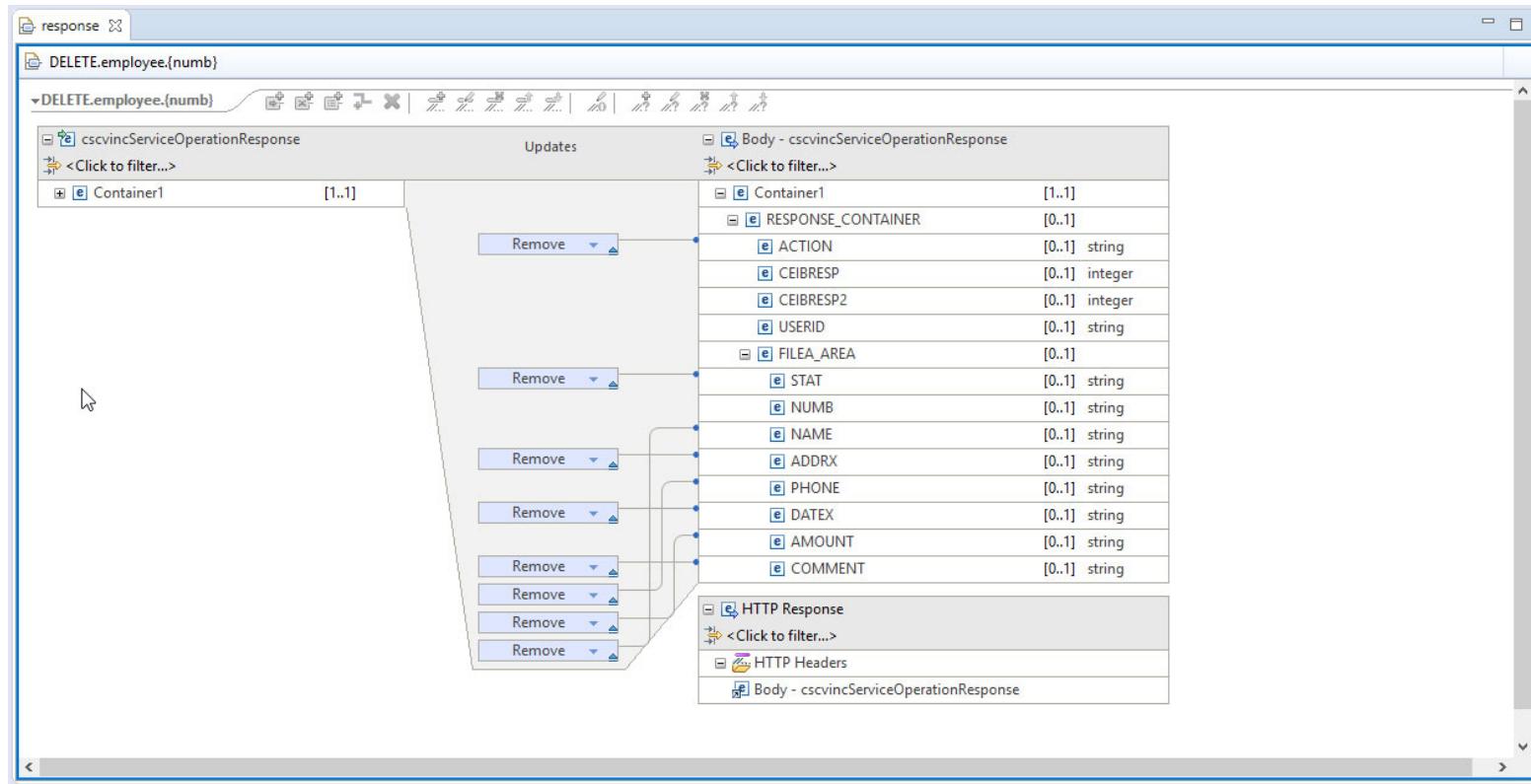
API toolkit – API Editor and Db2 REST service

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



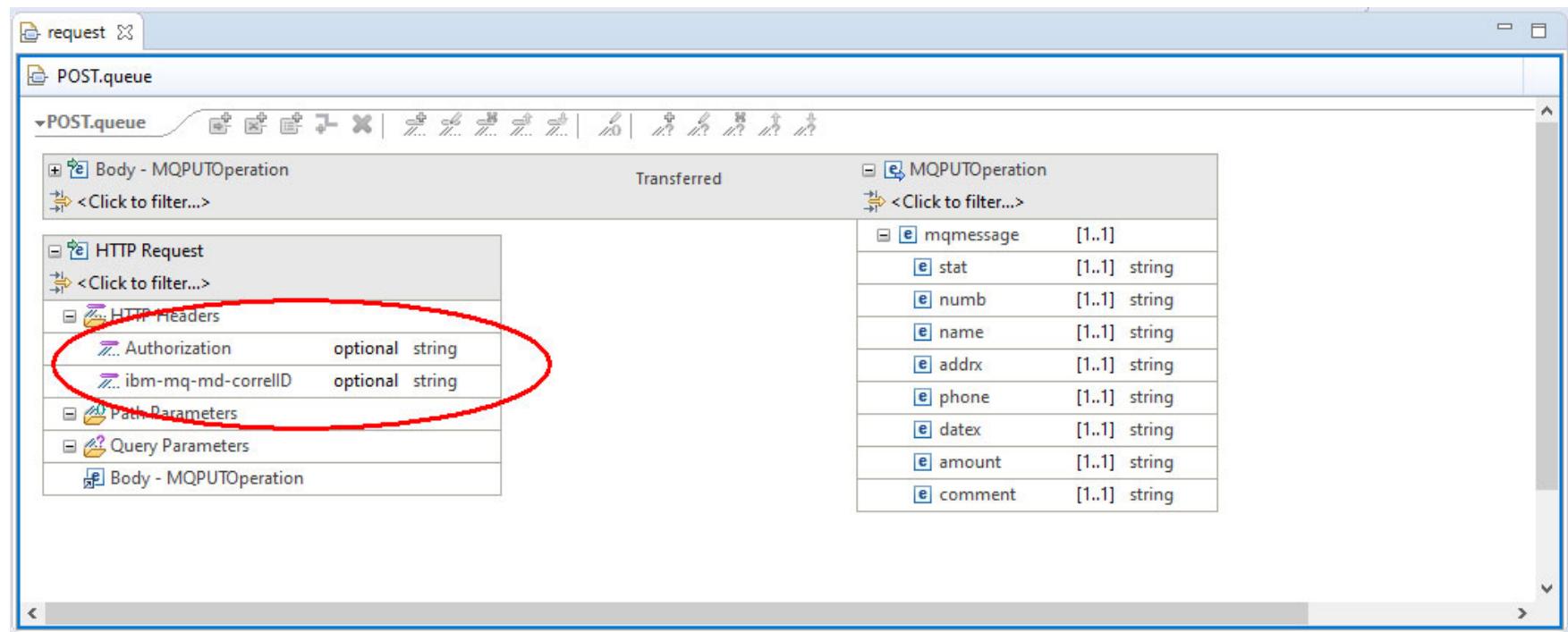
API toolkit – API Editor

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



API toolkit – API Editor

API mapping: Allows adding HTTP header properties



The screenshot shows the API Editor interface for defining an API mapping. The left pane displays the request structure for a POST operation on a queue, while the right pane shows the corresponding response structure. A red oval highlights the 'HTTP Headers' section in the request mapping, specifically the 'Authorization' and 'ibm-mq-md-correlID' fields.

Request (Left):

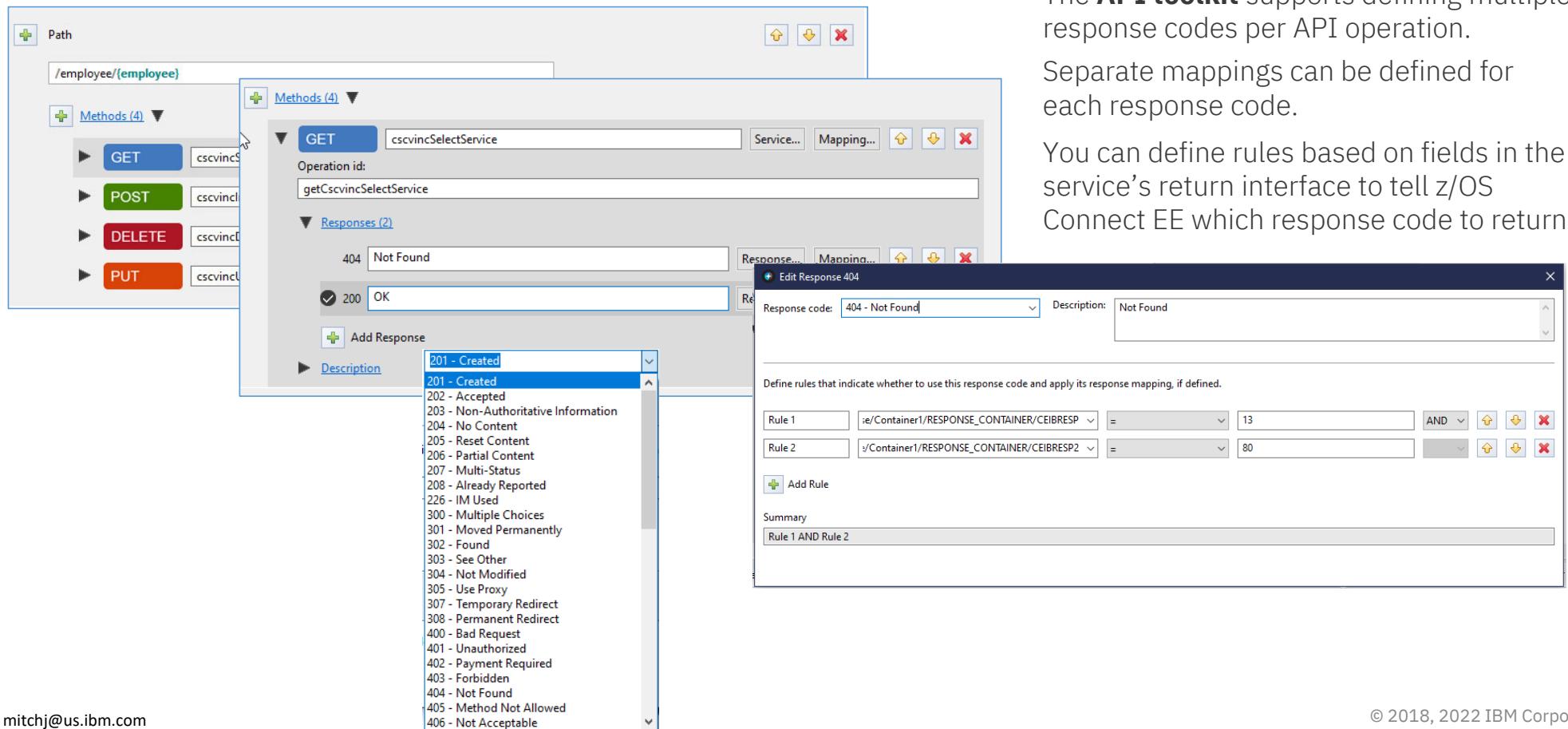
- POST.queue
- Body - MQPUTOperation
- HTTP Request
 - HTTP Headers
 - Authorization
 - ibm-mq-md-correlID
 - Path Parameters
 - Query Parameters
 - Body - MQPUTOperation

Response (Right):

- MQPUTOperation
- mqmessage [1..1]
 - stat [1..1] string
 - numb [1..1] string
 - name [1..1] string
 - addrx [1..1] string
 - phone [1..1] string
 - datex [1..1] string
 - amount [1..1] string
 - comment [1..1] string

API toolkit

API mapping: API definition with multiple response codes



The screenshot shows the API toolkit interface for defining API mappings. On the left, a tree view shows a path: /employee/{employee}. Underneath it, four methods are listed: GET, POST, DELETE, and PUT, each associated with a service name like cscvinc...

