



IBM z/OS Connect Enterprise Edition

Introduction and Overview

Mitch Johnson

mitchj@us.ibm.com

Washington System Center



IBM WSC IBM Z Systems
Wildfire Team

Agenda

- z/OS Connect Introduction and overview
- Self paced, hands-on exercises to API enable z application from various sub-systems, e.g.
 - CICS
 - DB2
 - IMS/TM
 - MQ
 - IBM DVM
 - IBM File Manager
 - MVS Batch
 - Outbound REST APIs
 - 3270 screen based applications
- z/OS Connect Security

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



z/OS Connect EE



© 2018, 2019 IBM Corporation

* Other Vendors or your own implementation

/but_first, what_is_REST?

What makes an API “RESTful”?



z/OS Connect EE

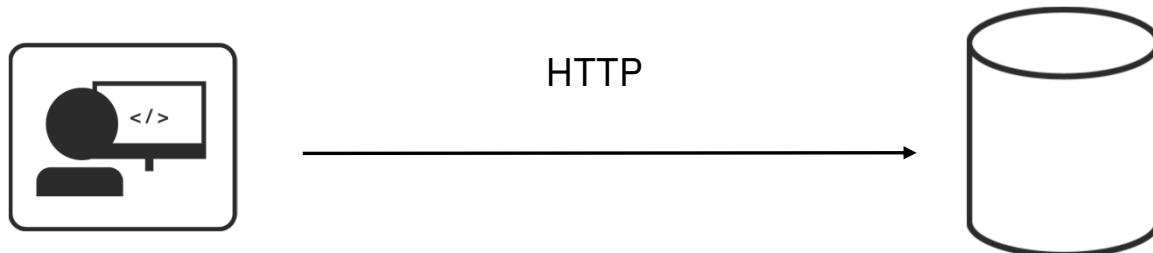
REST is an Architectural Style

REST stands for **R**epresentational **S**tate **T**ransfer.

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

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

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



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



z/OS Connect EE

Key Principles of REST

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

GET
POST
PUT
DELETE

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

Path and Query parameters are used for refinement of the request

URIs represent things (or lists of things)

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
- It should access all the resources from the server using only URI
- For performing CRUD operations, it should use HTTP verbs such as get, post, put and delete
- It should return the result only in the form of JSON
- REST based services follow some of the above principles and not all, whereas RESTful means it follows all the above principles.
- Remember - Not all REST APIs are RESTful APIs
- The key is consistency, RESTful APIs are consistent, REST APIs are not

RESTful Examples



z/OS Connect EE

z/OS Connect Enterprise Edition:

POST /account?name=Fred +  (*JSON with Fred's information*)

GET /account?number=1234

PUT /account?number=1234 +  (*JSON 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>



z/OS Connect EE

Not every REST API is a RESTful API

(How to know if you are doing it wrong)

1. Unique URIs for different 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

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 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" }



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.



z/OS Connect EE

Why use Swagger?

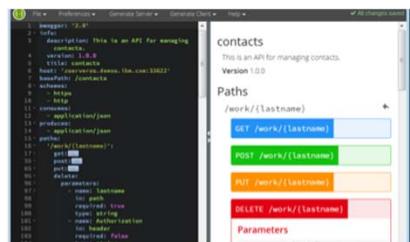
It is more than just an API framework



There are a number of tools available to aid consumption:

Write Swagger

Swagger Editor allows API developers to design their swagger documents.



Read Swagger

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



Consume Swagger

Swagger Codegen create stub code to consume APIs from various languages



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

© 2018, 2019 IBM Corporation

Example: <https://developer.psa-peugeot-citroen.com/inc/>



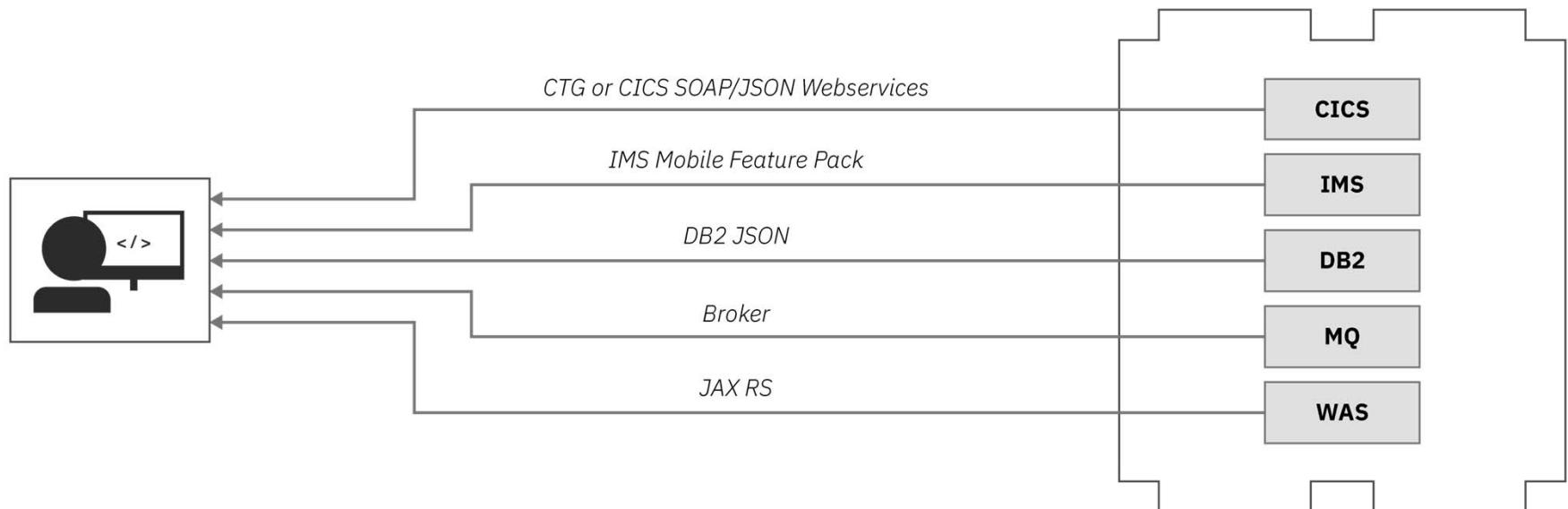
Why /zos_connect_ee?

Truly RESTful APIs to and from your mainframe.

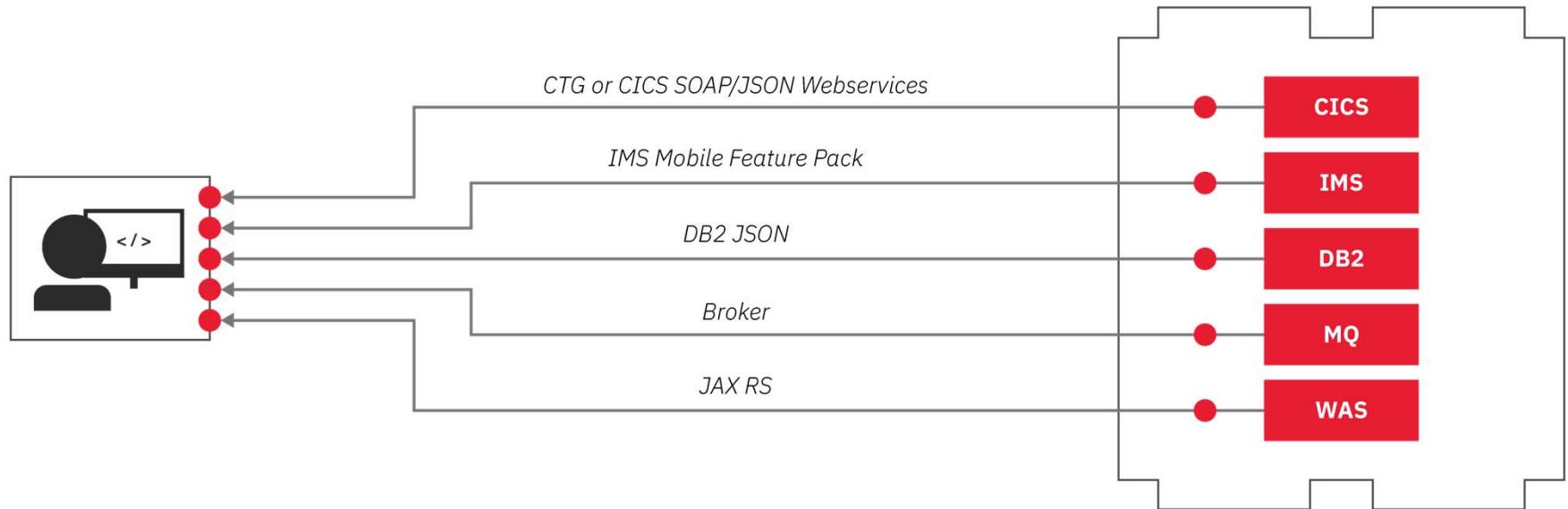
Can't we do REST and JSON today?



z/OS Connect EE



Yes, but....



Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

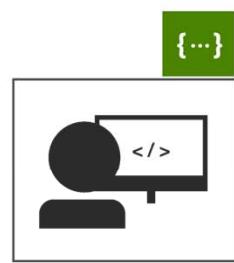
These are typically not RESTful!

A single entry point is needed

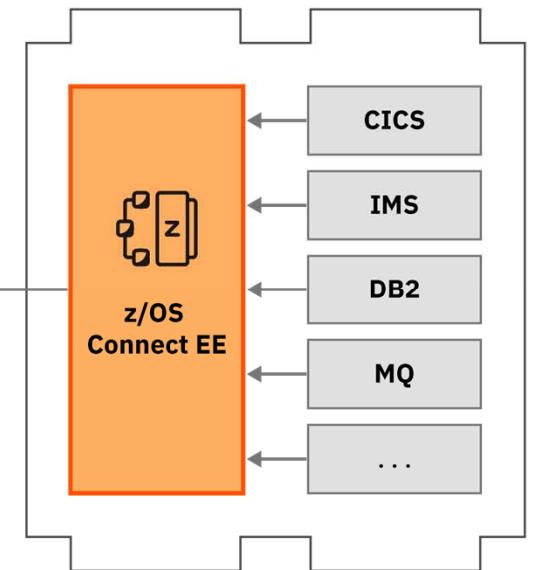


z/OS Connect EE

Expose z/OS resources without writing any code.



RESTful APIs available from one endpoint



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



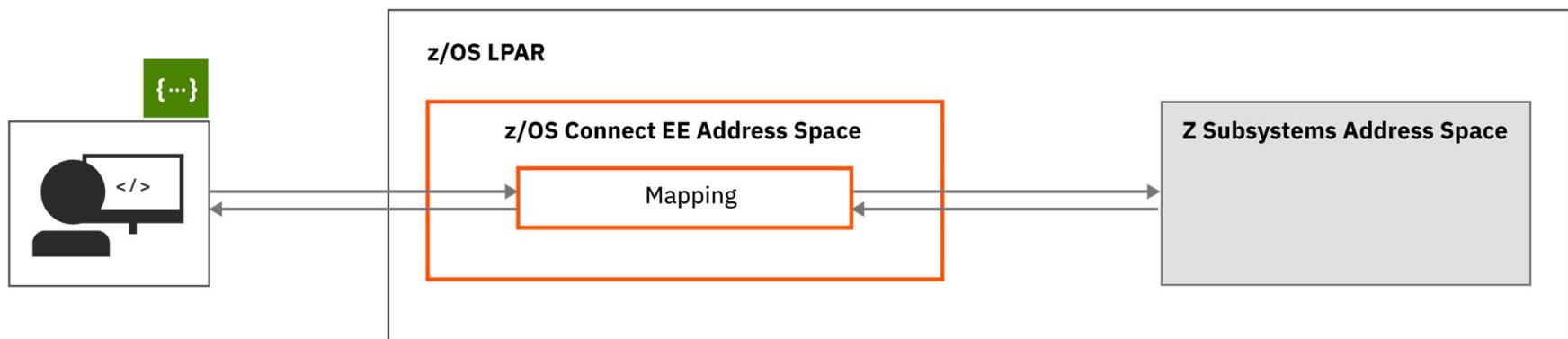
**Other than a RESTful interface,
what does z/OS Connect provide?**

Let's Start with Data mapping



z/OS Connect EE

Converting from JSON to the target's subsystem's format

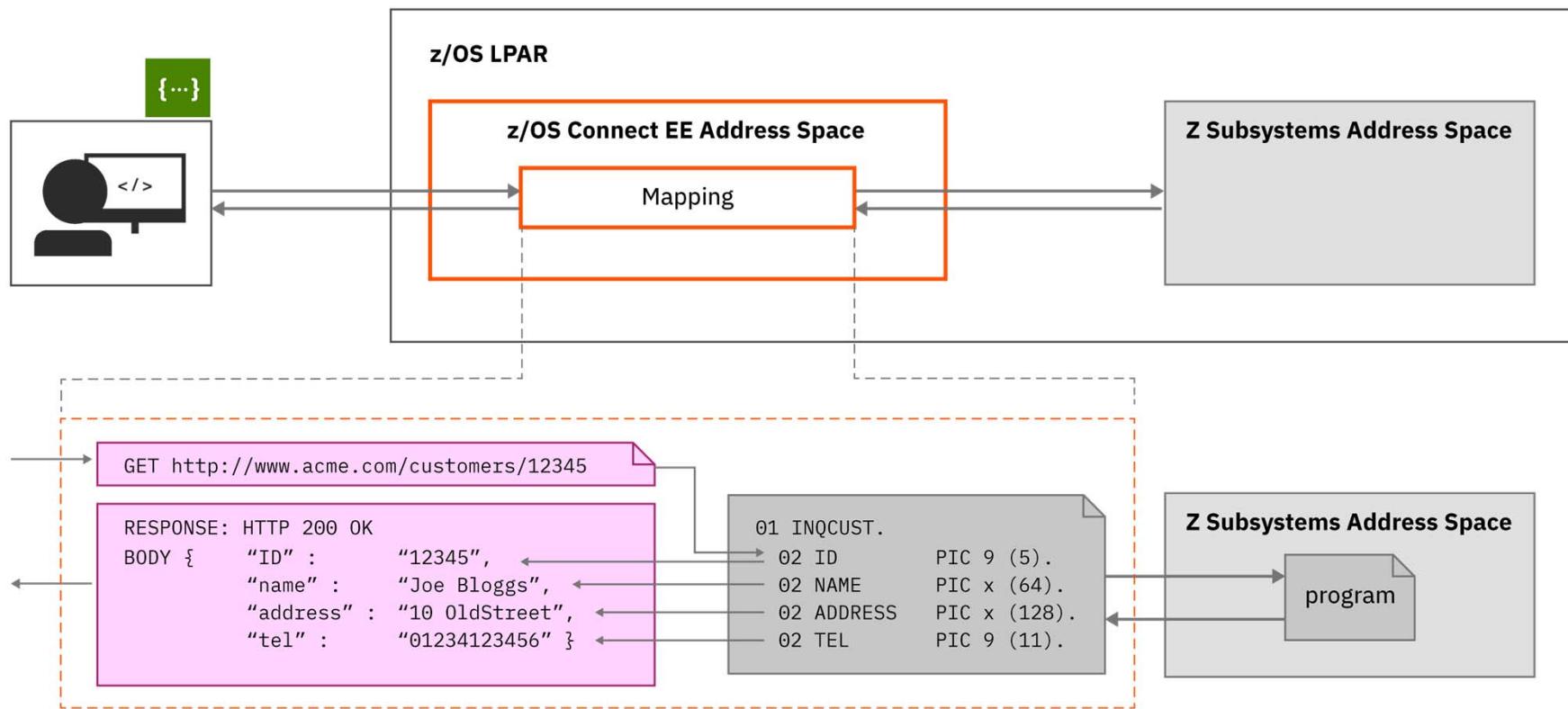




z/OS Connect EE

Data mapping Example

A closer look





COBOL versus JSON 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 99 times  
      depending on messages-Num.
```