In the center, a detailed view of the GET method is shown. It has an operation ID: getCscvincSelectService and a response section titled 'Responses (2)'. It lists two responses: '404 Not Found' and '200 OK'. A dropdown menu is open over the '200 OK' entry, showing a list of other response codes: 201 - Created, 202 - Accepted, 203 - Non-Authoritative Information, 204 - No Content, 205 - Reset Content, 206 - Partial Content, 207 - Multi-Status, 208 - Already Reported, 226 - IM Used, 300 - Multiple Choices, 301 - Moved Permanently, 302 - Found, 303 - See Other, 304 - Not Modified, 305 - Use Proxy, 307 - Temporary Redirect, 308 - Permanent Redirect, 400 - Bad Request, 401 - Unauthorized, 402 - Payment Required, 403 - Forbidden, 404 - Not Found, 405 - Method Not Allowed, and 406 - Not Acceptable.

A modal window titled 'Edit Response 404' is displayed, showing the current response code as '404 - Not Found' and a description of 'Not Found'. It contains a section for defining rules:

Define rules that indicate whether to use this response code and apply its response mapping, if defined.

Rule 1: /e/Container1/RESPONSE_CONTAINER/CEIBRESP = 13 AND Rule 2: /Container1/RESPONSE_CONTAINER/CEIBRESP2 = 80

Summary: Rule 1 AND Rule 2

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

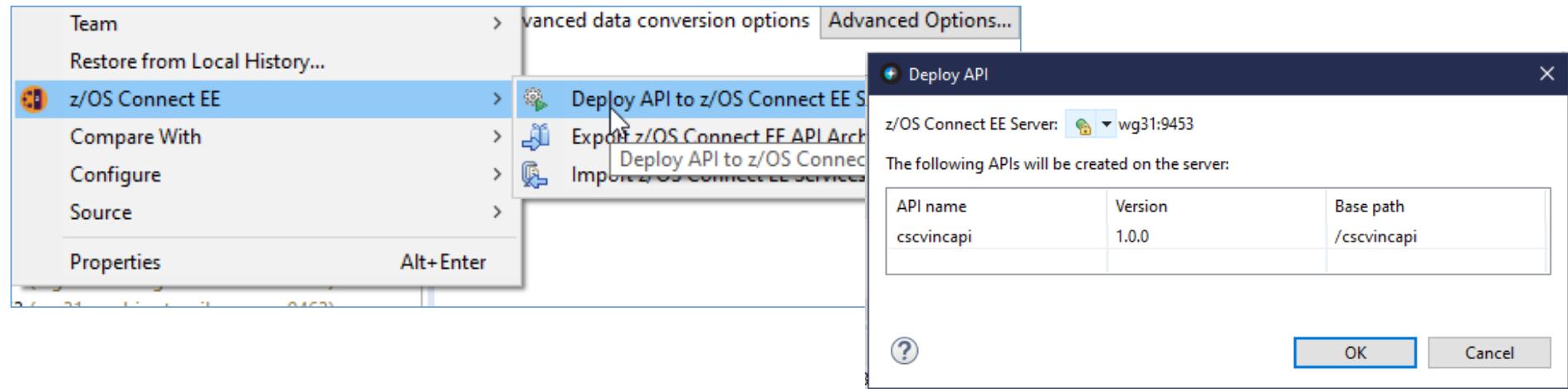
Separate mappings can be defined for each response code.

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

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.



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.

The screenshot shows the z/OS Connect EE API Editor interface. On the left, the 'API Catalog' window displays a tree view of deployed services and APIs. A context menu is open over the 'cscvinc' service, showing options like 'Open In Swagger UI'. Two separate browser windows are shown in the center-right, both titled 'Swagger UI'. The left window shows the Swagger UI for the 'cscvinc' service, listing four endpoints: POST /employee, DELETE /employee/{employee}, GET /employee/{employee}, and PUT /employee/{employee}. The right window shows a detailed view of the GET /employee/{employee} endpoint, including the Response Class (Status 200), OK status, Model (Employee), and Example Value (JSON code). The JSON code for the example value is:

```
{  
  "cscvincSelectServiceOperationResponse": {  
    "cscvincContainer": {  
      "response": {  
        "CEIBRESP": 0,  
        "CEIBRESP2": 0,  
        "USERID": "string",  
        "file": {  
          "employeeNumber": "string",  
          "name": "string",  
          "type": "string"  
        }  
      }  
    }  
  }  
}
```

The Response Content Type is set to application/json. Parameters for Authorization (header, string) and employee (path, string) are listed. Response Messages include 404 Not Found with a Model Example Value.



API Testing with Postman

A screenshot of the Postman application interface. The top navigation bar shows "File", "Edit", "View", "Help", "Home", "Workspaces", "Reports", and "Explore". A search bar says "Search Postman". On the right, there are icons for cloud, file, settings, and a gear, with a "Upgrade" button. The main workspace shows a list of requests. One request is selected: "GET https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111". Below it, the "Params" tab is selected, showing a table with columns "KEY", "VALUE", "DESCRIPTION", and "Bulk Edit". The table has one row with a single entry. At the bottom, tabs for "Body", "Cookies (1)", "Headers (8)", and "Test Results" are visible. The "Test Results" tab is selected, displaying a JSON response. The response body is:

```
1  "cscvincSelectServiceOperationResponse": {  
2      "cscvincContainer": {  
3          "response": {  
4              "CEIBRESP": 0,  
5              "CEIBRESP2": 0,  
6              "USERID": "CICSUSER",  
7              "filea": {  
8                  "employeeNumber": "111111",  
9                  "name": "C. BAKER",  
10                 "address": "OTTAWA, ONTARIO",  
11                 "phoneNumber": "511212003",  
12                 "date": "26 11 81",  
13                 "amount": "00011 00"  
14             }  
15         }  
16     }  
17 }
```

The status bar at the bottom shows "Status: 200 OK", "Time: 205 ms", and "Size: 899 B".

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

Save

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
..

Body Cookies (1) Headers (8) Test Results

Status: 200 OK Time: 205 ms Size: 899 B Save Response

Pretty Raw Preview Visualize JSON

1 "cscvincSelectServiceOperationResponse": {
2 "cscvincContainer": {
3 "response": {
4 "CEIBRESP": 0,
5 "CEIBRESP2": 0,
6 "USERID": "CICSUSER",
7 "filea": {
8 "employeeNumber": "111111",
9 "name": "C. BAKER",
10 "address": "OTTAWA, ONTARIO",
11 "phoneNumber": "511212003",
12 "date": "26 11 81",
13 "amount": "00011 00"
14 }
15 }
16 }
17 }

Find and Replace Console

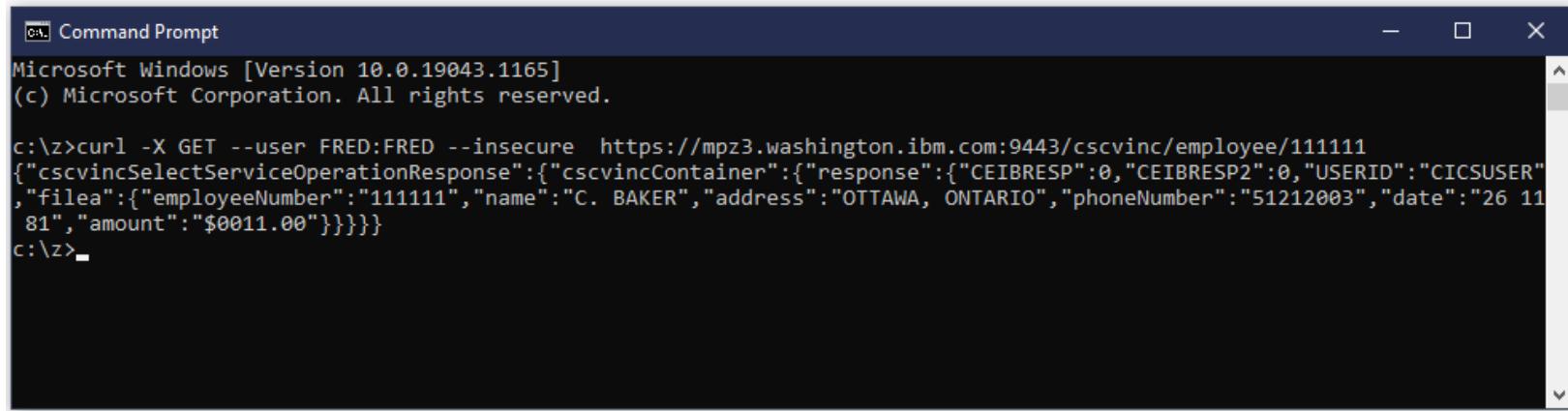
Bootcamp Runner Trash

mitchj@us.ibm.com

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

© 2018, 2022 IBM Corporation

API Testing with cURL



Command Prompt

```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

c:\z>curl -X GET --user FRED --insecure https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111
{"cscvincSelectServiceOperationResponse":{"cscvincContainer":{"response":{"CEIBRESP":0,"CEIBRESP2":0,"USERID":"CICSUSER","filea":{"employeeNumber":"111111","name":"C. BAKER","address":"OTTAWA, ONTARIO","phoneNumber":"51212003","date":"26 11 81","amount":"$0011.00"}}}}
c:\z>
```

<https://curl.se/download.html>

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



IBM

all Filter

Liberty REST APIs

Discover REST APIs available within Liberty

cscvinc

Show/Hide | List Operations | Expand

POST	/cscvinc/employee
DELETE	/cscvinc/employee/{employee}
GET	/cscvinc/employee/{employee}
PUT	/cscvinc/employee/{employee}

db2employee

Show/Hide | List Operations | Expand

filemgr

Show/Hide | List Operations | Expand

imsPhoneBook

Show/Hide | List Operations | Expand

jwtlvpDemoApi

Show/Hide | List Operations | Expand

miniloancics

Show/Hide | List Operations | Expand

mqapi

Show/Hide | List Operations | Expand

phonebook

Show/Hide | List Operations | Expand

File Edit View History Bookmarks Tools Help REST API Documentation + https://mpz3.washington.ibm.com:9443/api/explorer/#/cscvinc/employee/111111

Curl Try it out! Hide Response

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic RnJlZDpmcmVk' 'https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111'
```

Request URL

https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111

Response Body

```
{
  "cscvincSelectServiceOperationResponse": {
    "cscvincContainer": {
      "response": {
        "CEIBRESP": 0,
        "CEIBRESP2": 0,
        "USERID": "CICCSUSER",
        "file": {
          "employeeNumber": "111111",
          "name": "C. BAKER",
          "address": "OTTAWA, ONTARIO",
          "phoneNumber": "51121003",
          "date": "26 11 81",
          "amount": "$0011.00"
        }
      }
    }
  }
}
```

Response Code

200

Response Headers

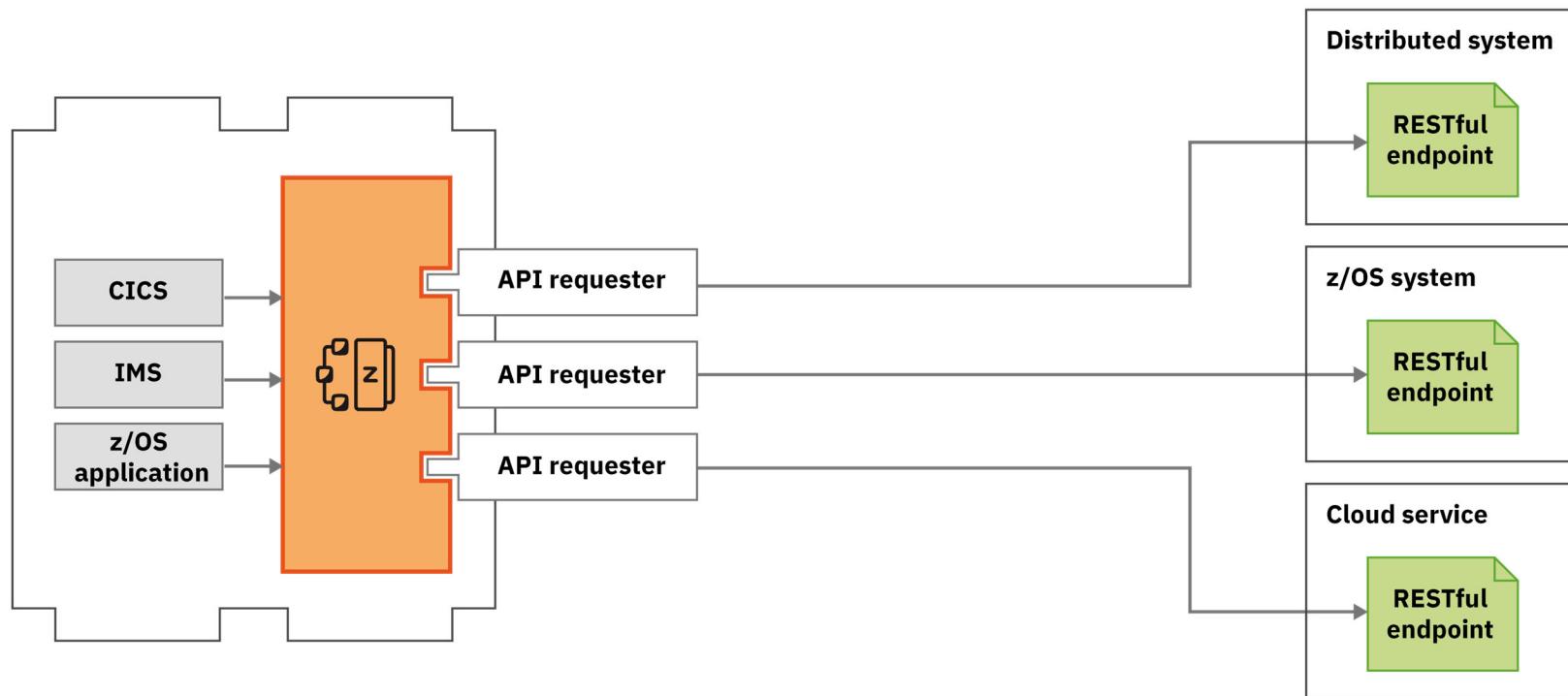
```
{
  "content-language": "en-US",
  "content-length": "269",
  "content-type": "application/json; charset=UTF-8"
}
```



/api_toolkit/apiRequesters

Quick and easy **API mapping**.

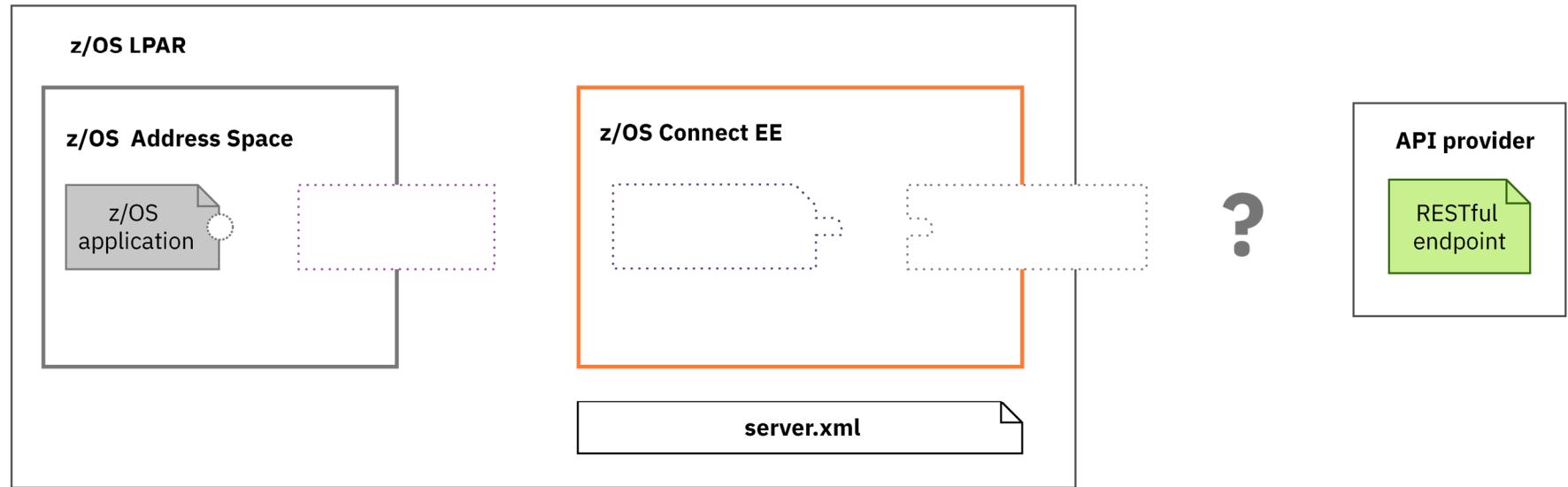
Use API requester to call external APIs from z/OS assets





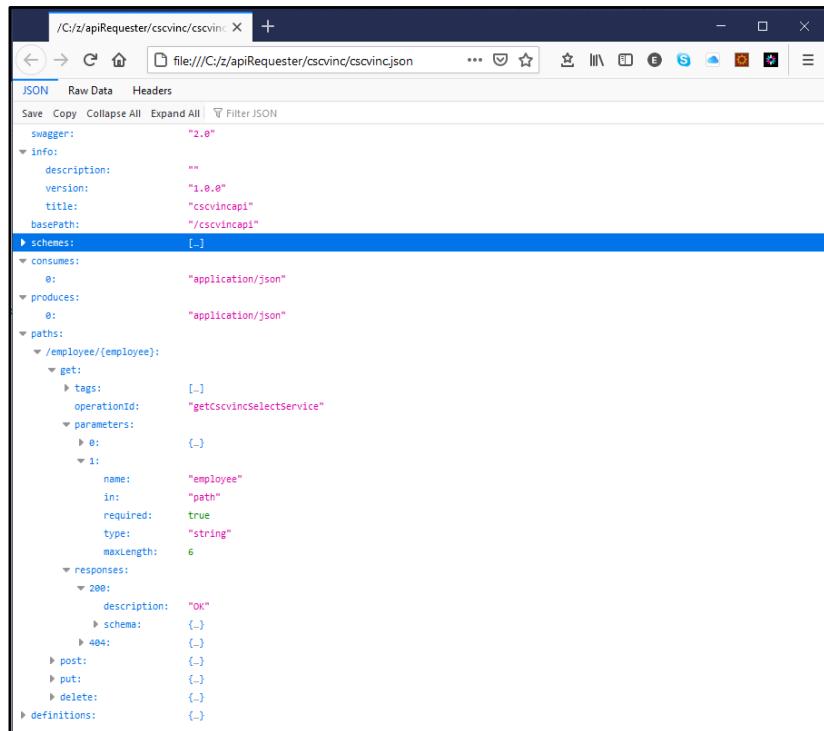
Steps to calling an external API

Starting point



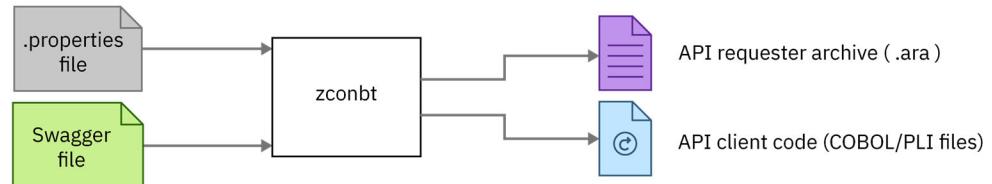
Steps to calling an external API

Generate API requester archive and API client code from Swagger



```

{
  "swagger": "2.0",
  "info": {
    "description": "",
    "version": "1.0.0",
    "title": "cscvincapi",
    "basePath": "/cscvincapi"
  },
  "schemes": [],
  "consumes": [
    {
      "o": "application/json"
    }
  ],
  "produces": [
    {
      "o": "application/json"
    }
  ],
  "paths": {
    "/employee/{employee}": {
      "get": {
        "tags": [
          "getCscvincSelectService"
        ],
        "parameters": [
          {
            "o": {}
          },
          {
            "name": "employee",
            "in": "path",
            "required": true,
            "type": "string",
            "maxLength": 6
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {}
          },
          "404": {}
        }
      }
    }
  }
}
  
```



```

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

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

Steps to calling an external API

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

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

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

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

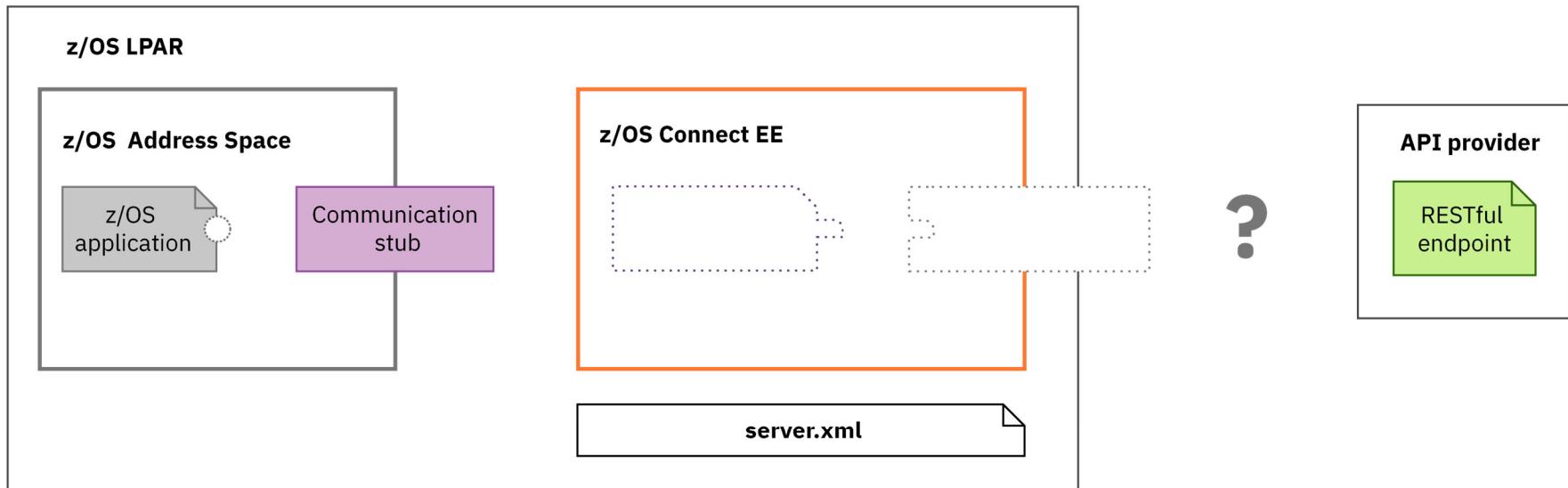
operationId: deleteCscvincDeleteService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: DELETE
- request data structure : CSC03Q01
- response data structure : CSC03P01
- api info file : CSC03I01

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



Steps to calling an external API

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



Configure a communication stub.

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

i ibm.biz/zosconnect-configure-comms-stub

Steps to calling an external API

Include the generated copy books in a COBOL program

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

  * Copy API Requester required copybook
  COPY BAQRINFO.

  * Request and Response
  01 API-REQUEST.
    COPY CSC02Q01.
  01 API_RESPONSE.
    COPY CSC02P01.