```
"miniloan_commarearea": {  
    "type": "object",  
    "properties": {  
        "name": {  
            "type": "string",  
            "maxLength": 20  
        },  
        "creditScore": {  
            "type": "number",  
            "format": "decimal",  
            "multipleOf": 0.01,  
            "maximum": 9999999999999999.99,  
            "minimum": 0  
        },  
        "yearlyIncome": {  
            "type": "string",  
            "format": "decimal",  
            "multipleOf": 0.01,  
            "maximum": 9999999999999999.99,  
            "minimum": 0  
        },  
        "age": {  
            "type": "string",  
            "format": "decimal",  
            "multipleOf": 1,  
            "maximum": 9999999999999999.99,  
            "minimum": 0  
        },  
        "amount": {  
            "type": "string",  
            "format": "decimal",  
            "multipleOf": 0.01,  
            "maximum": 9999999999999999.99,  
            "minimum": 0  
        },  
        "approved": {  
            "type": "boolean",  
            "value": true  
        },  
        "effectDate": {  
            "type": "string",  
            "format": "date",  
            "multipleOf": 1,  
            "maximum": "9999-12-31",  
            "minimum": "0001-01-01"  
        },  
        "yearlyInterestRate": {  
            "type": "string",  
            "format": "decimal",  
            "multipleOf": 0.01,  
            "maximum": 9999999999999999.99,  
            "minimum": 0  
        },  
        "yearlyRepayment": {  
            "type": "string",  
            "format": "decimal",  
            "multipleOf": 0.01,  
            "maximum": 9999999999999999.99,  
            "minimum": 0  
        },  
        "messages-Num": {  
            "type": "string",  
            "format": "decimal",  
            "multipleOf": 1,  
            "maximum": 9999999999999999.99,  
            "minimum": 0  
        },  
        "messages": {  
            "type": "array",  
            "items": {  
                "type": "string",  
                "format": "date",  
                "multipleOf": 1,  
                "maximum": "9999-12-31",  
                "minimum": "0001-01-01"  
            }  
        }  
    }  
}
```

COBOL Source v JSON

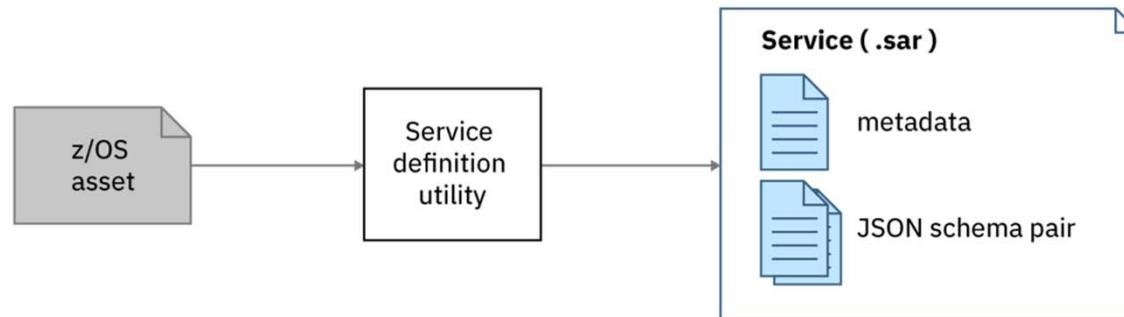
“name”:“Mitch Johnson”,
“creditScore”:100

All data is sent as character strings and numeric precision and sign bit is removed as an issue

Steps to expose a z/OS application

1. Create a service definition

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



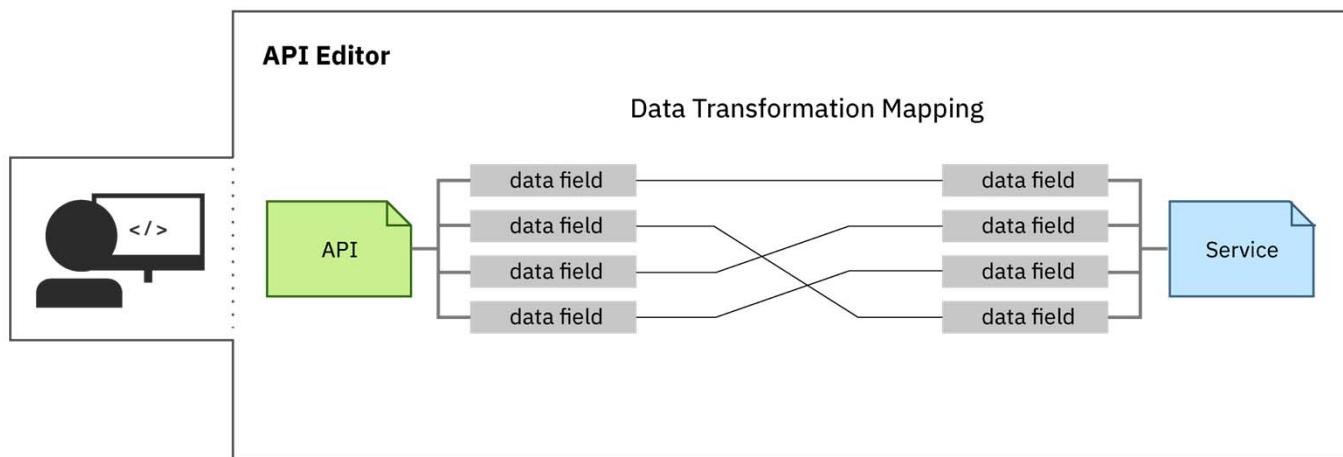
Use a system-appropriate utility to generate a `.sar` file for the z/OS application

- API Toolkit (CICS and IMS)
- BAQLS2JS (MQ and WOLA)
- z/OS Connect EE Build Toolkit (DB2 and HATS)
- DVM Toolkit

 ibm.biz/zosconnect-sar-creation

Steps to expose a z/OS application

2. Create an API



Import your `.sar` file into the **API toolkit**, and start designing your API.

From the editor, create an **API Archive file** (`.aar`), which describes your API and how it maps to underlying services.

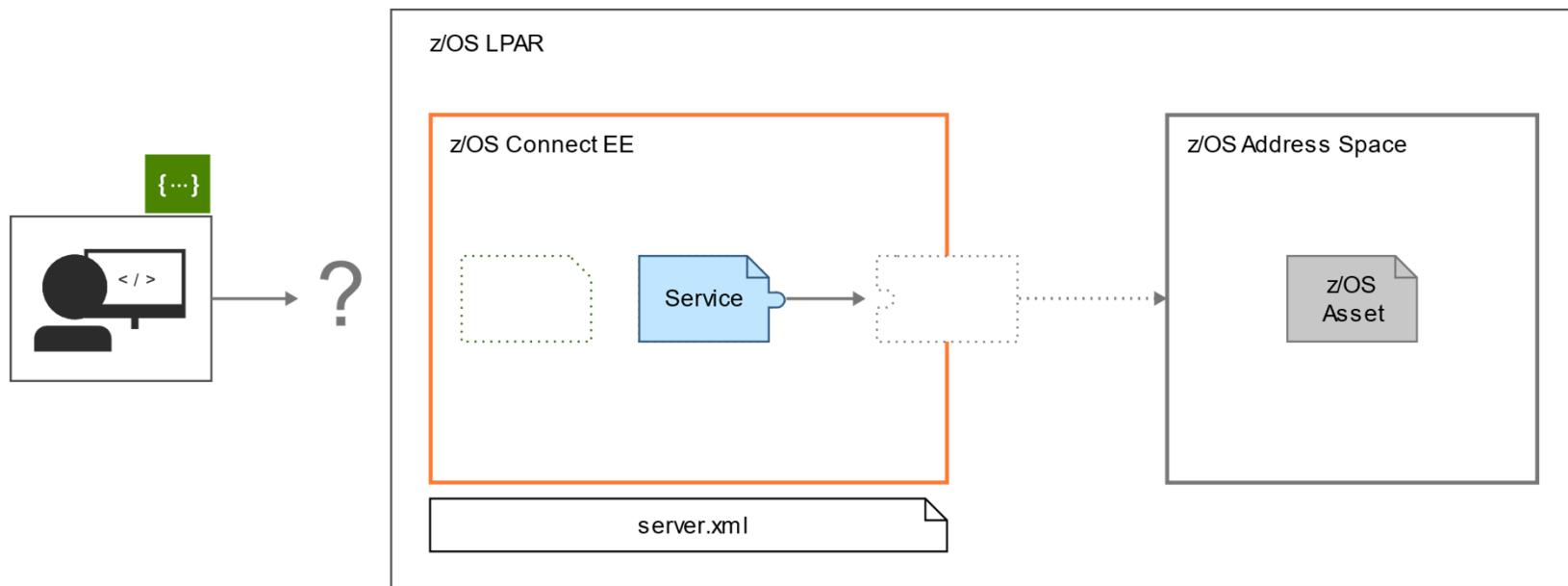
 ibm.biz/zosconnect-create-api



z/OS Connect EE

Steps to expose a z/OS application

3. Deploy your service



Deploy the `.sar` file generated in **Step 2** using the right-click deploy in **the API toolkit**, or by copying the `.sar` file to the services directory.



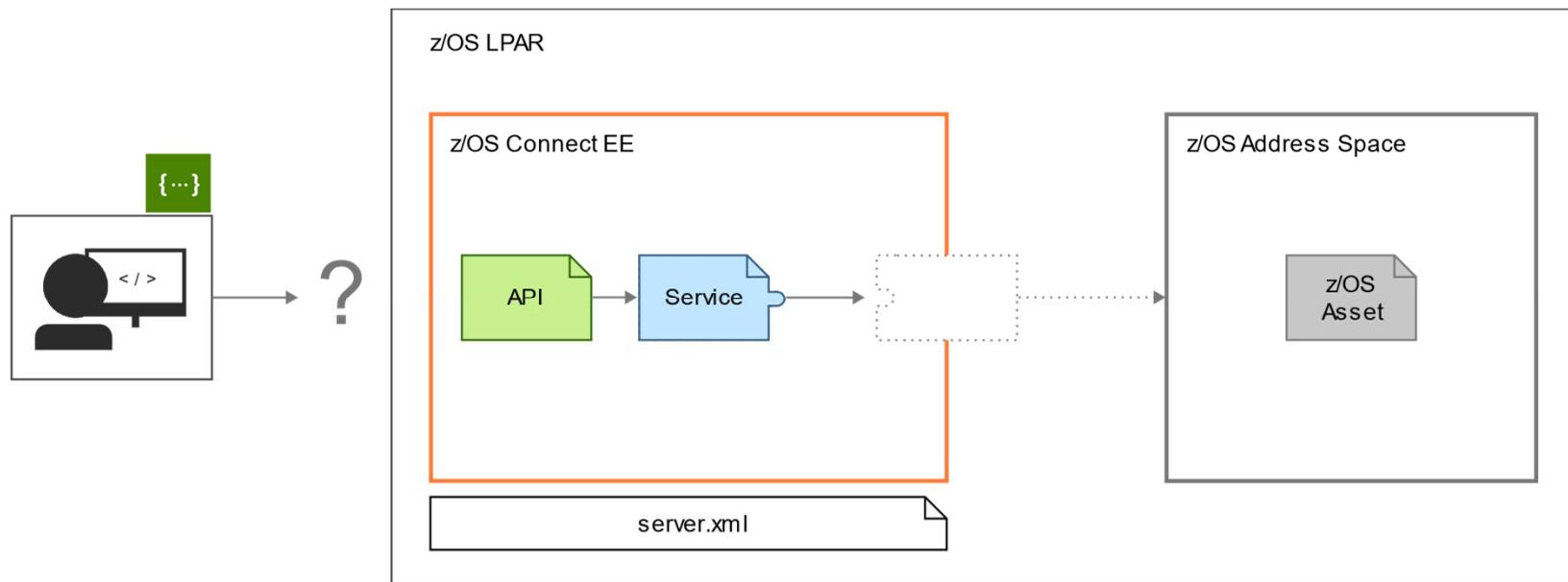
ibm.biz/zosconnect-define-services



z/OS Connect EE

Steps to expose a z/OS application

4. Deploy your API



Deploy your API using the right-click deploy in **the API toolkit**, or by copying the `.aar` file to the `apis` directory.

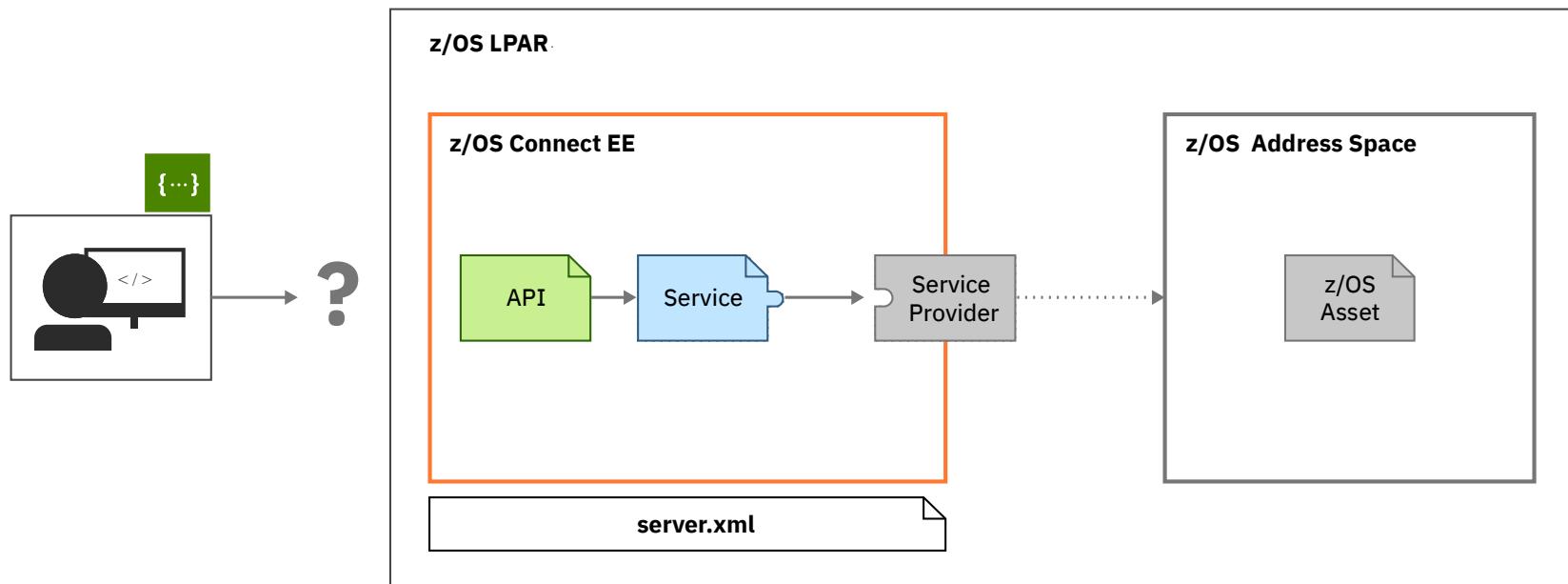
 ibm.biz/zosconnect-deploy-api



z/OS Connect EE

Steps to expose a z/OS application

5. Configure your service provider



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



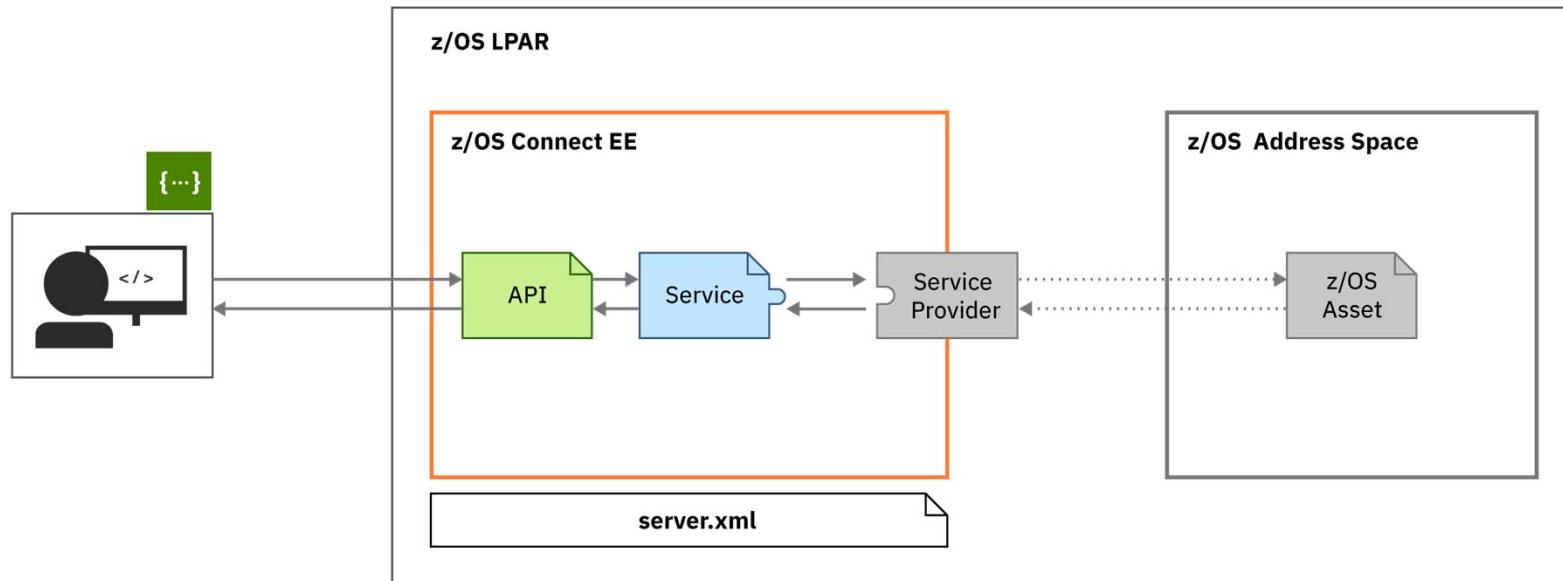
ibm.biz/zosconnect-configuring



z/OS Connect EE

Steps to expose a z/OS application

6. Done



Your API is ready to be consumed: go tell your developers!



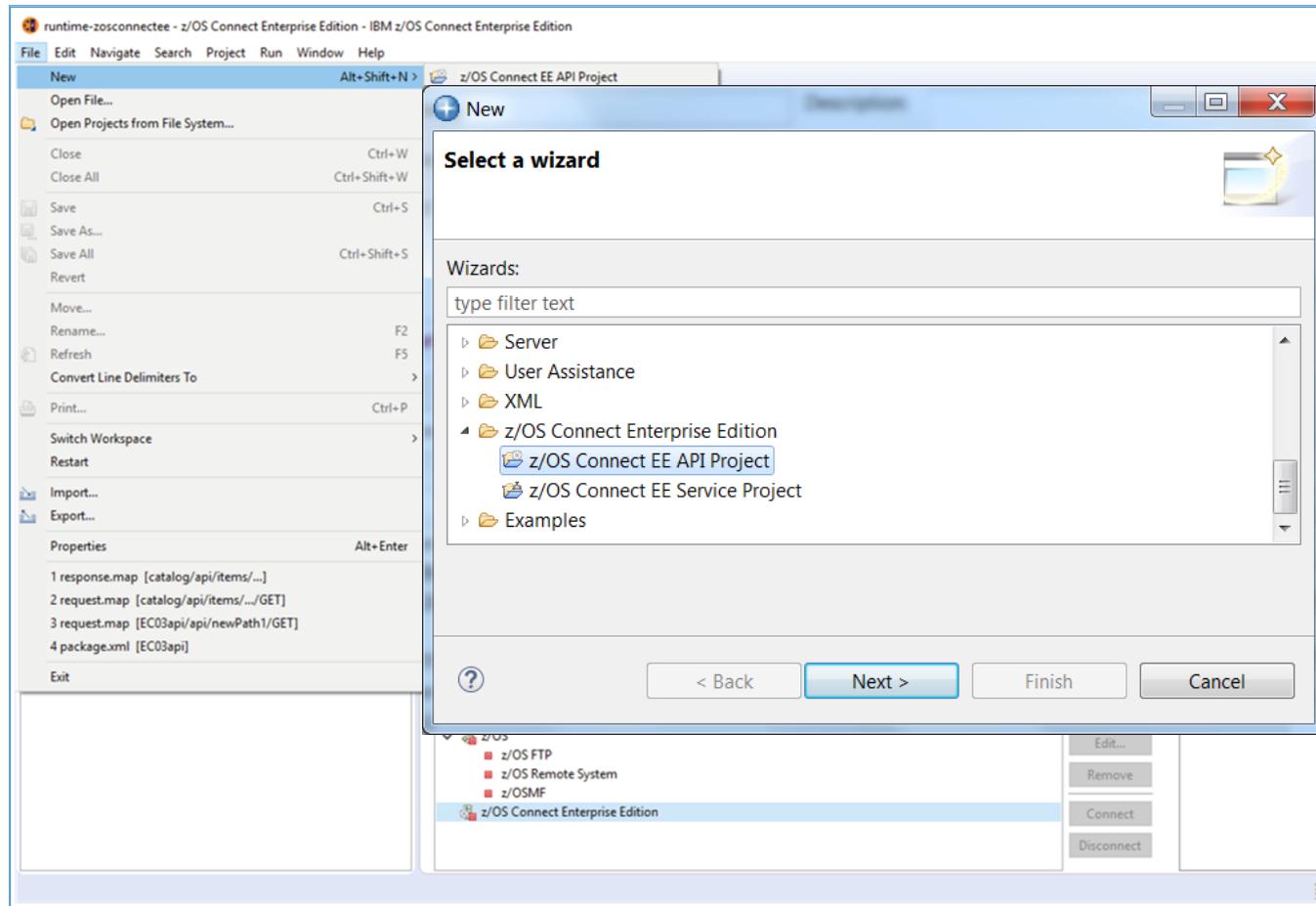
/api_toolkit/services

Simple **service creation.**

API toolkit – Creating Services for CICS and IMS



z/OS Connect EE



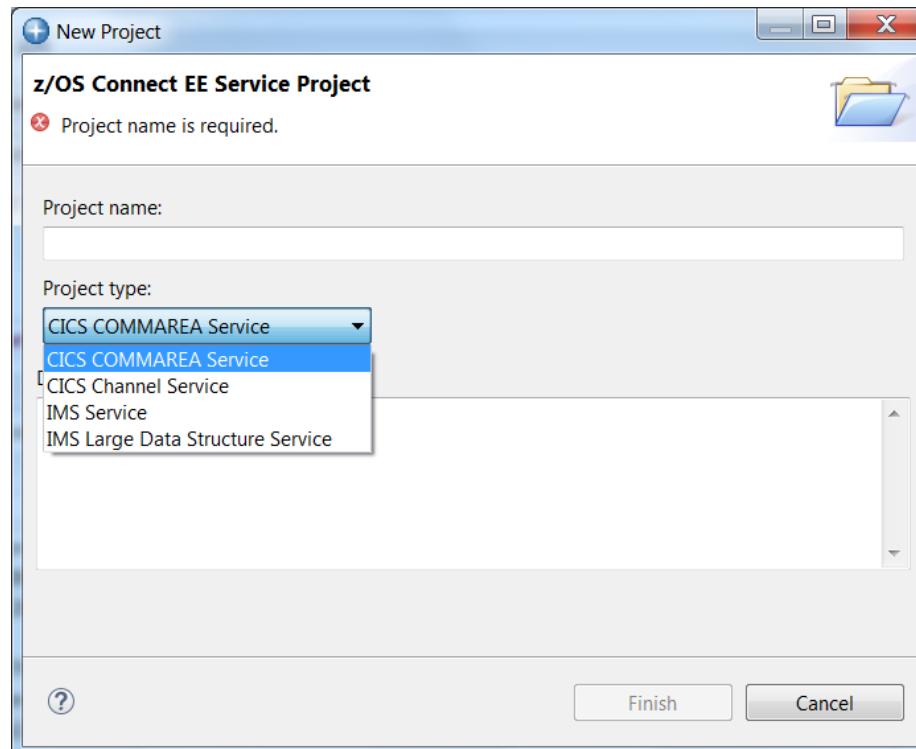
Use the **API toolkit** to create services through Eclipse-based tooling.

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

API toolkit – Creating Services for CICS and IMS



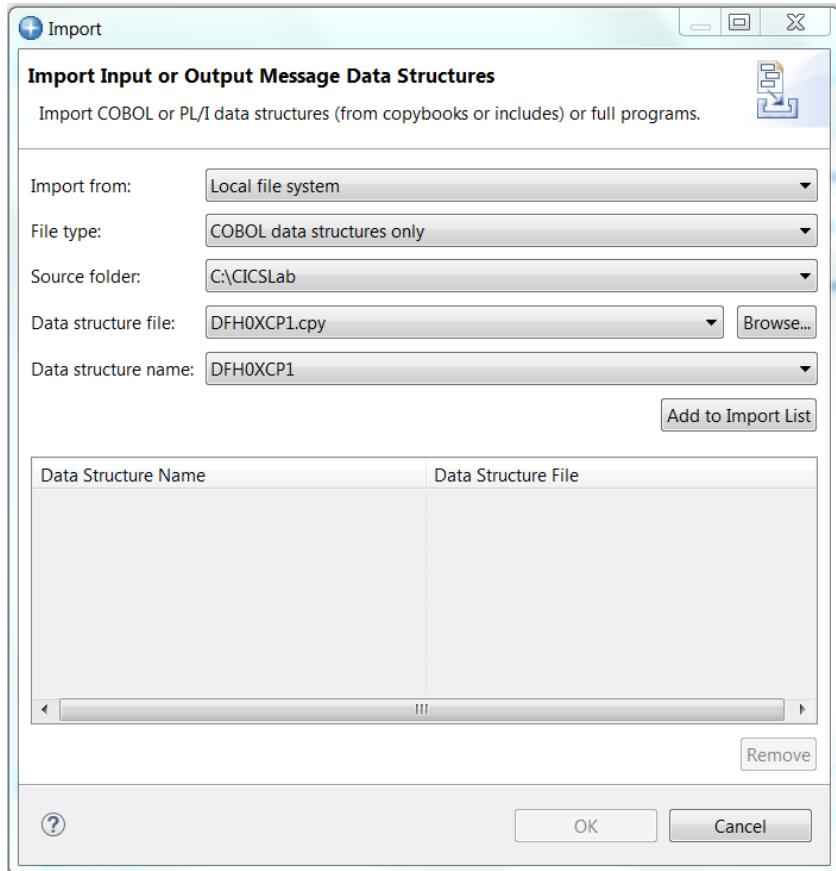
Service creation – a common interface



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

API toolkit – Creating Services for CICS and IMS

Creating a service project



You start by importing data structures into the service interface from the local file system or the workspace.

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

API toolkit – Creating Services for CICS and IMS



Creating a service interface definition

*inquireSingle Service *inquireSingleRequest

Service Interface Definition

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

Search:

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	<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 redefines CA_INQUIRE_SINGLE redefines CA_ORDER_REQUEST	<input type="checkbox"/>	CA_INQUIRE REQUEST		STRUCT	911	88
CA_ITEM_REF_REQ	<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_SINGLE redefines CA_INQUIRE REQUEST	<input type="checkbox"/>	CA_ORDER_REQUEST		STRUCT	911	88

You 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.

API toolkit – Creating Services for CICS and IMS

Creating a service – response message



z/OS Connect EE

*inquireSingleResponse

Service Interface Definition

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

Search: ↓ ↑ ↶ ↷ ↶↶ ↷↷ ↶↶↶ ↷↷↷ ↶↶↶↶ ↷↷↷↷ ↶↶↶↶↶ ↷↷↷↷↷ ↶↶↶↶↶↶ ↷↷↷↷↷↷ ↶↶↶↶↶↶↶ ↷↷↷↷↷↷↷