  * Structure with the API information
  01 API-INFO-OPER1.
    COPY CSC02I01.

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

API-REQUEST

```
CSC00I01  CSC00Q01 X
  * JSON schema keyword 'minLength' value: '0'.
  * JSON schema keyword 'maxLength' value: '6'.
  * This field contains a varying length array of characters or
  * binary data.
  *      09 employee-length          PIC S9999 COMP-5 SYNC.
  *      09 employee                PIC X(6).
  *
  * ++++++
  06 ReqPathParameters.
    09 employee-length          PIC S9999 COMP-5 SYNC.
    09 employee                PIC X(6).
```

API-RESPONSE

```
CSC00I01  CSC00Q01  CSC00P01 X
  * ++++++
  06 RespBody.
    09 cscvincap0-num          PIC S9(9) COMP-5 SYNC.
    09 cscvincap0-operatio.
      12 Container1.
    15 RESPONSE-CONTAINER2-num PIC S9(9) COMP-5
      SYNC.
```

API-INFO-OPER1

```
CSC00I01 X
  03 BAQ-APINAME           PIC X(255)
    VALUE 'cscvincapi_1.0.0'.
  03 BAQ-APINAME-LEN        PIC S9(9) COMP-5 SYNC
    VALUE 16.
  03 BAQ-APIPATH            PIC X(255)
    VALUE '%2Fcvincap%2Femployee%2F%7Bemployee%7D'.
  03 BAQ-APIPATH-LEN         PIC S9(9) COMP-5 SYNC
    VALUE 41.
  03 BAQ-APIMETHOD          PIC X(255)
    VALUE 'GET'.
  03 BAQ-APIMETHOD-LEN       PIC S9(9) COMP-5 SYNC
    VALUE 3.
```

Steps to calling an external API

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

The diagram illustrates the steps to calling an external API. It shows the GETAPI PGM code, the communication stub definitions (CSC00101), and the copy book definitions (CSC00Q01) used in the stubs.

GETAPI PGM Code:

```
* 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
* API request was successful.
```

CSC00101 Communication Stub Definitions:

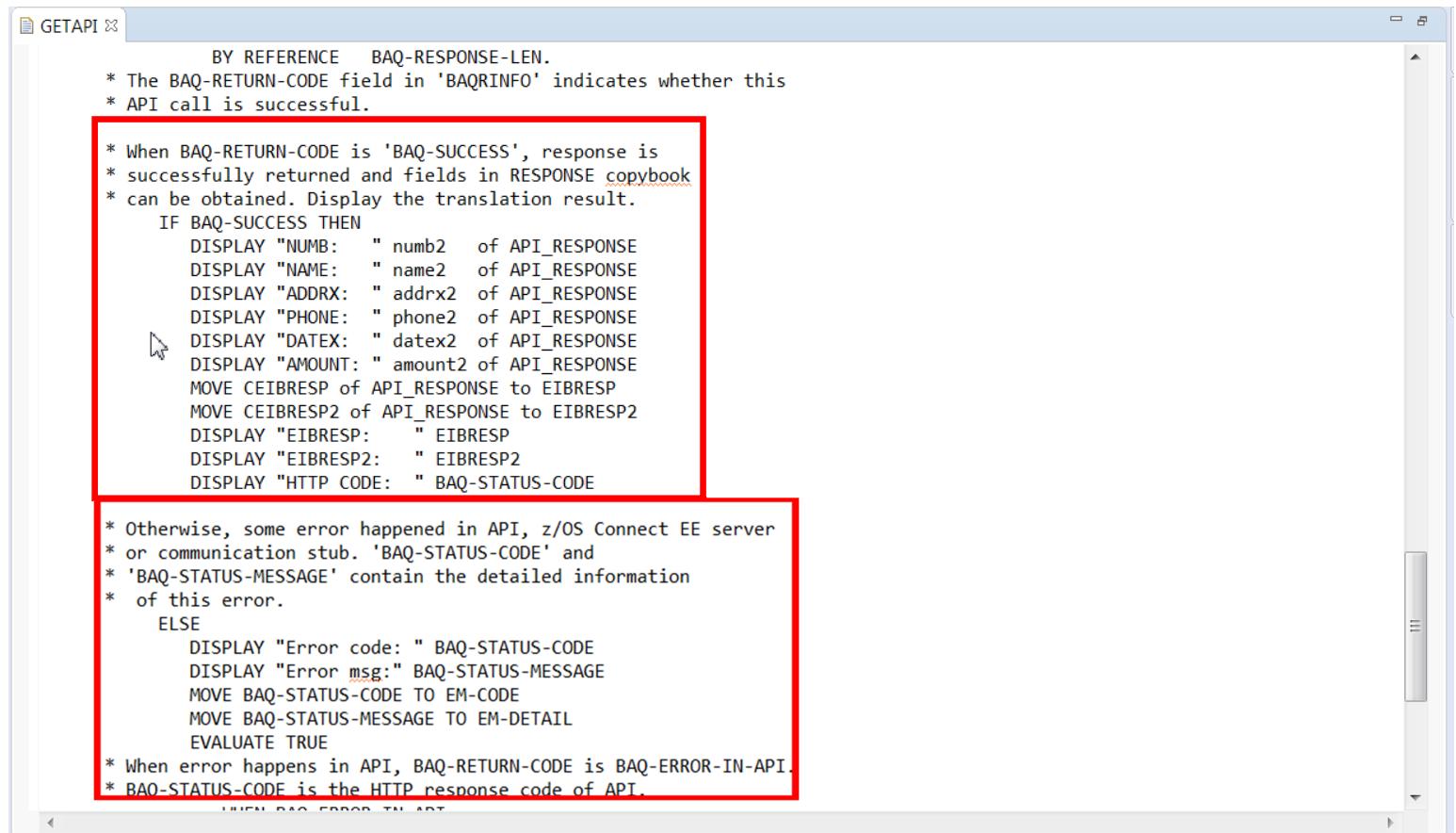
Field	Type	Description
03 BAQ-APINAME	PIC X(255)	Value: 'cscvincap1_1.0.0'.
03 BAQ-APINAME-LEN	PIC S9(9) COMP-5 SYNC	Value: 16.
03 BAQ-APIPATH	PIC X(255)	Value: '\$2\$cscvincap1%2F%7Bemployee%7D'.
03 BAQ-APIPATH-LEN	PIC S9(9) COMP-5 SYNC	Value: 41.
03 BAQ-APIMETHOD	PIC X(255)	Value: 'GET'.
03 BAQ-APIMETHOD-LEN	PIC S9(9) COMP-5 SYNC	Value: 3.

CSC00Q01 Copy Book Definitions:

Field	Type	Description
09 employee-length	PIC S9999 COMP-5 SYNC.	JSON schema keyword 'minLength' value: '0'.
09 employee	PIC X(6).	JSON schema keyword 'maxLength' value: '6'.
09 employee-length	PIC S9999 COMP-5 SYNC.	This field contains a varying length array of characters or binary data.
09 employee	PIC X(6).	06 ReqPathParameters.
09 employee-length	PIC S9999 COMP-5 SYNC.	06 RespBody.
09 cscvincSelectServiceOperatio	PIC X(6).	09 cscvincSelectServiceOp-num
12 Container1.	PIC X(6).	12 Container1.
15 RESPONSE-CONTAINER2-num	PIC S9(9) COMP-5 SYNC.	15 RESPONSE-CONTAINER2-num

Steps to calling an external API

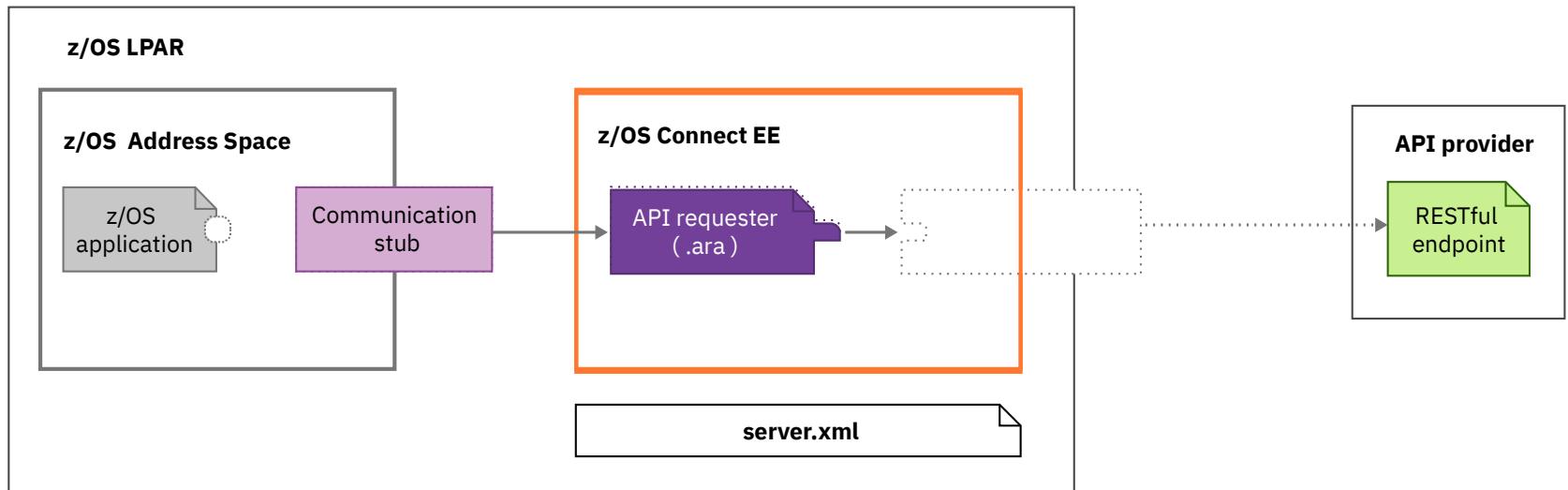
Access the results



```
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.  
    WHEN BAQ-ERROR-IN-API
```

Steps to calling an external API

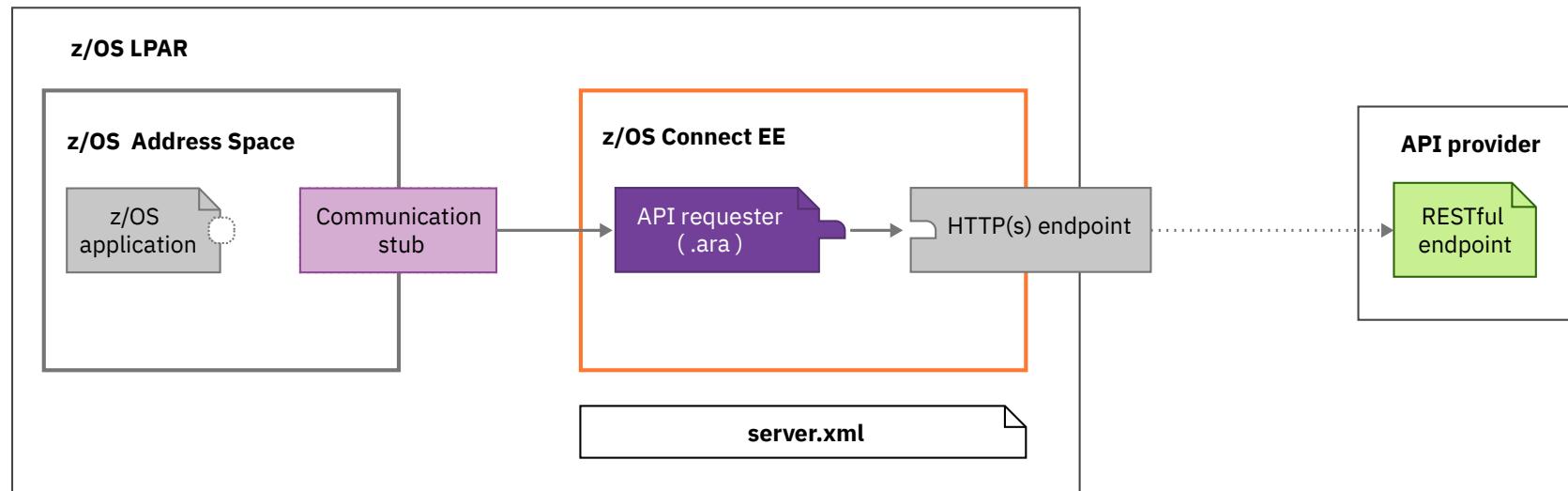
Deploy API requester (.ara) archive



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

Steps to calling an external API

Configure HTTP(S) endpoint configuration element

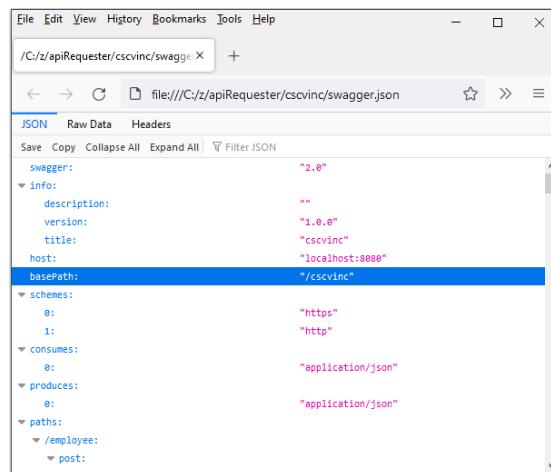


Configure the connection between z/OS Connect EE and the external API.

 ibm.biz/zosconnect-configure-endpoint-connection

Steps to calling an external API

Update the server XML configuration for the endpoint



A screenshot of a browser window displaying the Swagger JSON for the cscvinc API. The URL is /C:/apiRequester/cscvinc/swagger.json. The JSON structure shows the API's version, title, host, basePath, schemes, consumes, produces, and paths. The basePath is highlighted with a blue selection bar.

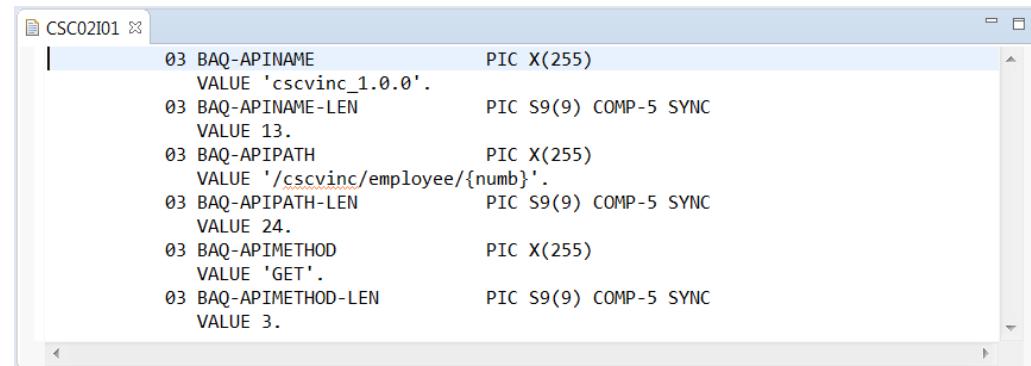
```
swagger: "2.0"
info:
  description: ""
  version: "1.0.0"
  title: "cscvinc"
  host: "localhost:8080"
  basePath: "/cscvinc"

schemes:
  0: "https"
  1: "http"

consumes:
  0: "application/json"

produces:
  0: "application/json"

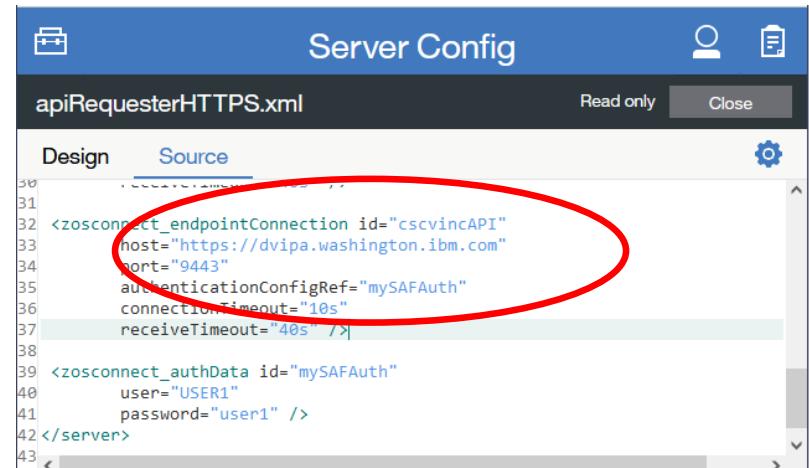
paths:
  /employee:
    post:
```



A screenshot of the CSC02I01 editor showing AS/400 source code. The code defines several parameters (BAQ-APINAME, BAQ-APINAME-LEN, BAQ-APIPATH, BAQ-APIPATH-LEN, BAQ-APIMETHOD, BAQ-APIMETHOD-LEN) with specific values and lengths, intended to generate an API request.

```
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.
```

cscvinc.properties
connectionRef=cscvincAPI



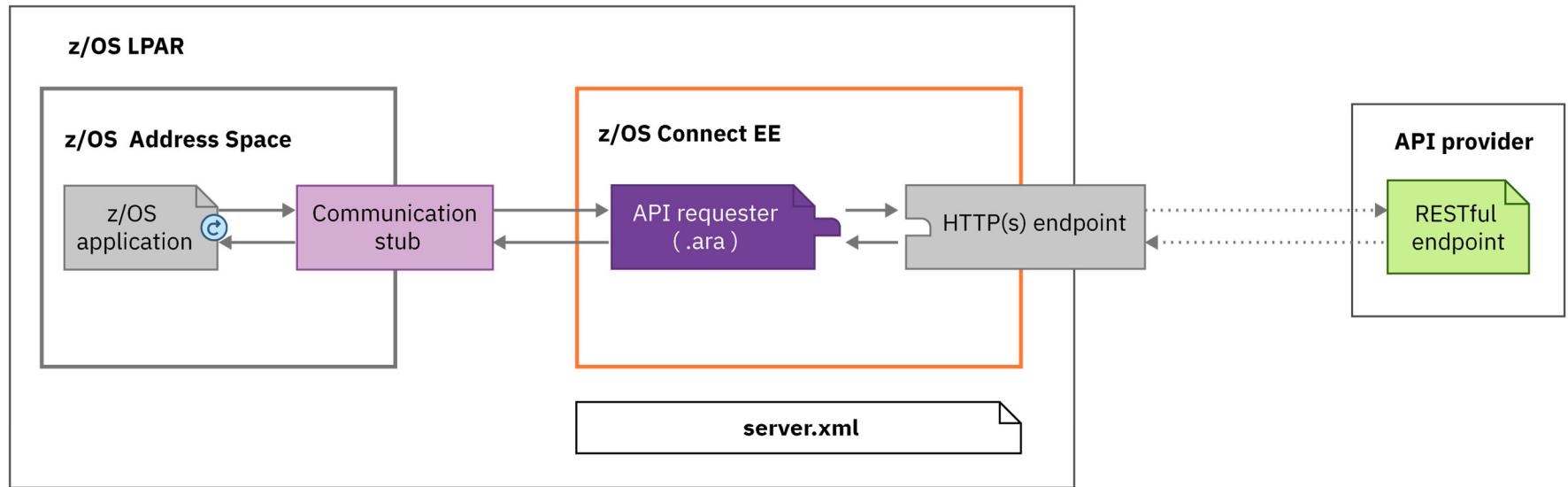
A screenshot of the Server Config interface showing the XML configuration file apiRequesterHTTPS.xml. The XML includes a <zosconnect_endpointConnection> element with attributes id="cscvincAPI", host="https://dvipa.washington.ibm.com", port="9443", authenticationConfigRef="mySAFAuth", connectionTimeout="10s", and receiveTimeout="40s". A red oval highlights this section of the XML code.

```
<zosconnect_endpointConnection id="cscvincAPI"
  host="https://dvipa.washington.ibm.com"
  port="9443"
  authenticationConfigRef="mySAFAAuth"
  connectionTimeout="10s"
  receiveTimeout="40s" />
```

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

Steps to calling an external API

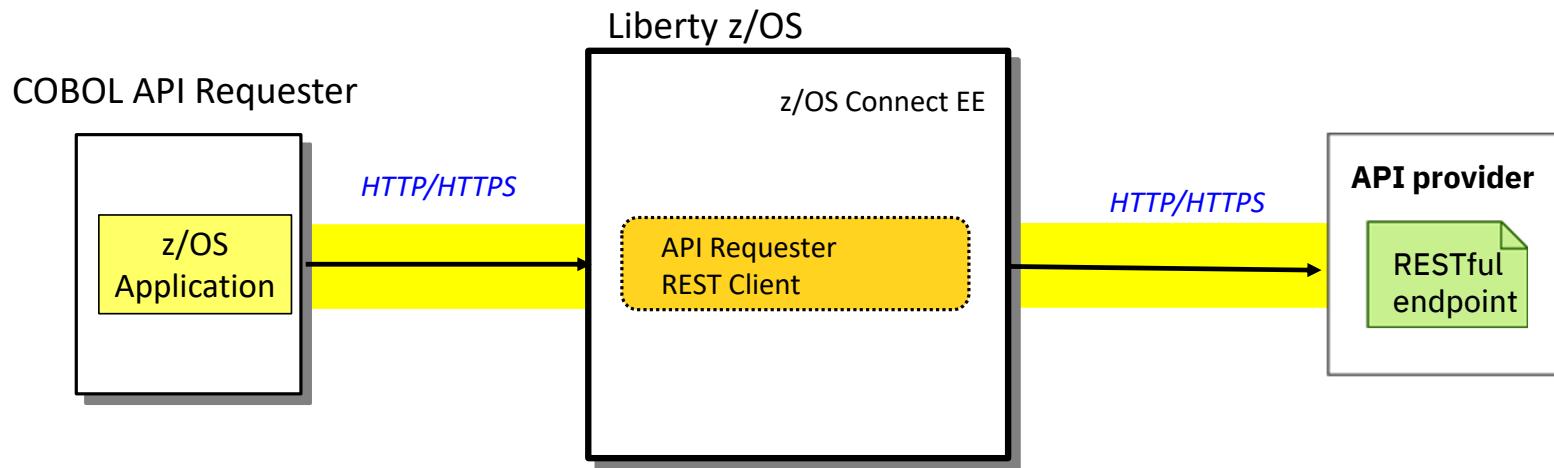
Done



API requester to API Provider connection overview



z/OS Connect EE



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:

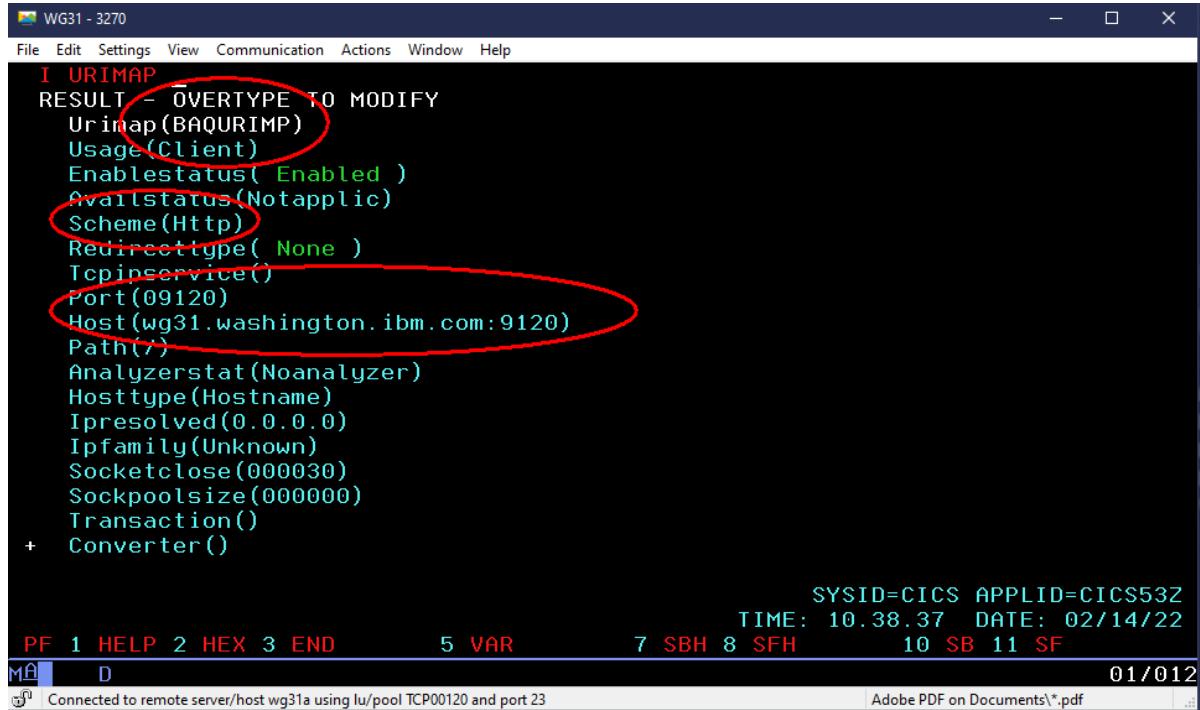
- CICS URIMAP resource (default BAQURIMP)
 - HOST
 - PORT
 - SCHEME (HTTP/HTTPS)

Configure connections to the z/OS API requester server



z/OS Connect EE

Default CICS URI MAP*



```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
I URIMAP
RESULT - OVERTYPE TO MODIFY
URIMAP(BAQURIMP)
Usage(Client)
Enablestatus(Enabled)
Availstatus(Notapplic)
Scheme(Http)
Redirecttype(None)
Tcpinservice()
Port(09120)
Host(wg31.washington.ibm.com:9120)
Path(/)
Analyzerstat(Noanalyzer)
Hosttype(Hostname)
Ipresolved(0.0.0.0)
Ipfamily(Unknown)
Socketclose(000030)
Sockpoolsize(000000)
Transaction()
+ Converter()

SYSID=CICS APPLID=CICS53Z
TIME: 10.38.37 DATE: 02/14/22
PF 1 HELP 2 HEX 3 END      5 VAR      7 SBH 8 SFH      10 SB 11 SF
01/012
M A D
Connected to remote server/host wg31a using lu/pool TCP00120 and port 23
Adobe PDF on Documents\*.pdf
```

LE Environment Variables

```
//DELTAPI EXEC PGM=DELTAPI,PARM='323232'
//STEPLIB DD
DISP=SHR,DSN=USER1.ZCEE.LOADLIB
//          DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=* //CEEOPTS DD *
  POSIX(ON),
ENVAR("BAQURI=wg31.washington.ibm.com",
"BAQPORT=9120")
```

* V3.0.37 added support for a CICS application to specify or request a specific URIMAP resource the using BAQ-ZCON-SERVER-URI variable in BAQRINFO

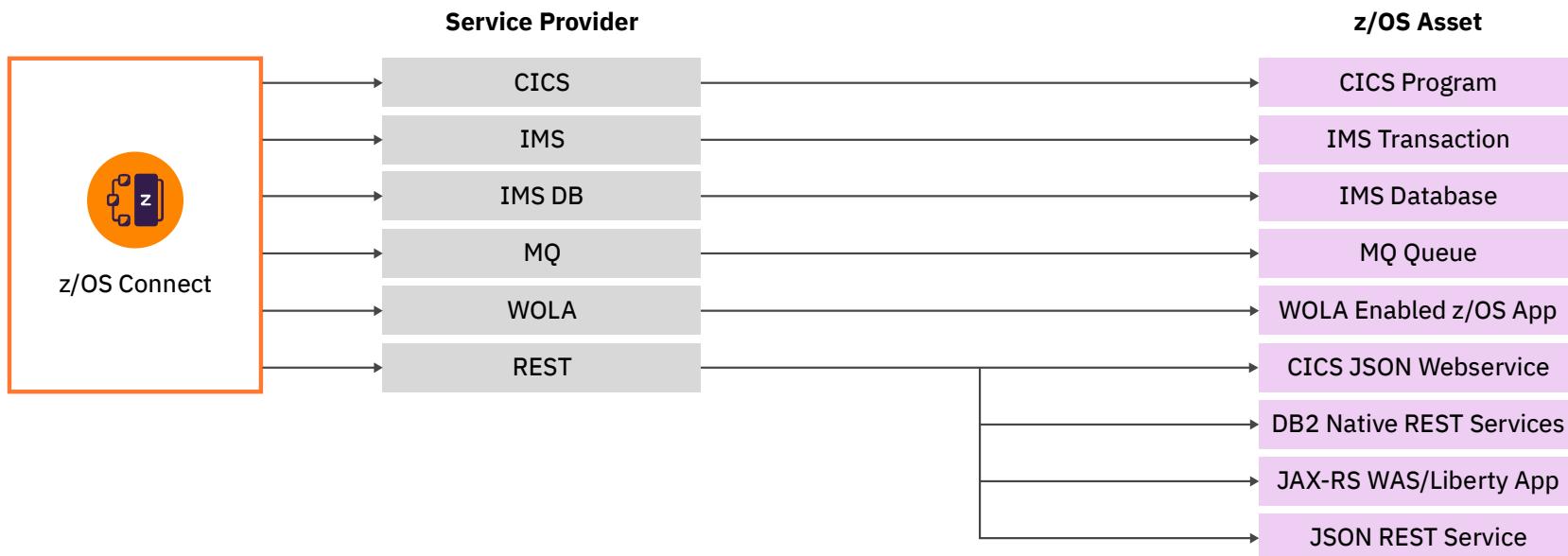


/miscellaneousTopics

performance, high availability, Liberty

What assets can z/OS Connect EE map to?

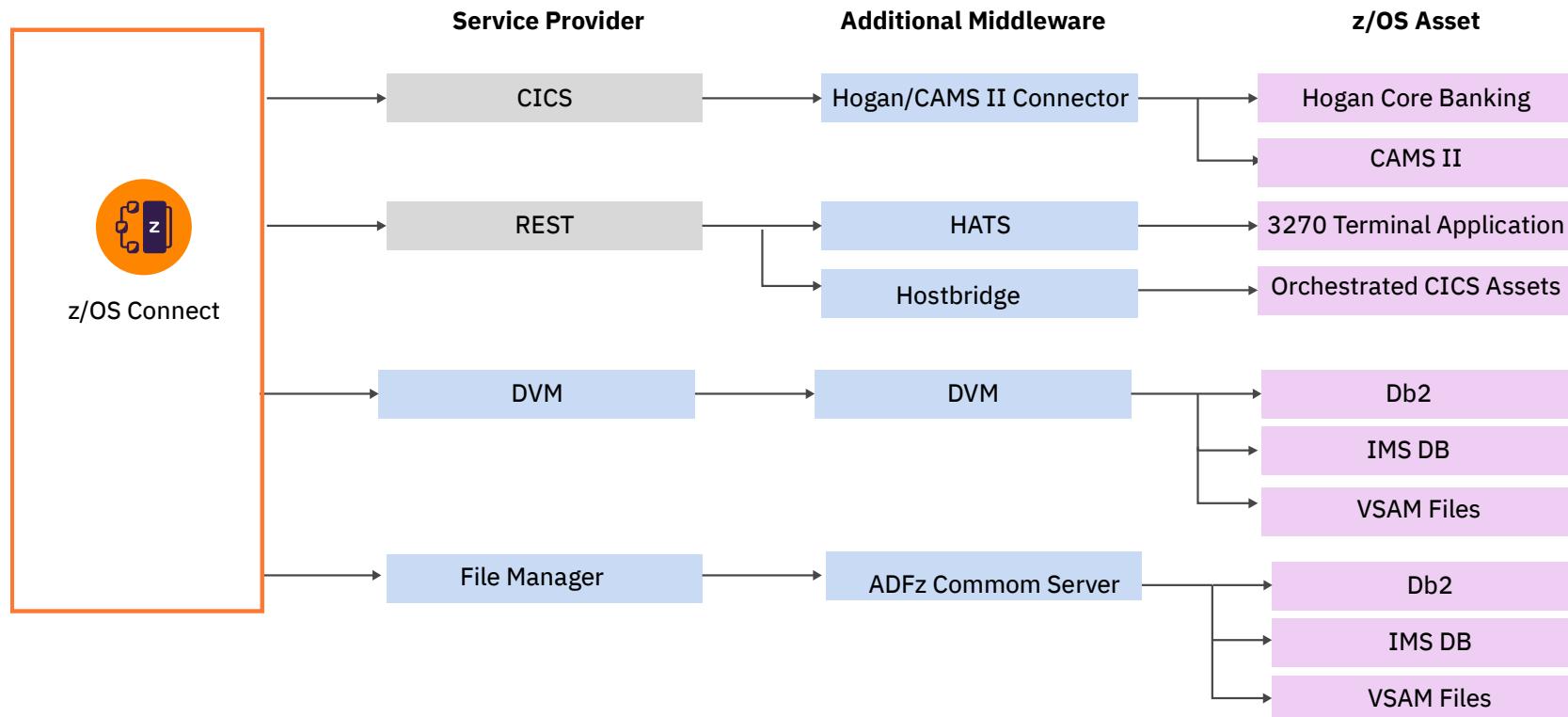
And which service provider could I use?



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

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

Tech/Tip: Providing access to service archives files



Index of /resources/zosConnect/

Name	Last Modified	Size	Description
apis	Fri Feb 19 13:46:13 GMT 2021	-	Directory
services	Sat Feb 20 20:54:41 GMT 2021	-	Directory
apiRequesters	Wed Feb 07 17:59:04 GMT 2018	-	Directory
rules	Tue Jan 26 20:34:05 GMT 2021	-	Directory

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

Index of /resources/zosConnect/services/

Name	Last Modified	Size	Description
csvincDeleteService.sar	Thu Feb 18 18:02:19 GMT 2021	4362	File
csvincInsertService.sar	Thu Feb 18 18:02:19 GMT 2021	4491	File
csvincSelectService.sar	Thu Feb 18 18:02:19 GMT 2021	4590	File

Opening csvincSelectService.sar

You have chosen to open:
csvincSelectService.sar
which is: SAR file (4.5 KB)
from: https://wg31.washington.ibm.com:9453

What should Firefox do with this file?
 Open with Applications\WINZIP32.EXE (default)
 Save File