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	<input 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 (Redefine)	<input type="checkbox"/>	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines	<input 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 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_INQUIRE_SINGLE	<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

You can then see the imported data structure and can **redact fields** and **rename fields**



z/OS Connect EE

API toolkit – Creating Services for CICS and IMS

Creating a “GET” service interface request definition

*invtnoDisplayService Service *invtnoDisplayRequest

Service Interface Definition

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

Search:

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
ivtnoDisplayRequest	<input type="checkbox"/>					
Segment 1	<input type="checkbox"/>					
INPUT_MSG	<input checked="" type="checkbox"/>					
IN_LL	<input type="checkbox"/>	IN_LL		SHORT	2	1
IN_ZZ	<input type="checkbox"/>	IN_ZZ		SHORT	2	3
IN_TRANCDE	<input type="checkbox"/>	IN_TRANCDE		CHAR	10	5
IN_COMMAND	<input type="checkbox"/>	IN_COMMAND		CHAR	8	15
IN_LAST_NAME	<input checked="" type="checkbox"/>	lastName	IVTNO DISPLAY	CHAR	10	23
IN_FIRST_NAME	<input type="checkbox"/>	IN_FIRST_NAME		CHAR	10	33
IN_EXTENSION	<input type="checkbox"/>	IN_EXTENSION		CHAR	10	43
IN_ZIP_CODE	<input type="checkbox"/>	IN_ZIP_CODE		CHAR	7	53

The service developer creates distinct services for each function.

DISPLAY
DELETE
ADD
UPDATE



z/OS Connect EE

API toolkit – Creating Services for CICS and IMS

Creating a “GET” service interface response definition

*invtnoDisplayService Service *invtnoDisplayResponse

Service Interface Definition

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

Search:

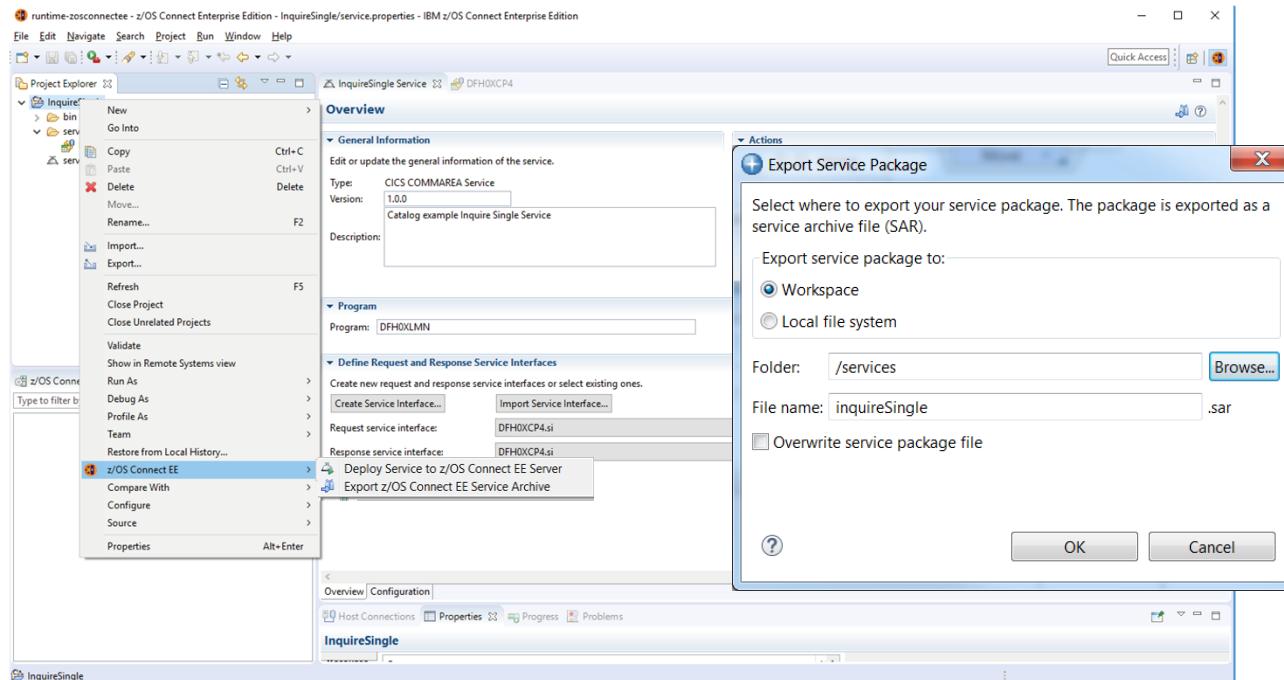
Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
ivtnoDisplayResponse	<input type="checkbox"/>					
Segment 1	<input type="checkbox"/>					
OUTPUT_AREA	<input checked="" type="checkbox"/>					
OUT_LL	<input type="checkbox"/>	OUT_LL		SHORT	2	1
OUT_ZZ	<input type="checkbox"/>	OUT_ZZ		SHORT	2	3
OUT_MESSAGE	<input checked="" type="checkbox"/>	message		CHAR	40	5
OUT_COMMAND	<input type="checkbox"/>	OUT_COMMAND		CHAR	8	45
OUT_LAST_NAME	<input checked="" type="checkbox"/>	lastName		CHAR	10	53
OUT_FIRST_NAME	<input checked="" type="checkbox"/>	firstName		CHAR	10	63
OUT_EXTENSION	<input checked="" type="checkbox"/>	extension		CHAR	10	73
OUT_ZIP_CODE	<input checked="" type="checkbox"/>	zipCode		CHAR	7	83
OUT_SEGMENT_NO	<input type="checkbox"/>	OUT_SEGMENT_NO		CHAR	4	90

API toolkit – Creating Services for CICS and IMS



z/OS Connect EE

Creating a service for CICS and IMS



Finally, you can export the service project as a **Service Archive file (.sar)**.

API toolkit – Creating Services for CICS and IMS



z/OS Connect EE

Creating a service for CICS and IMS

The screenshot shows the 'InquireSingle Service' configuration window in the IBM z/OS Connect Enterprise Edition. The 'General Information' section shows the service type as 'CICS COMMAREA Service' and version '1.0.0'. The 'Program' field contains 'DFH0XLMN'. Under 'Actions', steps for creating a service are listed: 1. Input service version, 2. Specify program or transaction code for the service, 3. Create or import a service interface for the request and response in your service, 4. Complete the configuration for the service, and 5. Export the service. The 'Define Request and Response Service Interfaces' section shows two interfaces: 'Request service interface: DFH0XCP4.si' and 'Response service interface: DFH0XCPA.si'. A context menu for 'z/OS Connect EE' is open, showing options like 'Compare With', 'Configure', and 'Source'. A separate 'Deploy Service' dialog box is open, showing the target server as 'wg31:9453' and the services to be created: 'inquireSingle' (Version 13.00, Type CICS COMMAREA Se...). The dialog has 'OK' and 'Cancel' buttons.

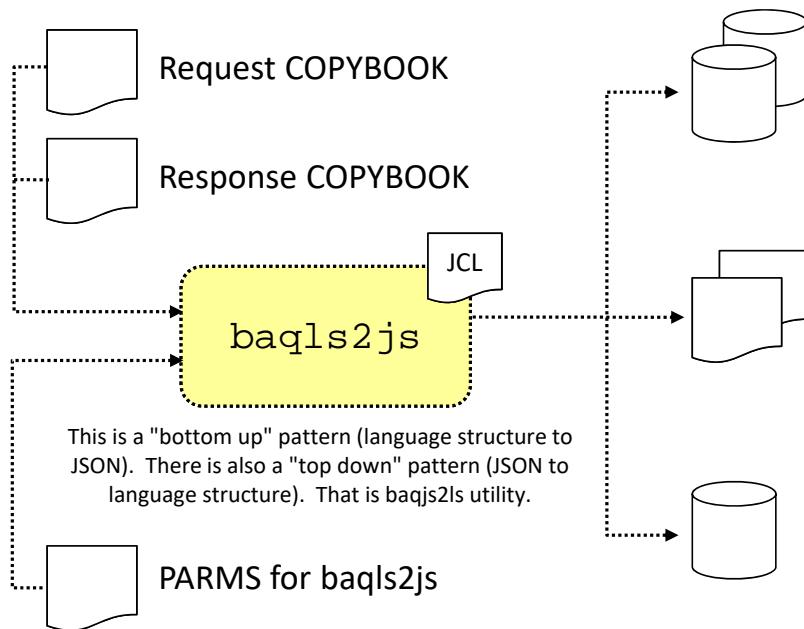
Finally, you can deploy the service project as a **Service Archive file (.sar)**

Creating Services without the Toolkit - 1



z/OS Connect EE

For MQ and MVS Batch use the supplied conversion utility BAQLS2JS



BIND Files

These are binary-format files that contain information about the field definitions and the data transformation requirements.

These are placed in a USS file system location based on input parms you specify.

JSON Schema Files

These provide the JSON schema used to interact with the backend program based on the COPYBOOK data requirements.

These are placed in a USS file system location based on input parms you specify.

Service Archive (SAR) File

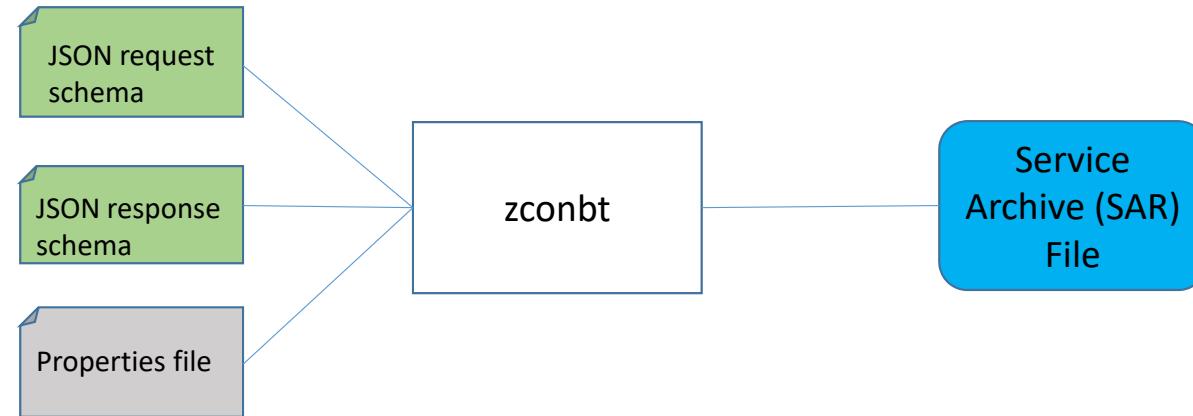
This is a ZIP-format file that contains the JSON schema and some meta-data. This is input to the API Editor (next unit) to create the APIs.

This is placed in a USS file system location based on input parms you specify.

Creating Services without the Toolkit – 2a



For DB2 and HATS REST Services use the z/OS Connect Build toolkit (zconbt)

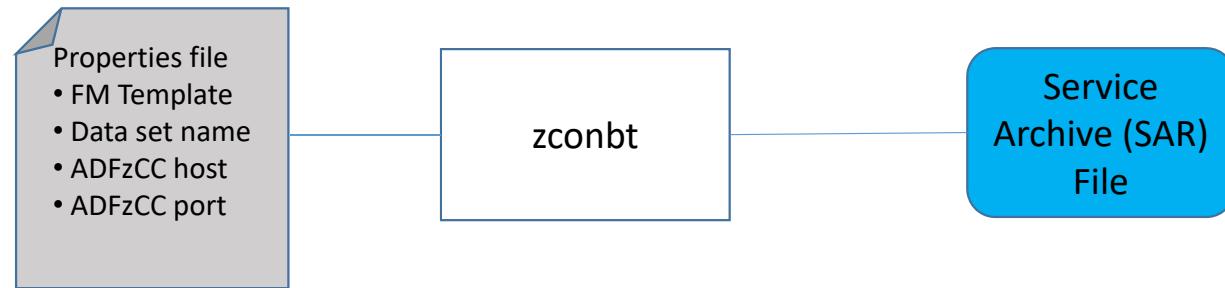


Generate the service archive file

Creating Services without the Toolkit – 2b



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



Generate the service archive file

Creating Services without the Toolkit – Part 3



z/OS Connect EE

For DVM use the DVM Studio

The screenshot shows the DVM Studio interface with a red border around the main window. On the left, the navigation pane lists various options like Services, New Target System, and z/OS Connect Configuration. The central pane displays a tree view under 'Services' with 'Web Services' expanded, showing 'Create Service' selected. A context menu is open over this item, with 'z/OS Connect REST Interface' highlighted. The right pane shows two tabs: 'Generated.sql' containing two SQL queries for retrieving catalog data, and 'SQL Res...' showing a result set for the first query. A small red circle highlights the 'Generate SAR File(s)' option in the bottom right corner of the context menu.