OK Cancel

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

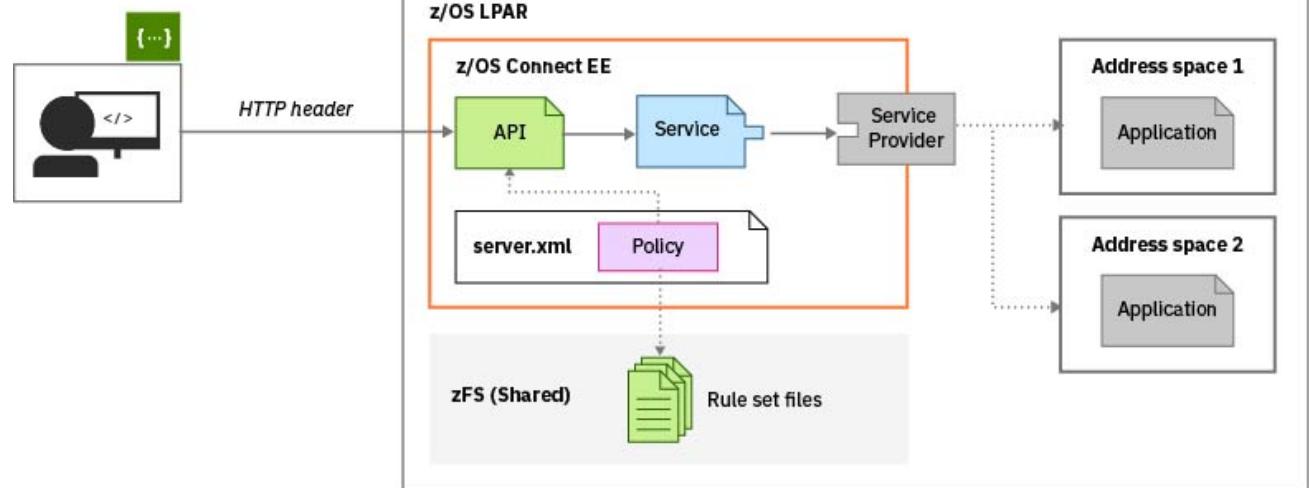
- imsConnectionRef
- imsInteractionRef
- imsInteractionTimeout
- imsLtermOverrideName
- imsTranCode
- imsTranExpiration

Db2 attributes

- db2ConnectionRef
- db2CollectionID

MQ attributes

- mqConnectionFactory
- mqDestination
- mqReplyDestination



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>
```

The screenshot shows the z/OS Connect API designer interface. It displays a configuration for a GET endpoint named "GET.employee.{numb}". The interface is organized into several sections:

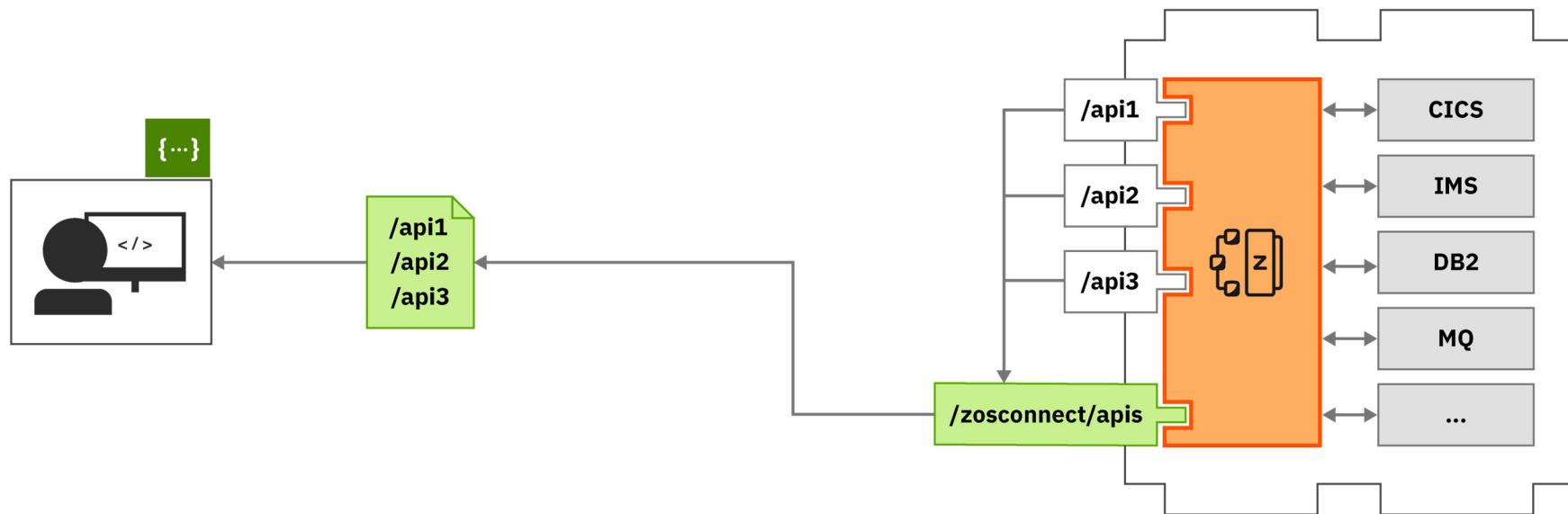
- Body - cscvincServiceOperation**: Contains a link "[Click to filter...](#)".
- HTTP Request**: Contains a link "[Click to filter...](#)". This section includes:
 - HTTP Headers**:
 - `cicsMirror` (optional string)
 - `cicsConnection` (optional string)
 - Path Parameters**:
 - `numb` (Required string)
 - Query Parameters**:
 - Body - cscvincServiceOperation**

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)

API Documentation



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



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.

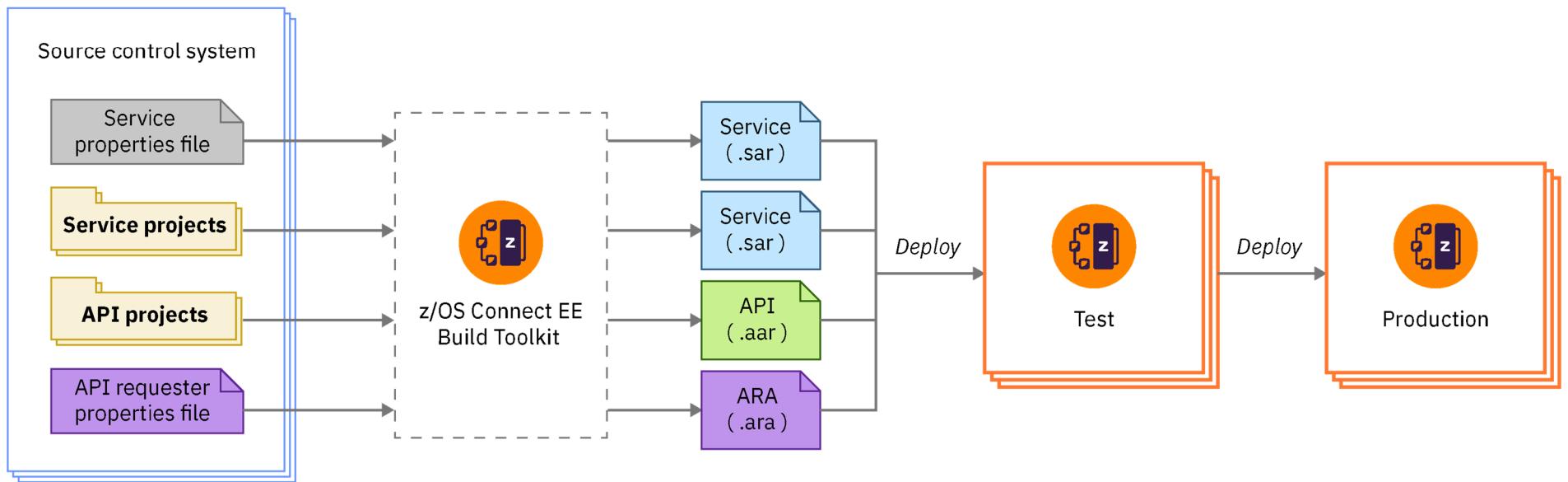
Show/Hide | List Operations | Expand Operations

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

DevOps using z/OS Connect EE

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

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

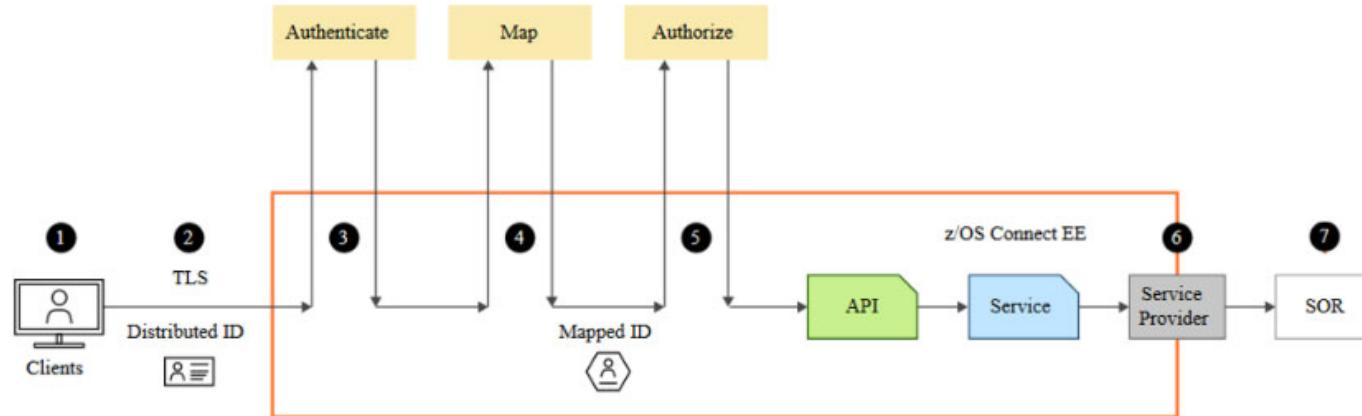




/security

How is security implement?

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

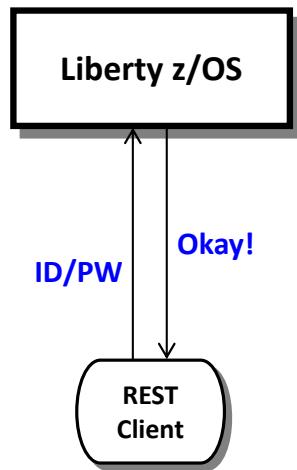
API Provider Authentication



z/OS Connect EE

Several different ways this can be accomplished:

Basic Authentication



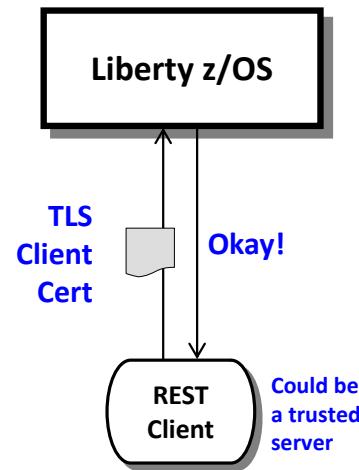
Server prompts for ID/PW

Client supplies ID/PW or
ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

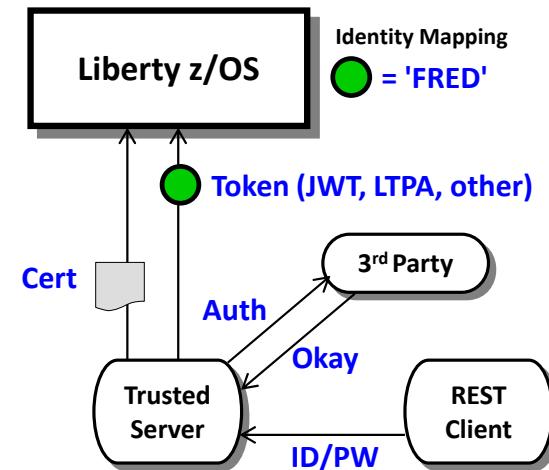
Client supplies certificate

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



Third Party Authentication Examples

The image displays two side-by-side screenshots of web pages illustrating third-party authentication.

Left Screenshot: UPS Sign Up

This screenshot shows the UPS "Sign Up" page. At the top, there's a banner stating "UPS is open for business: Service impacts related to Coronavirus ...More". Below the banner, the UPS logo is displayed. A "Sign Up / Log in" link and a "Search or Track" input field are visible. The main section is titled "Sign Up" and includes a link for users who already have an ID. It provides several social media integration options: Google, Facebook, Amazon, Apple, and Twitter. Below these, there are fields for entering personal information: Name*, Email*, User ID*, Password*, and Phone. The "Password" field includes a "Show" link. A "Feedback" button is located on the right side of the form.

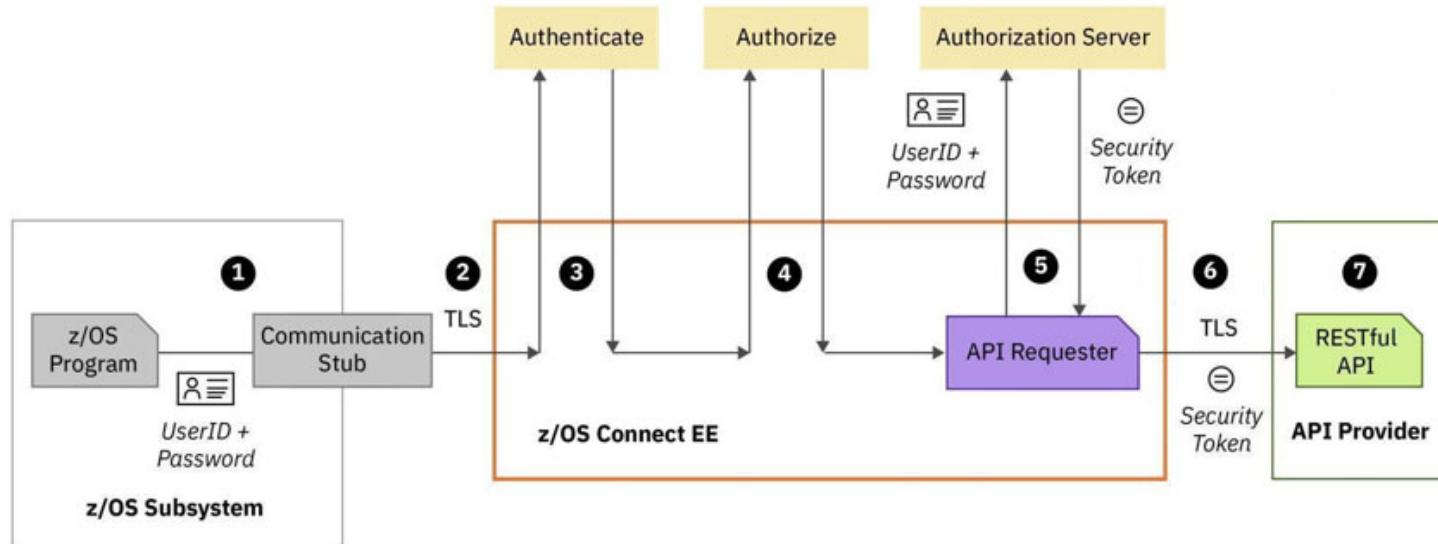
Right Screenshot: myNCDMV Log In

This screenshot shows the "Log In" page for myNCDMV. The background features a scenic view of autumn foliage. The page has "Log In" and "Sign Up" tabs at the top. The "Log In" tab is selected. It contains fields for "Email Address" (with placeholder "name@example.com") and "Password" (with placeholder "*****"). There is also a "Remember Me" checkbox. Below these are "Log In" and "Forgot Password" buttons. Further down, there are three social media log-in options: "Continue with Apple", "Continue with Facebook", and "Continue with Google". A "Continue as Guest" link is also present. A small notice at the bottom left reads: "NOTICE FOR PUBLIC COMPUTER USERS - If you sign in with Google, Apple, or Facebook you are also signing into that account on this computer. Remember to sign out when you're done." The page is powered by "payit".

mitchj@us.ibm.com

© 2018, 2022 IBM Corporation

Typical z/OS Connect EE API Requester security flow



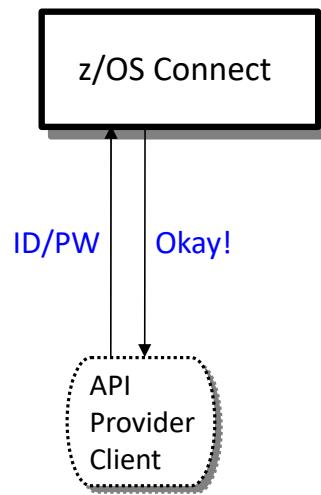
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS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

z/OS Application to z/OS Connect API Requester



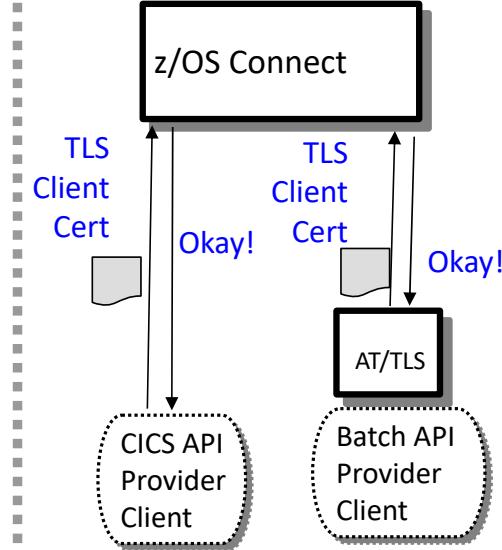
Two options for providing credentials for authentication

Basic Authentication



**Application provides
ID/PW or ID/PassTicket**

Client Certificate



**z/OS Connect requests a
client certificate**

**CICS or AT/TLS supplies a
client certificate**

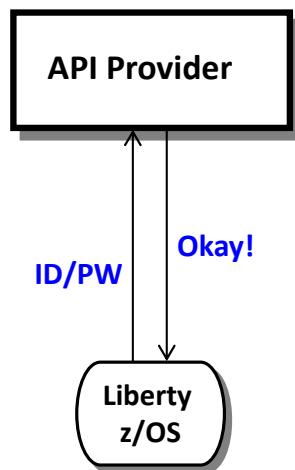
API Requester - API Provider Authentication



z/OS Connect EE

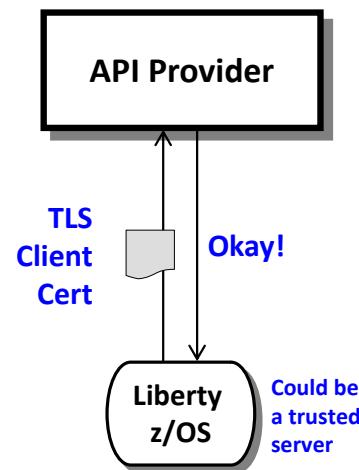
Several different ways this can be accomplished:

Basic Authentication



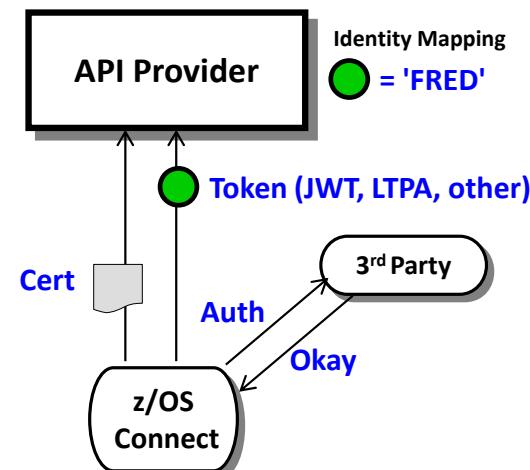
zCEE supplies ID/PW or
ID/Passticket

Client Certificate



Server prompts for certificate
zCEE supplies certificate

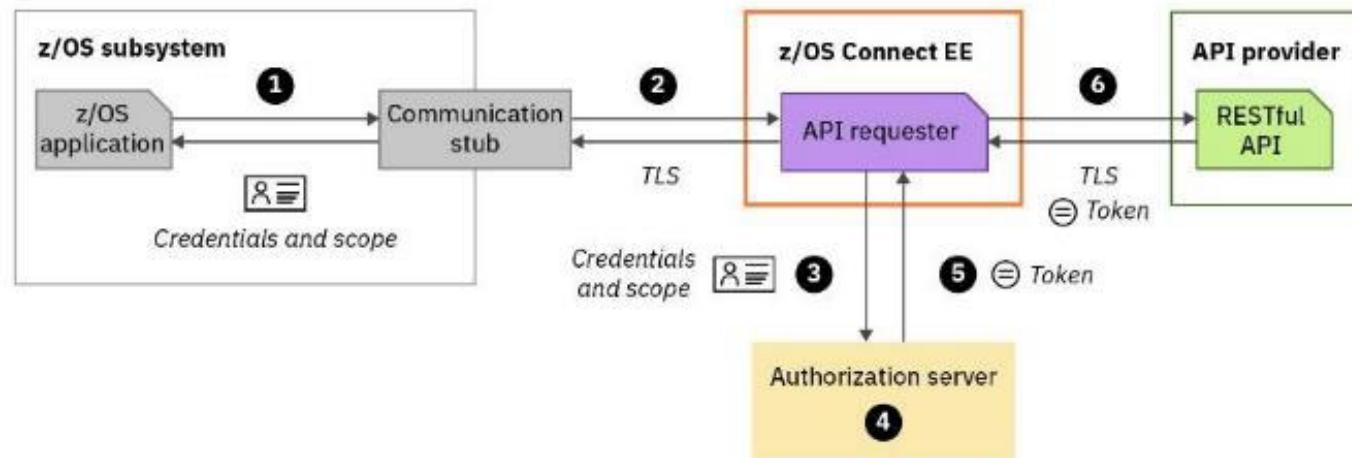
Third Party Authentication



zCEE authenticates to 3rd party sever
zCEE receives a trusted 3rd party token
Token flows to API Provider



Calling an API with OAuth 2.0 support





Configuring OAuth support – BAQRINFO copy book

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

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

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

Scope is always required.

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

Agenda

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

*For more on security, contact your local IBM rep regarding the schedule of workshop *zOSSEC1 IBM z/OS Connect Administration/Security Wildfire Workshop*
© 2018, 2022 IBM Corporation

z/OS Connect Wildfire Github Site

<https://ibm.biz/Bdf8BZ>



The screenshot shows a GitHub repository page for 'ibm-wsc/zCONNEE-Wildfire-Workshop'. The left sidebar lists various branches and topics, with 'exercises' highlighted and circled in red. The main content area shows a list of files under the 'master' branch, all uploaded by user 'emitchj'. The files are primarily PDFs related to developing APIs for various IBM services like CICS, IMS, MVS, RESTful APIs, and MQ.

File Name	Description	Last Updated
Developing CICS API Requester Applications.pdf	Add files via upload	2 months ago
Developing IMS API Requester Applications.pdf	Add files via upload	2 months ago
Developing MVS Batch API Requester Applications.pdf	Add files via upload	2 months ago
Developing RESTful APIs for DVM VSAM Services.pdf	Add files via upload	20 days ago
Developing RESTful APIs for DVM VSAMCICS Services.pdf	Add files via upload	20 days ago
Developing RESTful APIs for Db2 REST Services.pdf	Add files via upload	2 months ago
Developing RESTful APIs for HATS REST Services.pdf	Add files via upload	2 months ago
Developing RESTful APIs for IMS Database REST Services....	Add files via upload	2 months ago
Developing RESTful APIs for IMS Transactions.pdf	Add files via upload	2 months ago
Developing RESTful APIs for MQ.pdf	Add files via upload	2 months ago
Developing RESTful APIs for MVS Batch.pdf	Add files via upload	2 months ago
Developing RESTful APIs for a CICS COMMAREA program.pdf	Add files via upload	2 months ago
Developing RESTful APIs for a CICS Container program.pdf	Add files via upload	2 months ago

- Contact your IBM representative to schedule access to these exercises

mitchj@us.ibm.com

© 2018, 2022 IBM Corporation



/questions?thanks=true

Thank you for listening.

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