```
-- Description: Retrieve the result set for CATALOG (up to 1000 rows)
-- Tree Location: wg31.washington.ibm.com/1200/SQL/Data/AVZS/Virtual Table
-- Remarks: VSAM - USER1.EXAMPLAPP.EXMPCAT
SELECT WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTMENT, WS_COST, WS_IN_STOCK,
WS_ON_ORDER
FROM CATALOG LIMIT 1000;

-- Description: Retrieve the result set for CATALOG (up to 1000 rows)
-- Tree Location: wg31.washington.ibm.com/1200/SQL/Data/AVZS/Virtual Table
-- Remarks: VSAM - USER1.EXAMPLAPP.EXMPCAT
SELECT WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTMENT, WS_COST, WS_IN_STOCK,
WS_ON_ORDER
FROM CATALOG LIMIT 1000;
```

VS_DESCRIPTION	WS_DEPARTMENT	WS_COST	WS_IN_STOCK	WS_ON_ORDER
All Pens ...	10	002.90	135	0
All Pens ...	10	002.90	6	50
All Pens ...	10	002.90	106	0
Pencil wit...	10	002.90	80	0
Pencil wit...	10	001.78	83	0
Highlighter	10	003.89	12	40



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

API definition

The screenshot shows the z/OS Connect EE API Editor interface with two API definitions:

- catalog API**:
 - Name:** catalog
 - Description:** APIs for browsing, inquiring and ordering items from a catalog
 - Base path:** /catalogManager
 - Version:** 1.0.0
 - Path:** /items?startItem
 - Methods**:
 - GET inquireCatalog
 - POST placeOrder
- Items API**:
 - Path:** /items/{itemId}
 - Methods**:
 - GET inquireSingle
 - PUT inquireSingle
 - DELETE inquireSingle



z/OS Connect EE

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

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

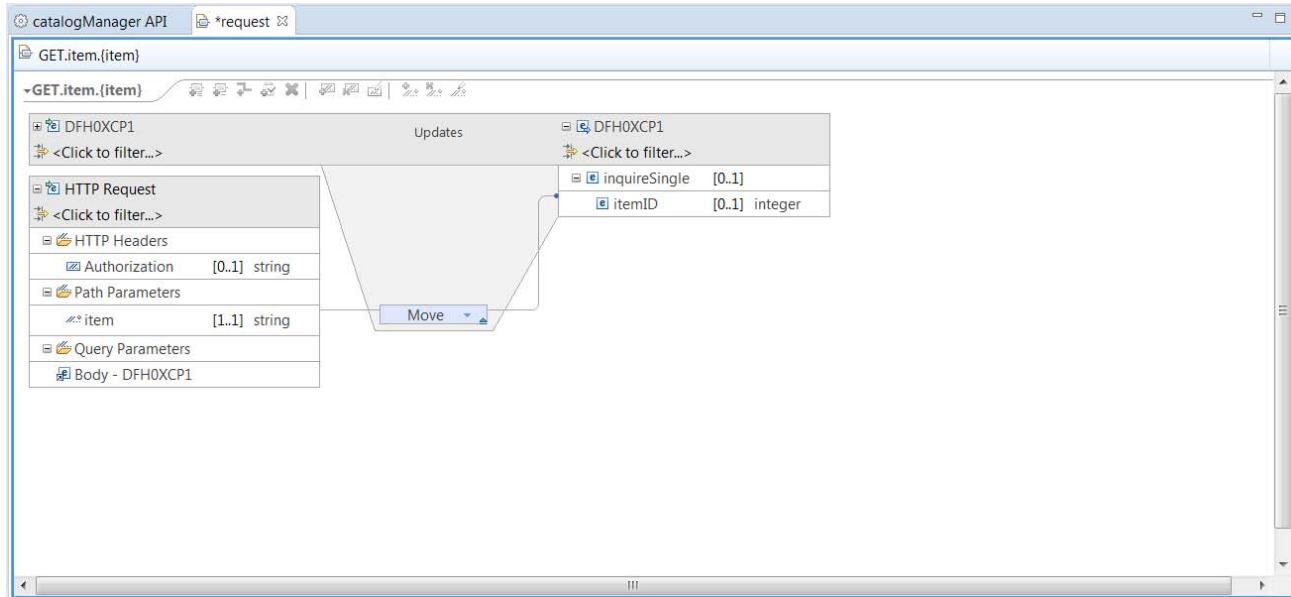
Your services are represented by **.sar** files, which you import into the **API toolkit**, regardless of how the .sar was generated.



z/OS Connect EE

API toolkit – API Editor

API mapping: Point-and-click interface



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).



z/OS Connect EE

API toolkit

API definition with multiple response codes

The screenshot shows the API toolkit interface for defining an API. At the top, under 'Methods (4)', there are four entries: POST, GET, PUT, and DELETE, all pointing to 'IVTNOService'. The 'GET' entry is expanded, showing its 'Operation id' as 'getContact'. Below this, under 'Responses (3)', three response codes are defined: 200 OK, 404 Not Found, and 400 Bad Request. The '400' entry is selected, opening a dialog titled 'Edit Response 400'. In this dialog, the 'Response code' is set to '404 - Not Found' and the 'Description' is 'Not Found'. The 'Rule' section contains two rules: 'Rule 1' (response/status starts with 'SPECIFIED PERSON') and 'Rule 2' (response/status ends with 'NOT FOUND'). A summary at the bottom shows '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



z/OS Connect EE

API toolkit – API Editor

API mapping: Point-and-click interface

Allows the API Developer to remove fields to simplify the API

The screenshot shows the API Editor interface for mapping API responses. On the left, the API endpoint `DELETE.employee.{numb}` is selected. The main area displays the `cscvincServiceOperationResponse` structure under the `Updates` section. A detailed view of the `Body - cscvincServiceOperationResponse` object is shown, containing the following fields:

- `Container1` [1..1]
- `ACTION` [0..1] string
- `CEIBRESP` [0..1] integer
- `CEIBRESP2` [0..1] integer
- `USERID` [0..1] string
- `FILEA_AREA` [0..1] string
- `STAT` [0..1] string
- `NUMB` [0..1] string
- `NAME` [0..1] string
- `ADDRX` [0..1] string
- `PHONE` [0..1] string
- `DATEX` [0..1] string
- `AMOUNT` [0..1] string
- `COMMENT` [0..1] string

Each field in the `Body` object has a "Remove" button associated with it. Below the `Body` object, there is a separate `HTTP Response` section containing `HTTP Headers` and the `Body - cscvincServiceOperationResponse`.

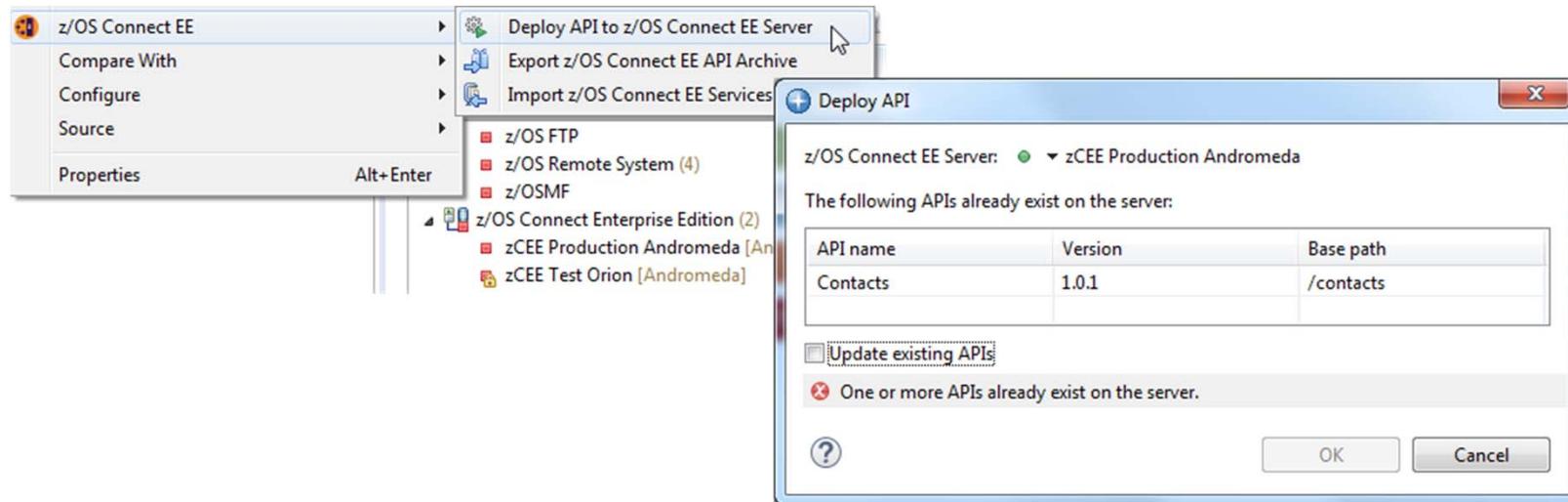


z/OS Connect EE

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.



z/OS Connect EE

API toolkit – API Editor

Testing with Swagger UI

The screenshot shows the z/OS Connect EE API Editor interface. On the left, there is a tree view of servers and services. A context menu is open over an API entry, with the option "Open In Swagger" selected. This has triggered the right-hand panel to switch to a Swagger UI view. The Swagger UI shows a "catalog" endpoint with three methods: GET /item/{itemID}, GET /items, and POST /orders. The POST /orders method is highlighted with a green background. The "Model" tab is active, displaying the JSON schema for the response:

```
{  
  "DFH0XCP1": {  
    "CA_RETURN_CODE": 0,  
    "CA_RESPONSE_MESSAGE": "string",  
    "CA_INQUIRE_SINGLE": {  
      "CA_SINGLE_ITEM": {  
        "CA_SNGL_ITEM_REF": 0,  
        "CA_SNGL_DESCRIPTION": "string",  
        "CA_SNGL_DEPARTMENT": 0,  
        "CA_SNGL_COST": "string",  
      }  
    }  
  }  
}
```

The "Parameters" section shows two parameters: "itemID" (required, path, string) and "Authorization" (header, string). A "Try it out!" button is present at the bottom of the UI.

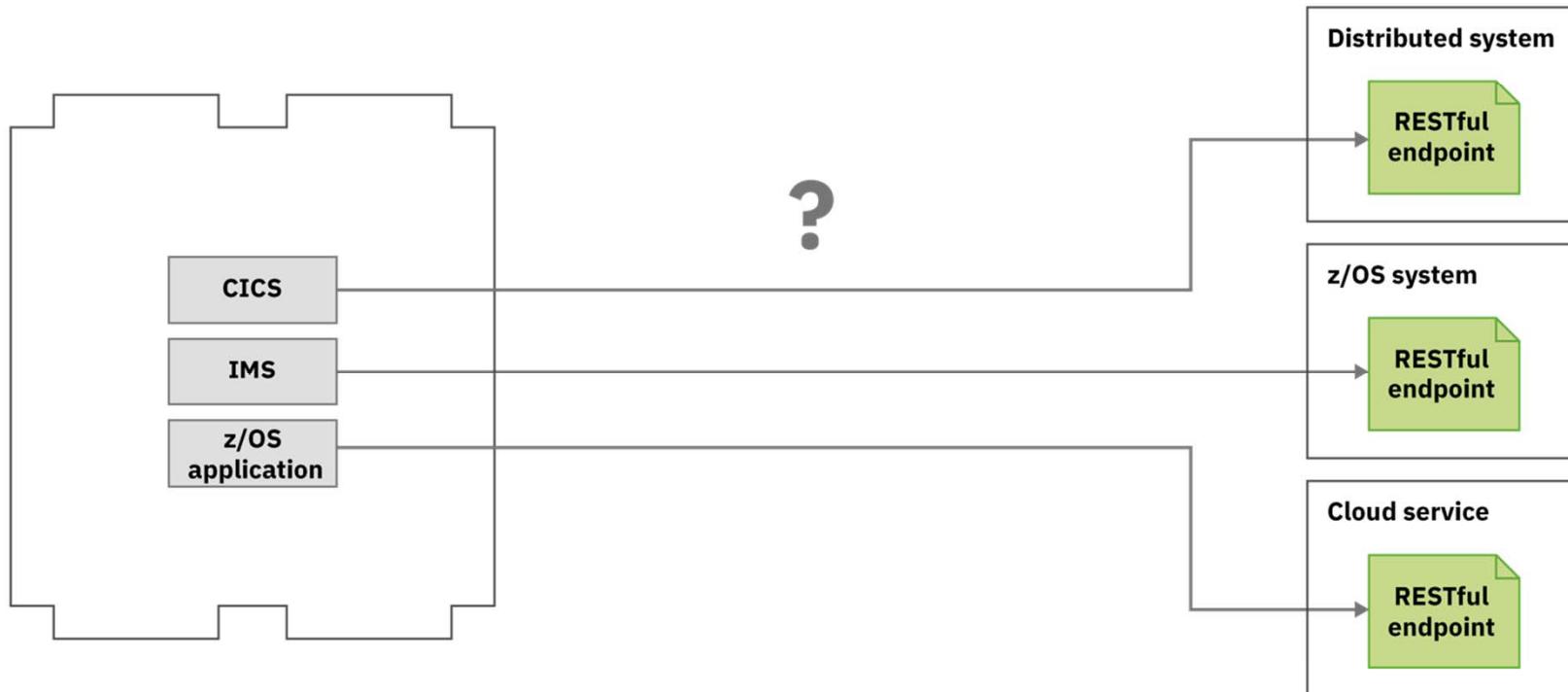
Test your deployed APIs directly with **Swagger UI** inside the editor.

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



z/OS Connect EE

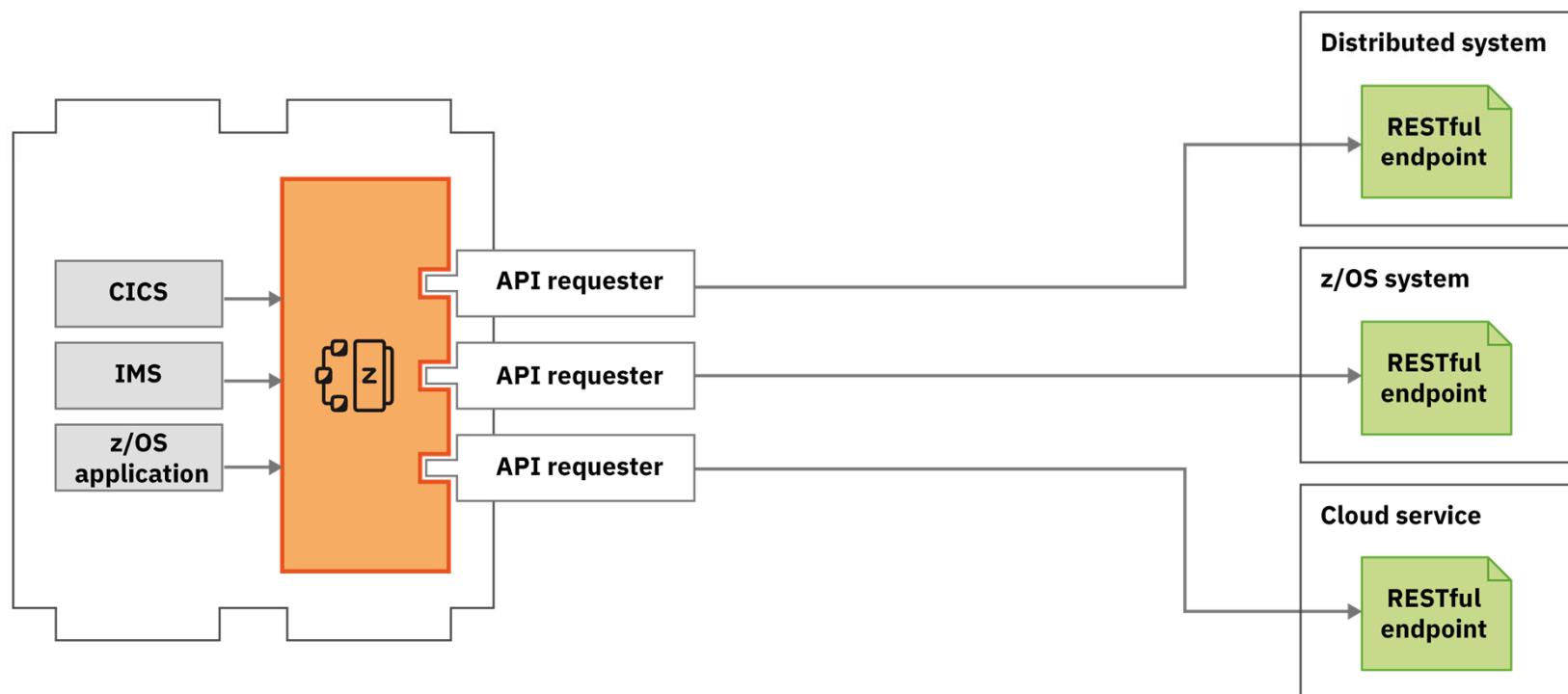
What about calling external APIs from my z/OS assets?





z/OS Connect EE

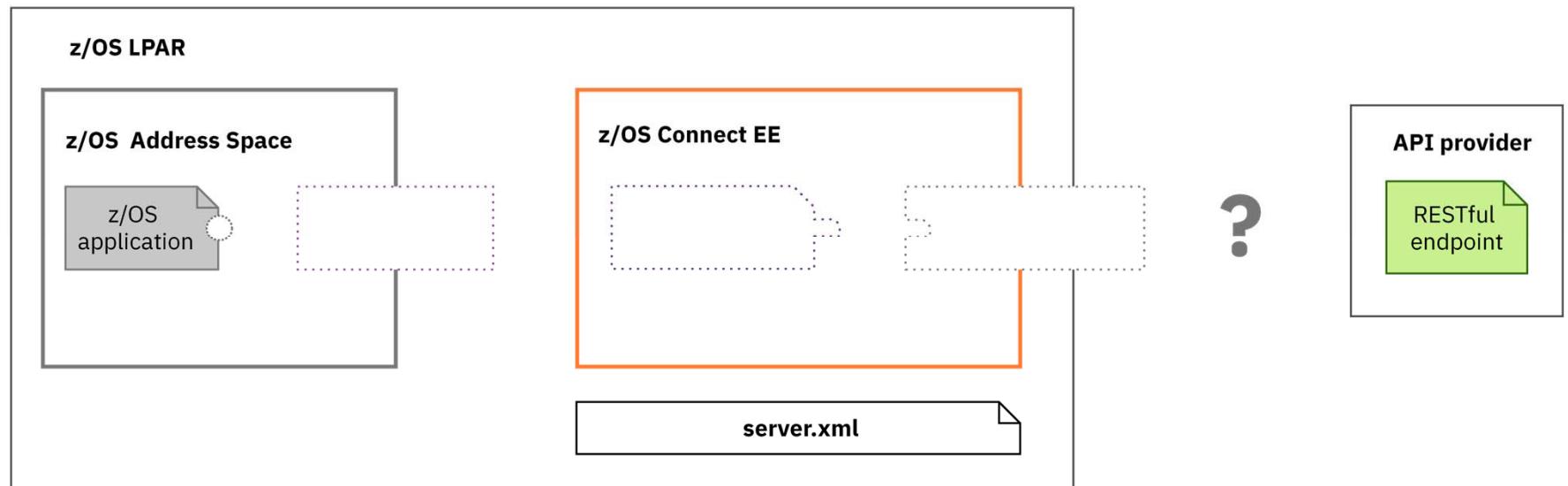
Use API requester to call external APIs from z/OS assets





Steps to calling an external API

Starting point

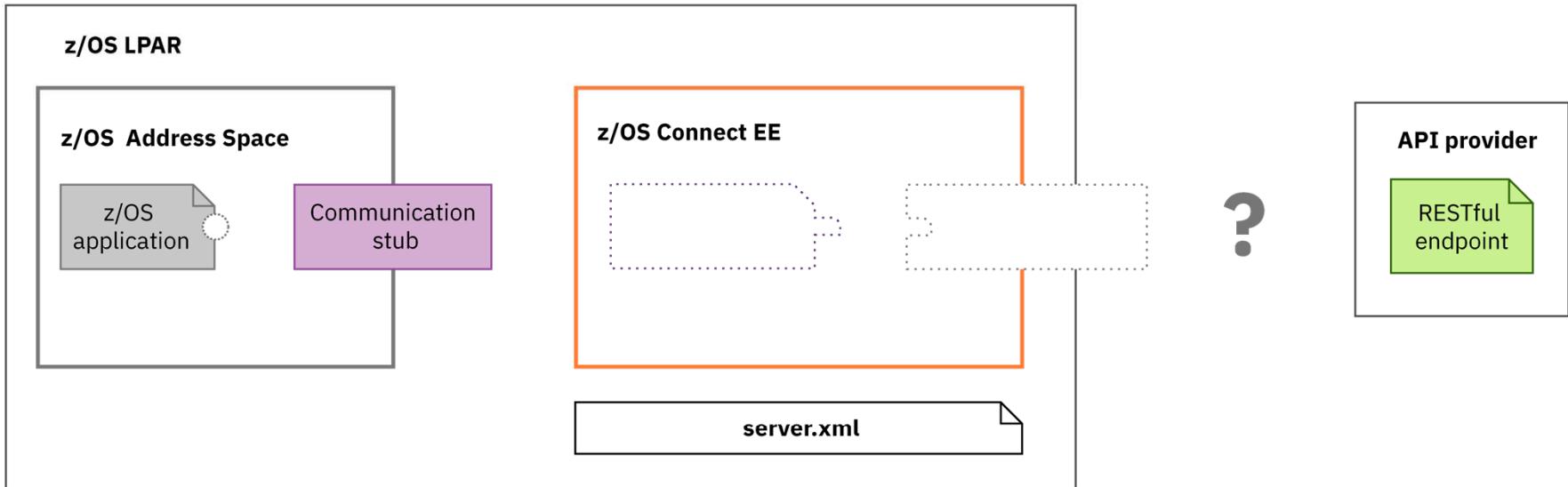




z/OS Connect EE

Steps to calling an external API

Step 1. Configure communication stub



Configure a communication stub. You only need to do this once per system.

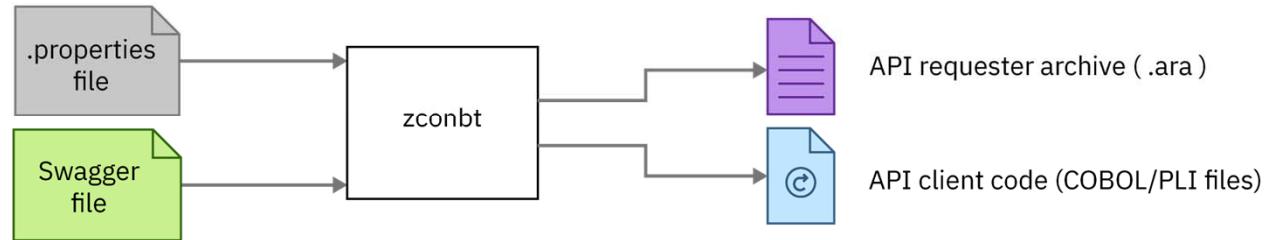
 ibm.biz/zosconnect-configure-comms-stub



z/OS Connect EE

Steps to calling an external API

Step 2. Generate API requester archive from Swagger

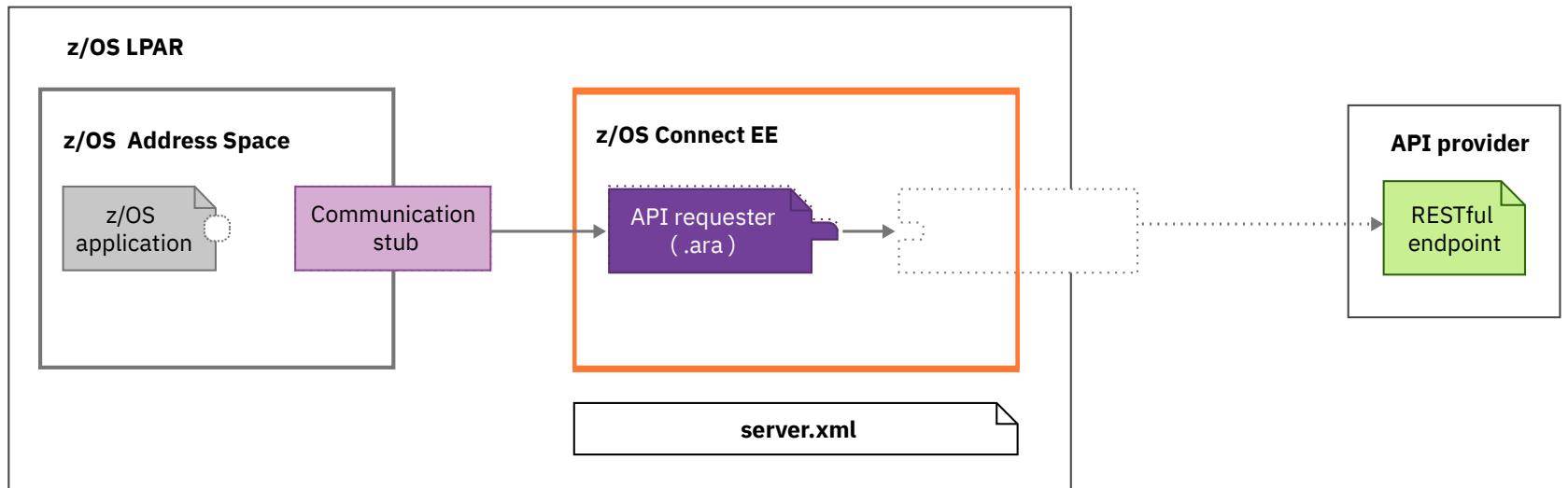


Generate your `.ara` file, and API client code.

 ibm.biz/zosconnect-generate-ara

Steps to calling an external API

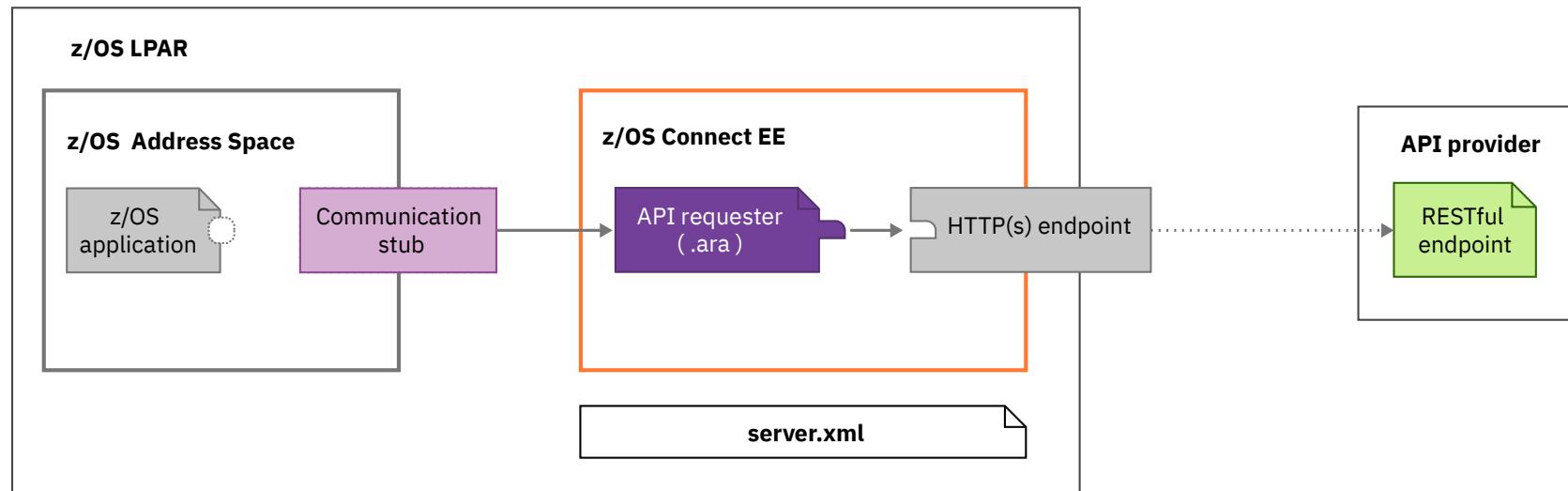
Step 3. Deploy API requester (.ara) archive



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

Steps to calling an external API

Step 4. Configure HTTP(S) endpoint



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

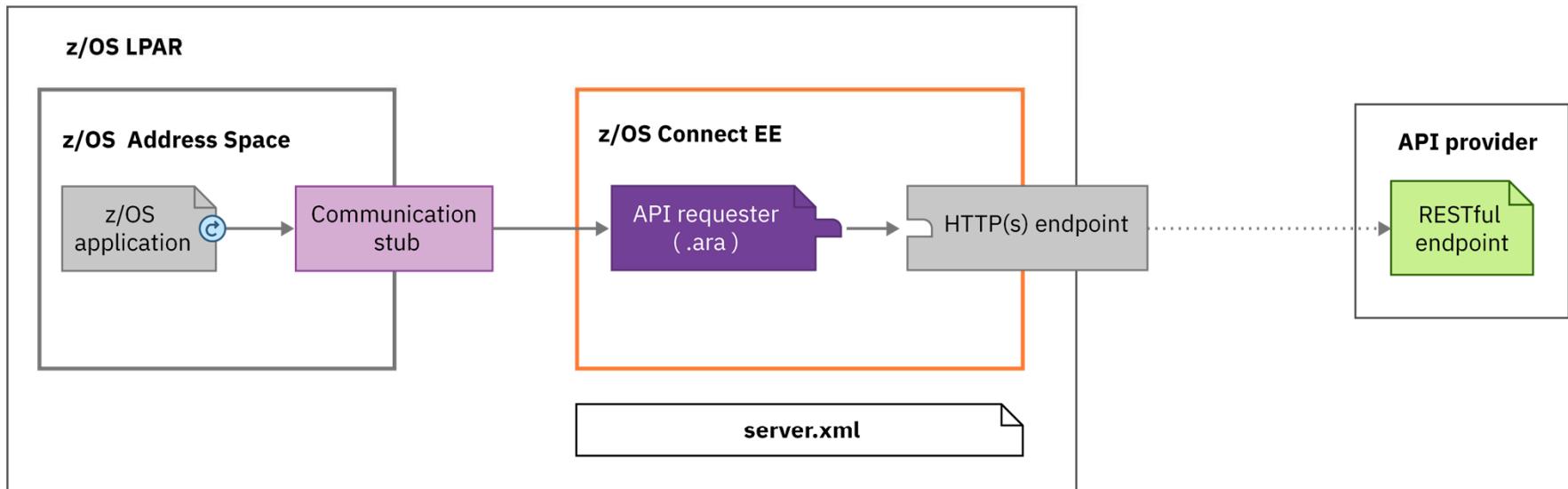
 ibm.biz/zosconnect-configure-endpoint-connection



z/OS Connect EE

Steps to calling an external API

Step 5. Update z/OS application



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

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

Steps to calling an external API

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

The screenshot shows the Rational Application Developer interface with three windows:

- GETAPI**: A COBOL editor window showing a copybook definition:

```
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.
```

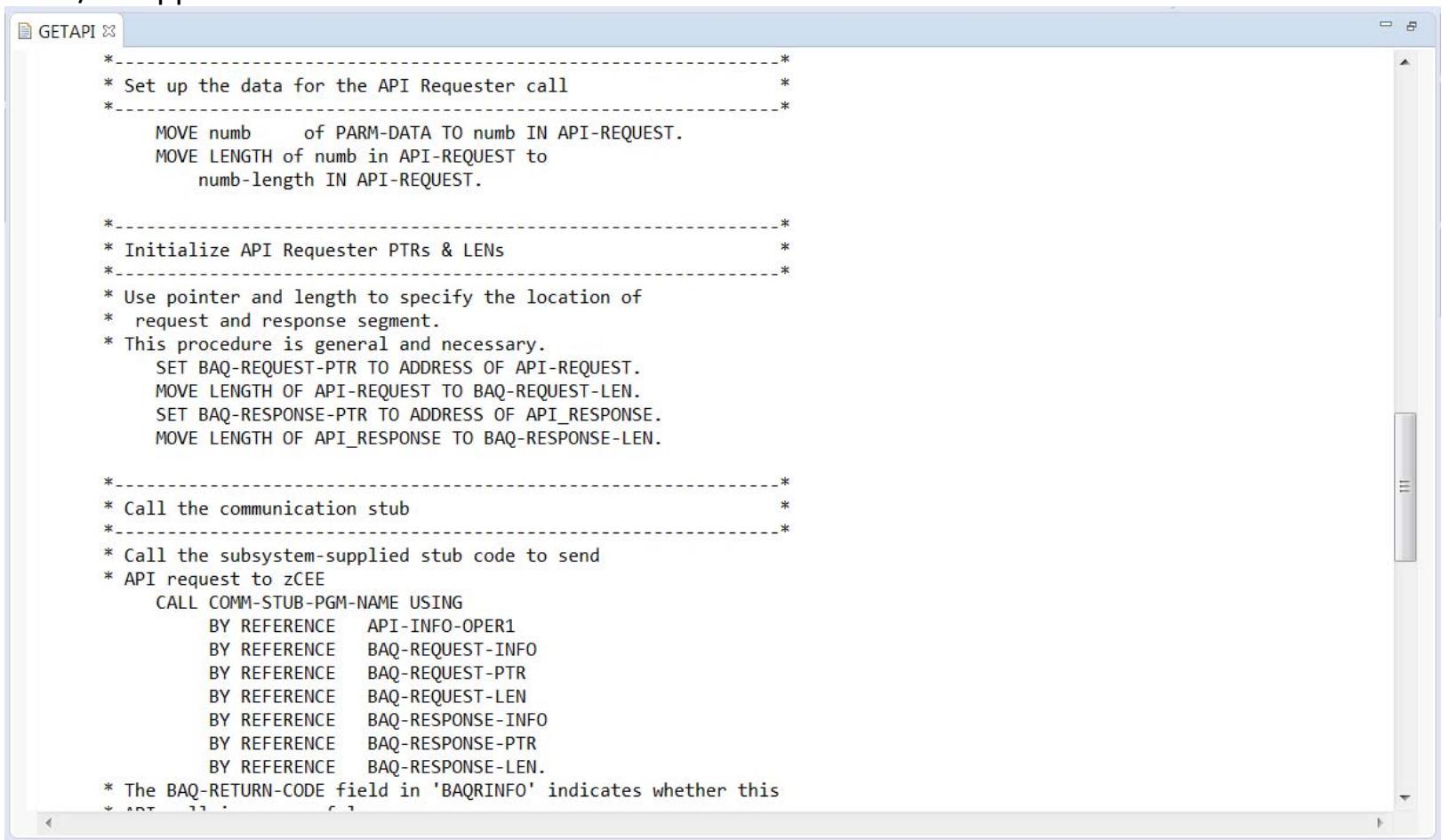
A red arrow points from the "CSC02I01" window to the "COPY CSC02I01." line in this window.
- apirs.xml**: An XML configuration file window showing the server configuration:

```
<server description="API Requester">  
    <!-- Enable features -->  
    <featureManager>  
        <feature>zosconnect:apiRequester-1.0</feature>  
    </featureManager>  
  
<zosconnect_apiRequesters location="">  
    <zosconnect_apiRequester name="cscvinc_1.0.0"/>  
</zosconnect_apiRequesters>  
  
<zosconnect_endpointConnection id="cscvincAPI">  
    host="http://wg31.washington.ibm.com"  
    port="9120"  
    basicAuthRef="myBasicAuth"  
    connectionTimeout="10s"  
    receiveTimeout="20s" />  
  
<zosconnect_authData id="myBasicAuth">  
    user="Fred"  
    password="fredpwd" />  
</server>
```
- CSC02I01**: A COBOL editor window showing the structure of the API request:

```
| 03 BAQ-APINAME          PIC 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.
```

Steps to calling an external API

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



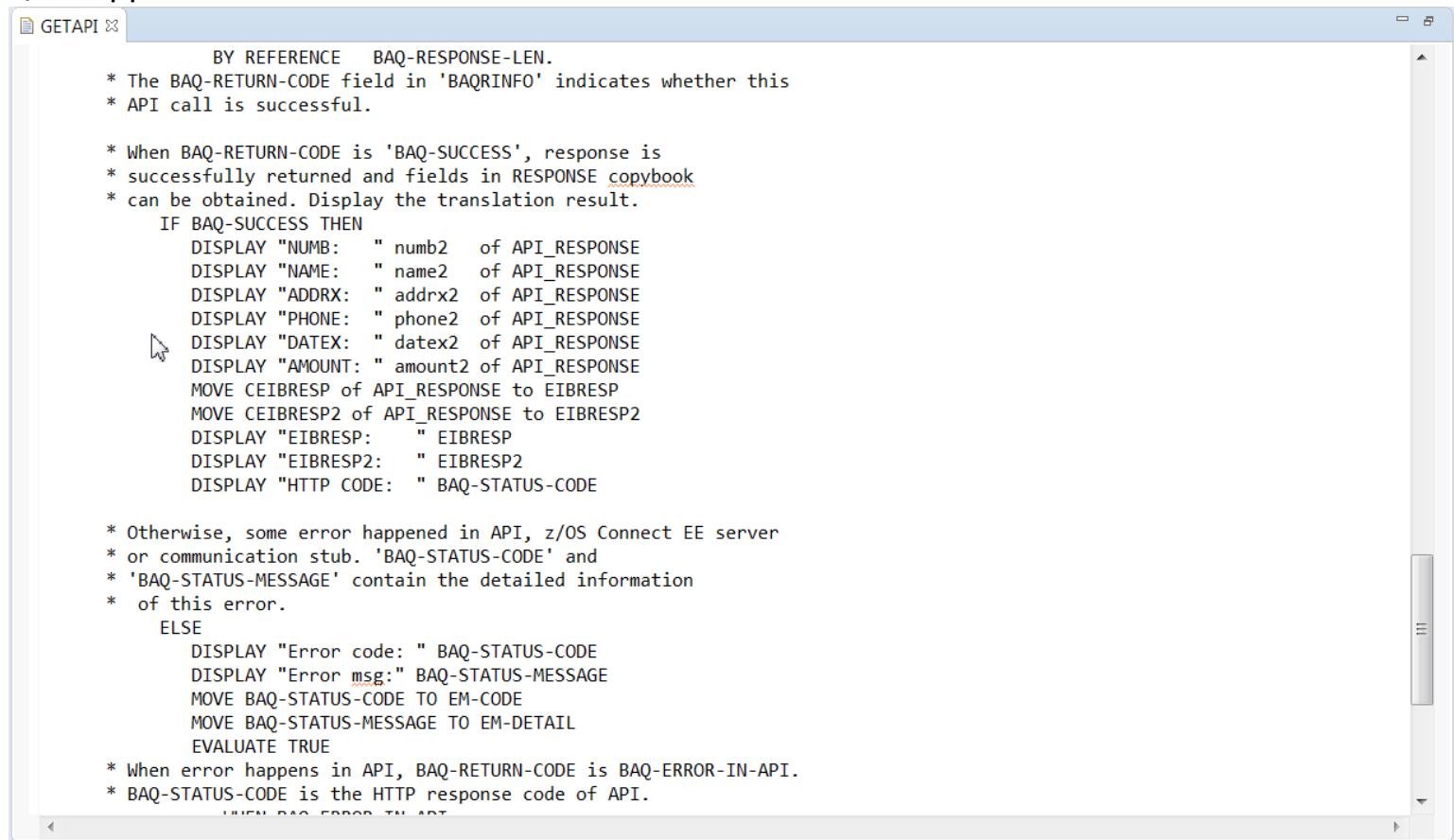
```
*-----*
* Set up the data for the API Requester call *
*-----*
      MOVE numb      of PARM-DATA TO numb IN API-REQUEST.
      MOVE LENGTH of numb in API-REQUEST to
            numb-length IN API-REQUEST.

*-----*
* 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.
```

Steps to calling an external API

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



The screenshot shows a window titled "GETAPI" containing AS/400 JCL (Job Control Language) code. The code is used to handle the response from an external API, specifically BAQ-RESPONSE-LEN. It includes logic to display various fields based on the success or failure of the API call, and to move data between EIBRESP and EIBRESP2 structures.

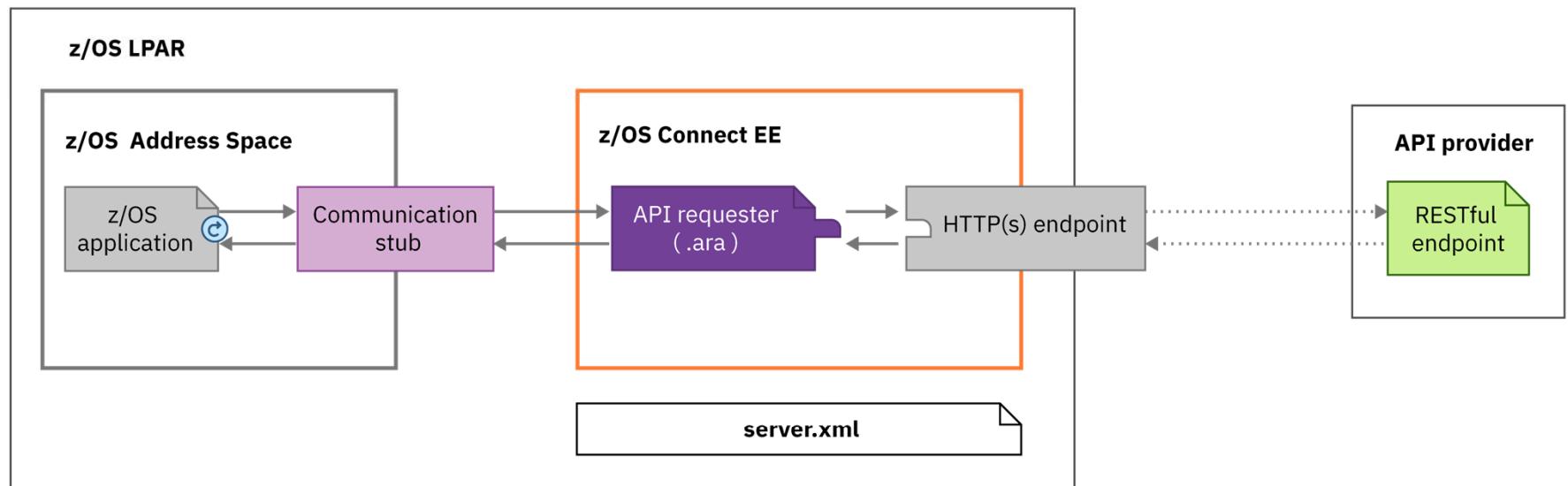
```
BY REFERENCE BAQ-RESPONSE-LEN.  
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this  
* API call is successful.  
  
* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is  
* successfully returned and fields in RESPONSE copybook  
* can be obtained. Display the translation result.  
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
```



z/OS Connect EE

Steps to calling an external API

Done

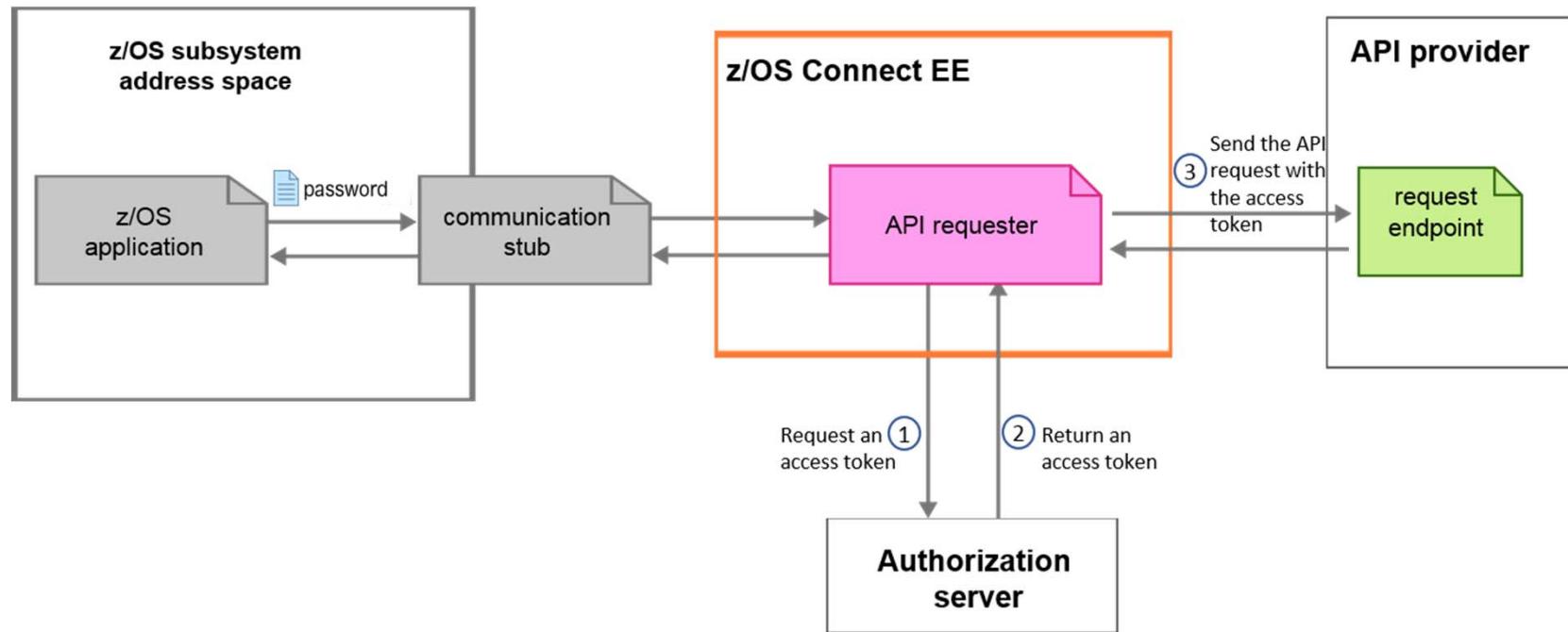




z/OS Connect EE

API endpoint secured by OAuth 2.0?

Not a problem



More information ibm.biz/zosconnect-oauth2



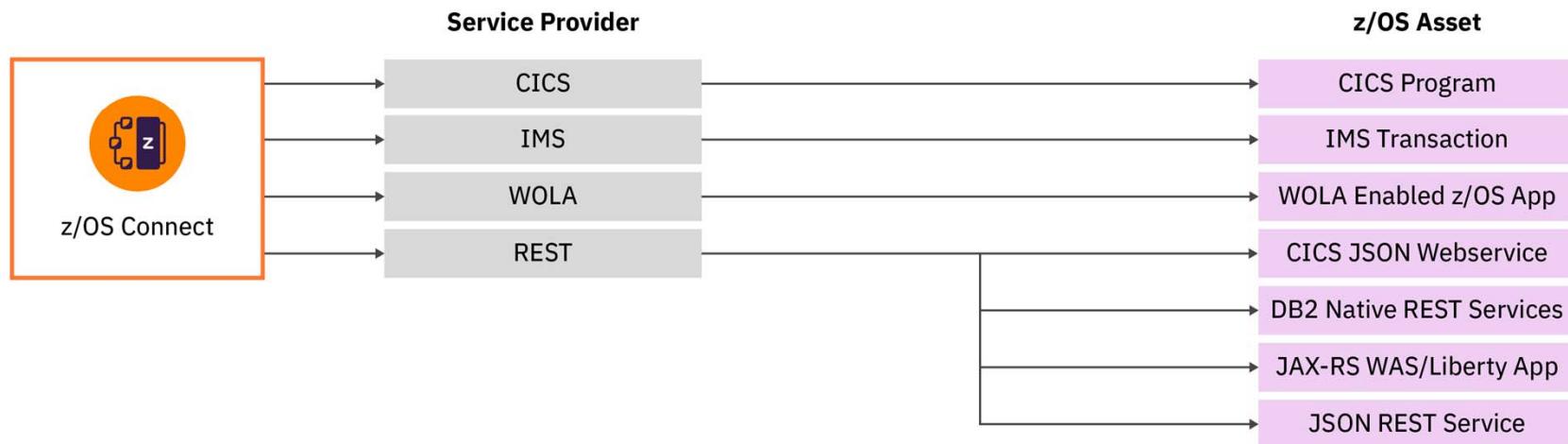
/common_scenarios

Typical connection patterns to different subsystems.

What assets can z/OS Connect EE map to?



And which service provider should I use?

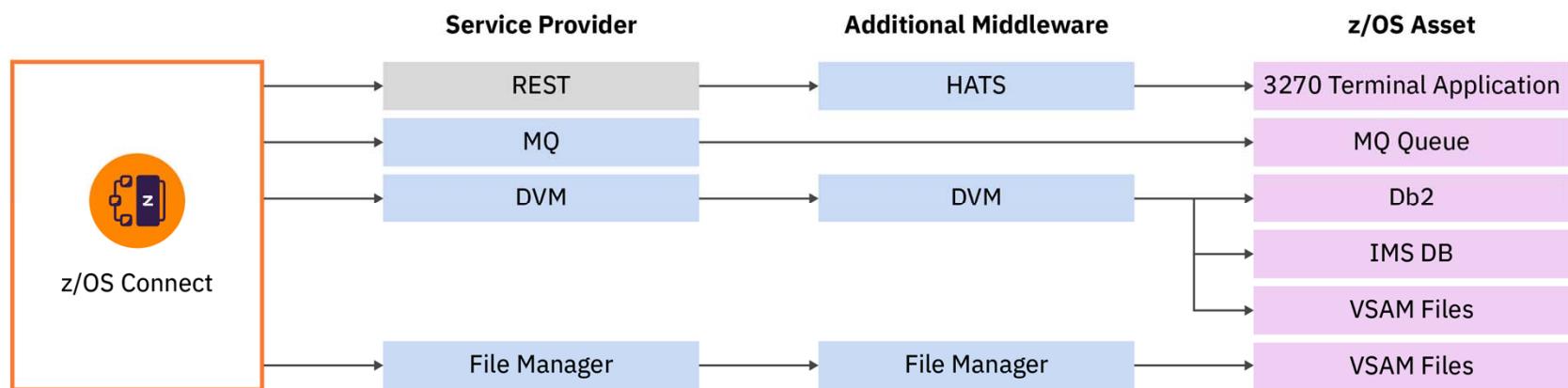


The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.

z/OS Connect EE 3rd party integrations



Additional value from the ecosystem



z/OS Connect EE is **pluggable** and **extensible** allowing 3rd Party Service Providers to expand the list of z/OS assets you can expose as APIs

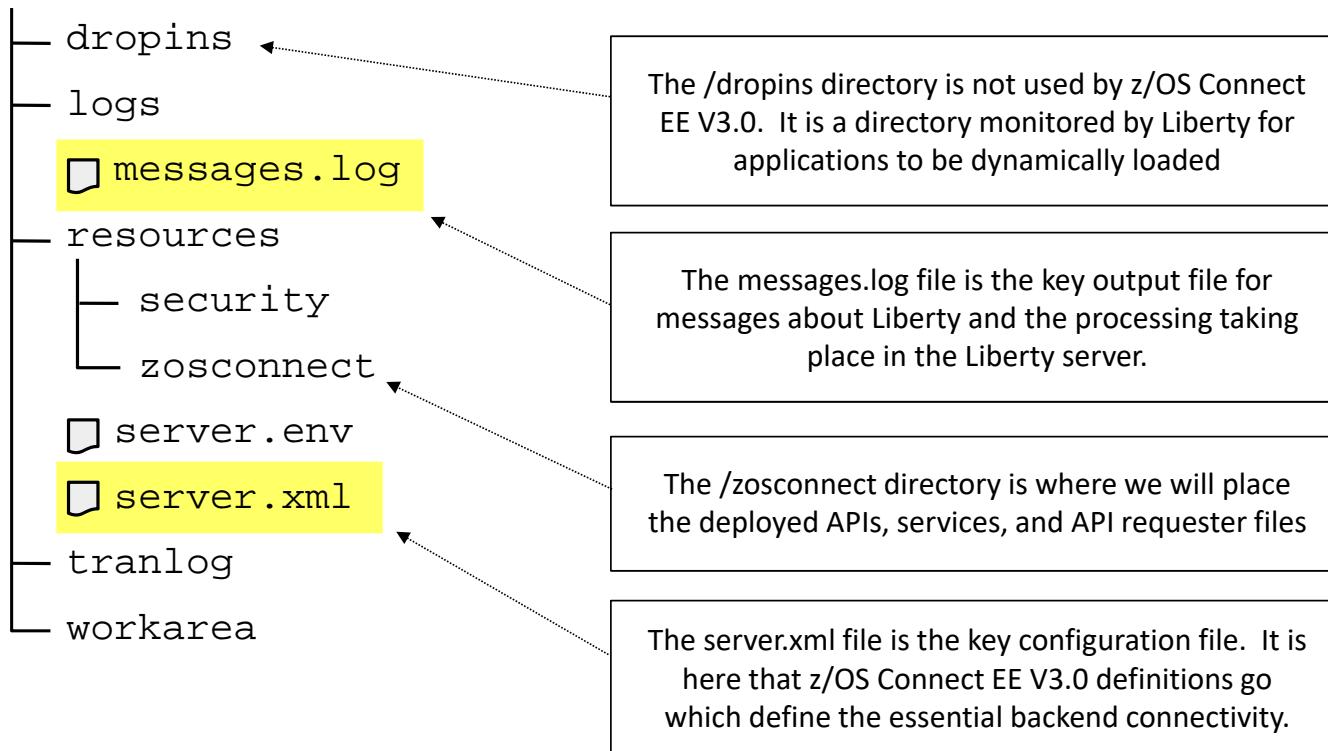
Tour of Server Configuration Directories and Files



z/OS Connect EE

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

/var/zosconnect/servers/<server_name>

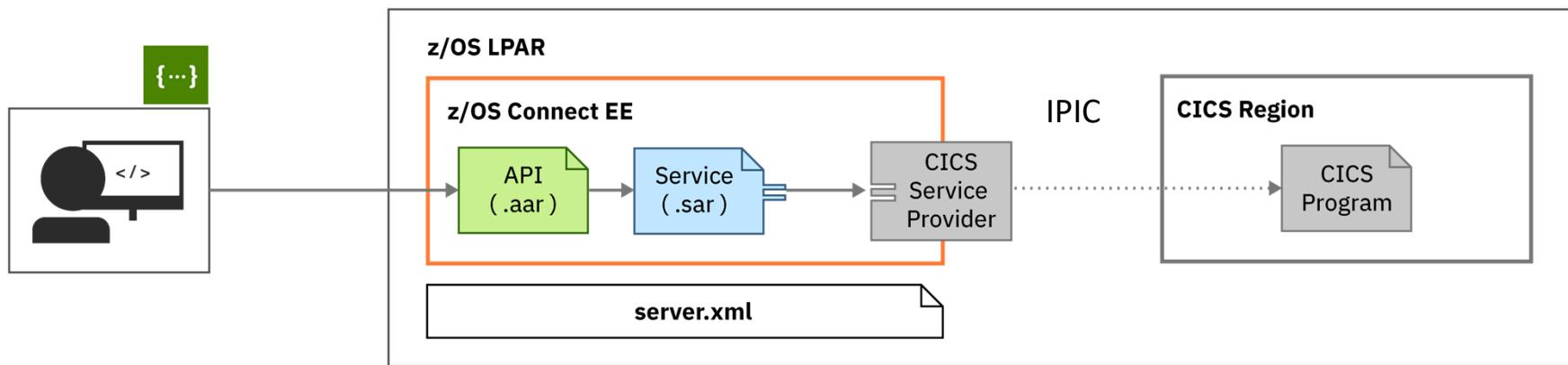


Connect to CICS



z/OS Connect EE

Topology



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

An IPIC connection must be configured in CICS.

ibm.biz/zosconnect-scenarios

The server.xml File (CICS IPIC)



z/OS Connect EE

The server.xml file is the key configuration file:

The diagram illustrates the configuration process for a CICS IPIC connection. It shows three main components:

- IBM Liberty Configuration interface:** A screenshot of the "inquireSingle Service" configuration window. It includes sections for "Required Configuration" (CCSID: 37, Connection reference: catalog) and "Optional Configuration" (Transaction ID, Transaction ID usage).
- catalog.xml:** A screenshot of the catalog.xml configuration page. It displays the XML code for defining a CICS IPIC connection:

```
<server description="CICS IPIC - catalog">
<!-- Enable features -->
<featureManager>
<feature>zosconnect:cicsService-1.0</feature>
</featureManager>
<zosconnect_cicsIpicConnection id="catalog"
host="wg31.washington.ibm.com"
port="1491"
transid="CSMI"
transidUsage="EIB_AND_MIRROR"/>
</server>
```
- server.xml:** A screenshot of the server.xml configuration page. It shows the "z/OS Connect CICS IPIC connection" section with the following fields:
 - ID:** catalog
 - Host:** wg31.washington.ibm.com
 - Port:** 1491
 - Shared port:** false (default)
 - z/OS Connect APPLID:** (no value)

Annotations with arrows point from the "catalog" field in the configuration interface to the "catalog" ID in the catalog.xml XML code, and from the "catalog" ID in the XML code to the "catalog" field in the server.xml configuration interface. Another annotation points from the "catalog" field in the configuration interface to the "catalog" ID in the XML code.

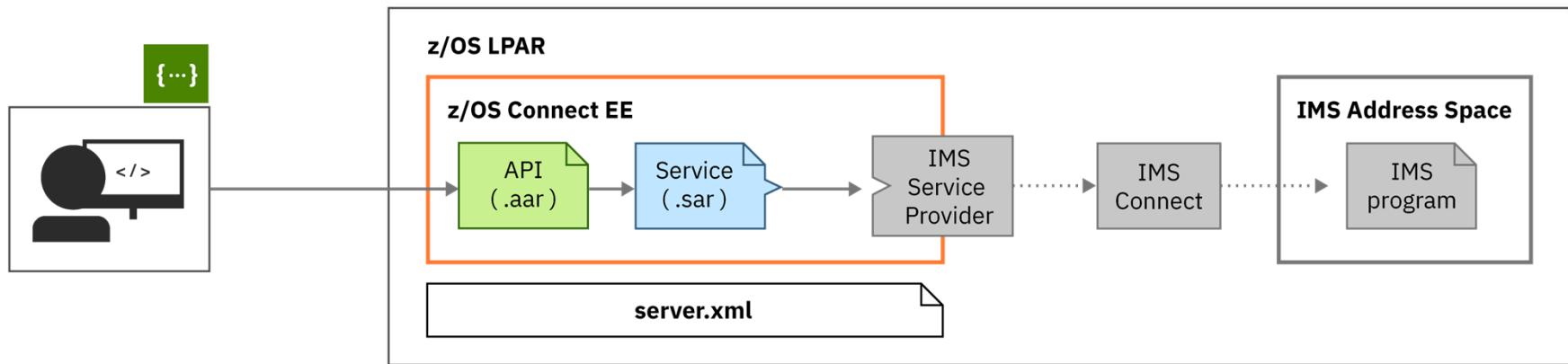
Features are functional building blocks. When configured here, that function becomes available to the Liberty server

Connect to IMS



z/OS Connect EE

Topology



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

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

ibm.biz/zosconnect-scenarios

© 2018, 2019 IBM Corporation

IMS Connections and Interactions



ivtnoService Service Configuration

Required Configuration

Enter the required configuration for this service.

Connection profile:

Interaction profile:

Optional Configuration

Enter the optional configuration for this service.

IMS destination override:

Program name:

Overview Configuration

Connection

```
<server>
<imsmobile_imsConnection comment="" connectionFactoryRef="IVP1" connectionTimeout="-1" connectionType="IMSCONNECT" id="IMSCONN"/>
<connectionFactory containerAuthDataRef="Connection1_Auth" id="IVP1">
    <properties.gmoa hostName="wg31.washington.ibm.com" portNumber="4000"/>
</connectionFactory>

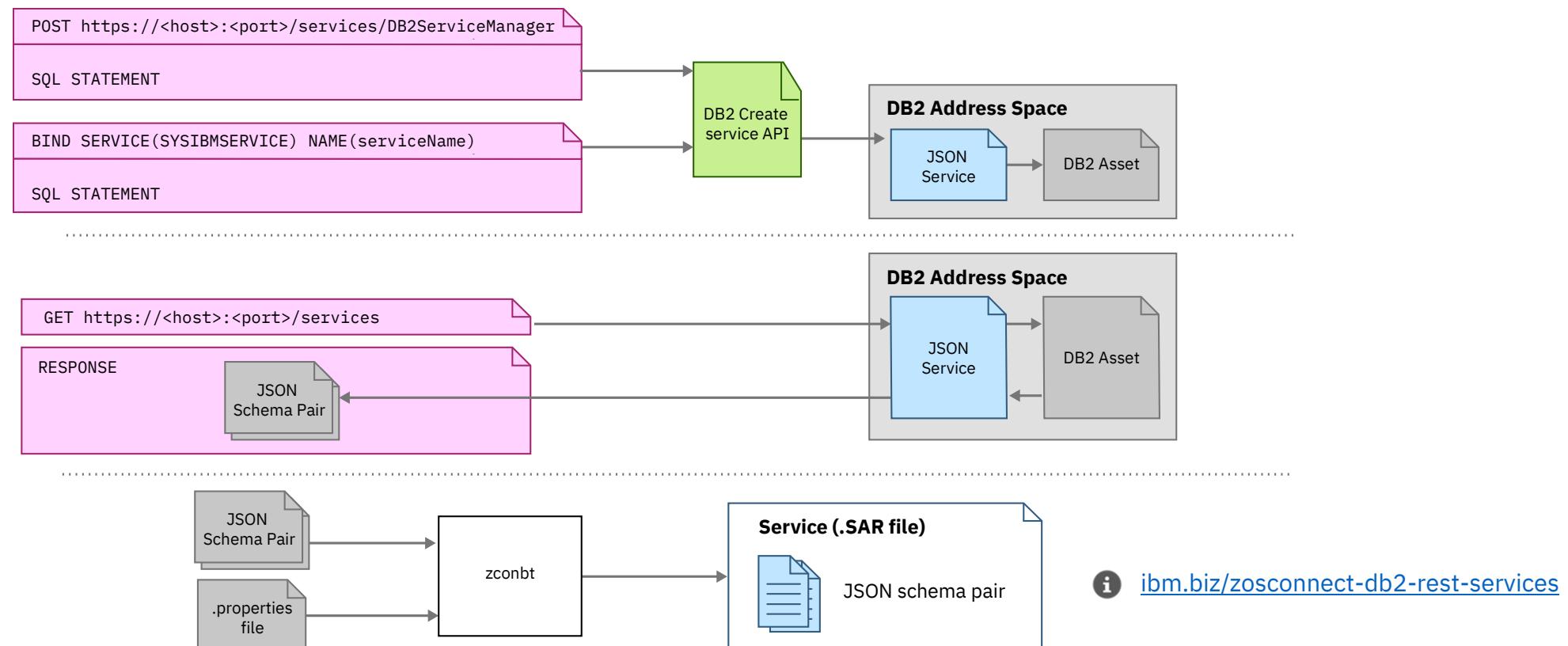
<authData id="Connection1_Auth" password="encryptedPassword1" user="userName1"/>
</server>
```

Interaction

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

Connect to Db2

Create the service definition



.sar file is created from the JSON schema of the DB2 service using the **zconbt utility**.

Deploying Db2 Service Archive Options



z/OS Connect EE

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

```
1  {
2    "zosConnect": {
3      "serviceName": "selectEmployee",
4      "serviceDescription": "Select a row from USER1.EMPLOYEE",
5      "serviceProvider": "IBM_ZOS_CONNECT_SERVICE_REST_CLIENT-1.0",
6      "serviceURL": "https://wg31.washington.ibm.com:9453/zosConnect/services/selectEmployee",
7      "serviceInvokeURL": "https://wg31.washington.ibm.com:9453/zosConnect/services/selectEmployee?action=invoke",
8      "dataXformProvider": "DATA_UNAVAILABLE",
9      "serviceStatus": "Started"
10 },
11   "selectEmployee": {
12     "receiveTimeout": 0,
13     "port": null,
14     "host": null,
15     "httpMethod": "POST",
16     "connectionTimeout": 0,
17     "uri": "/services/selectEmployee"
18 }
```

Command:

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

Results:

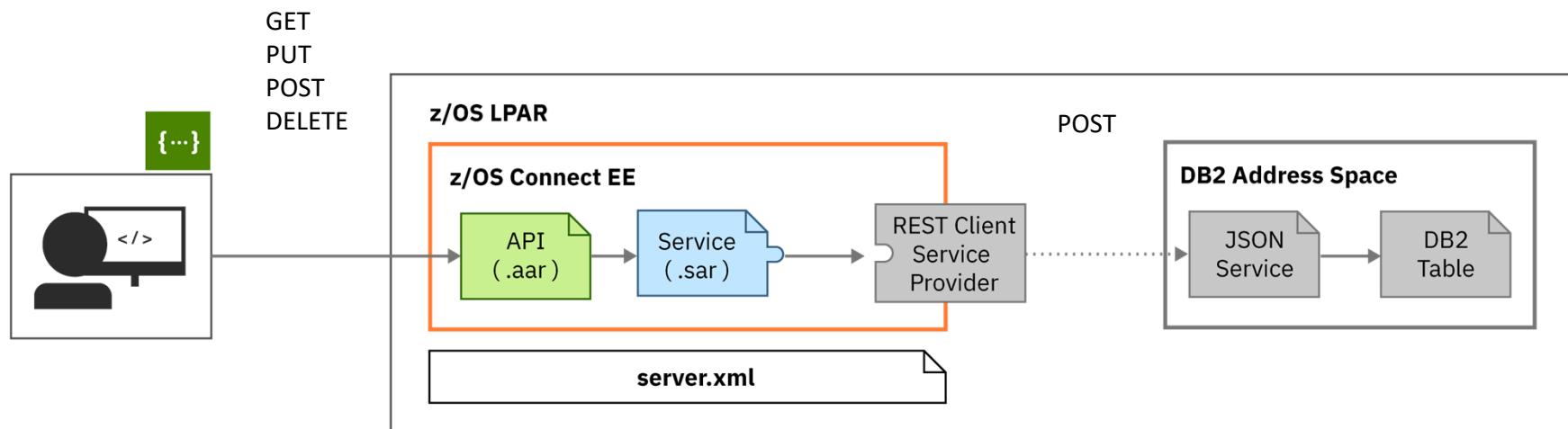
```
{"zosConnect":{"serviceName":"selectEmployee","serviceDescription":"Select a row from USER1.EMPLOYEE","serviceProvider":"IBM_ZOS_CONNECT_SERVICE_REST_CLIENT-1.0","serviceURL":"https://mpxm:9453/zosConnect/services/selectEmployee","serviceInvokeURL":"https://mpxm:9453/zosConnect/services/selectEmployee?action=invoke","dataXformProvider":"DATA_UNAVAILABLE","se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

Connect to Db2



z/OS Connect EE

Topology



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

A JSON Service must be configured in DB2.



ibm.biz/zosconnect-db2-rest-services

The server.xml File (Db2)



z/OS Connect EE

The server.xml file is the key configuration file:

db2pass.xml

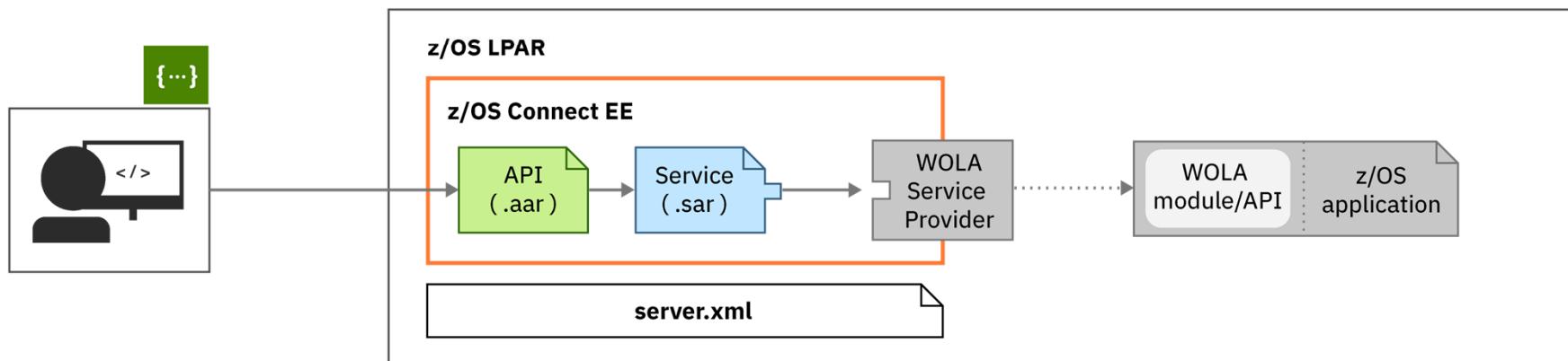
Design	Source
1	<server description="DB2 REST">
2	
3	<zosconnect_zosConnectServiceRestClientConnection id="db2conn"
4	host="wg31.washington.ibm.com"
5	port="2446"
6	basicAuthRef="dsn2Auth" />
7	
8	<zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
9	appName="DSN2APPL"/>
10	
11	</server>
12	

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

Connect to a MVS batch application



Topology



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

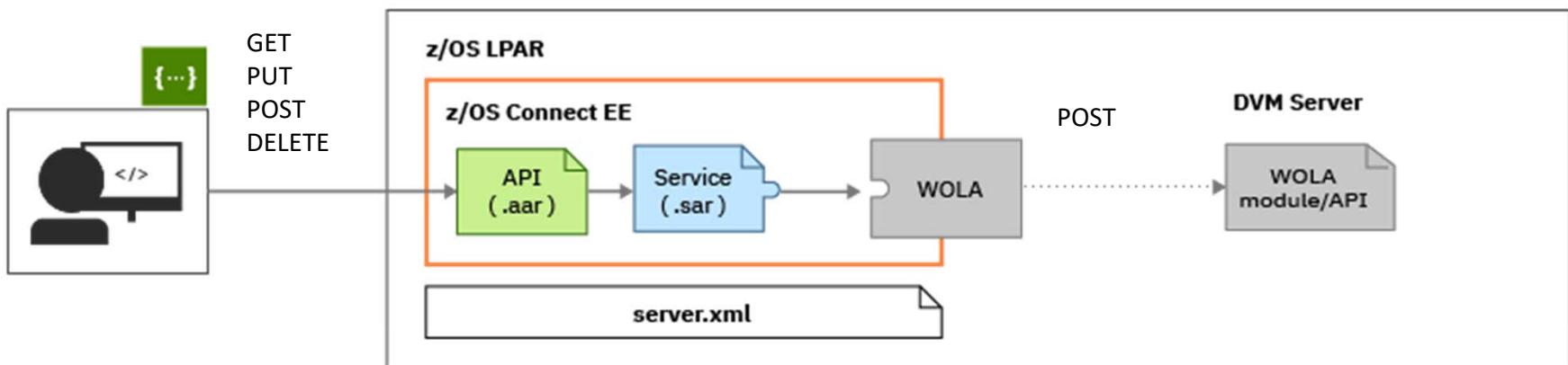
The z/OS application must be WOLA-enabled.

Connect to DVM



z/OS Connect EE

Topology



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

A REST service must be configured in the HATS.

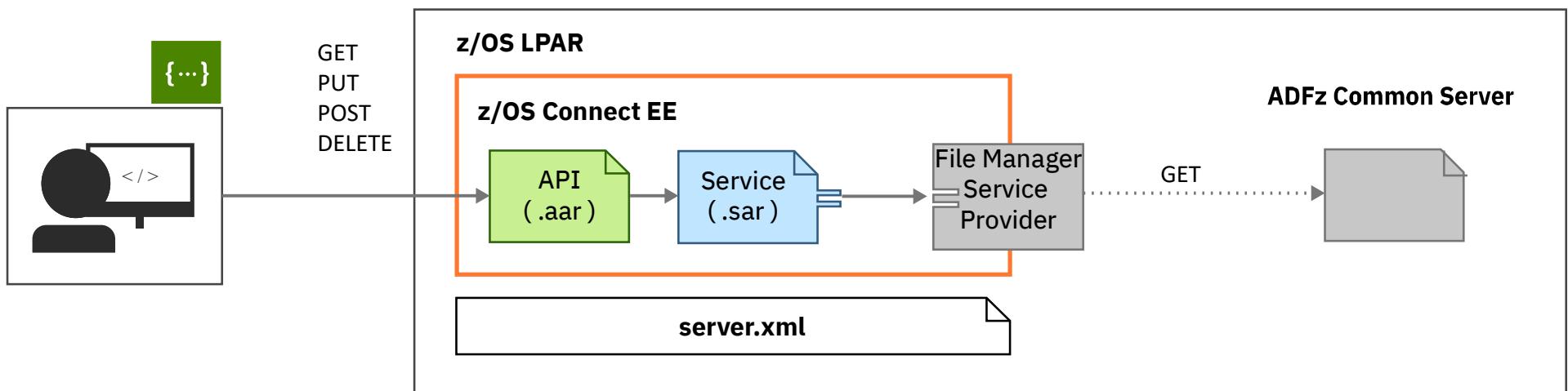
ibm.biz/zosconnect-db2-rest-services

Connect to File Manager



z/OS Connect EE

Topology



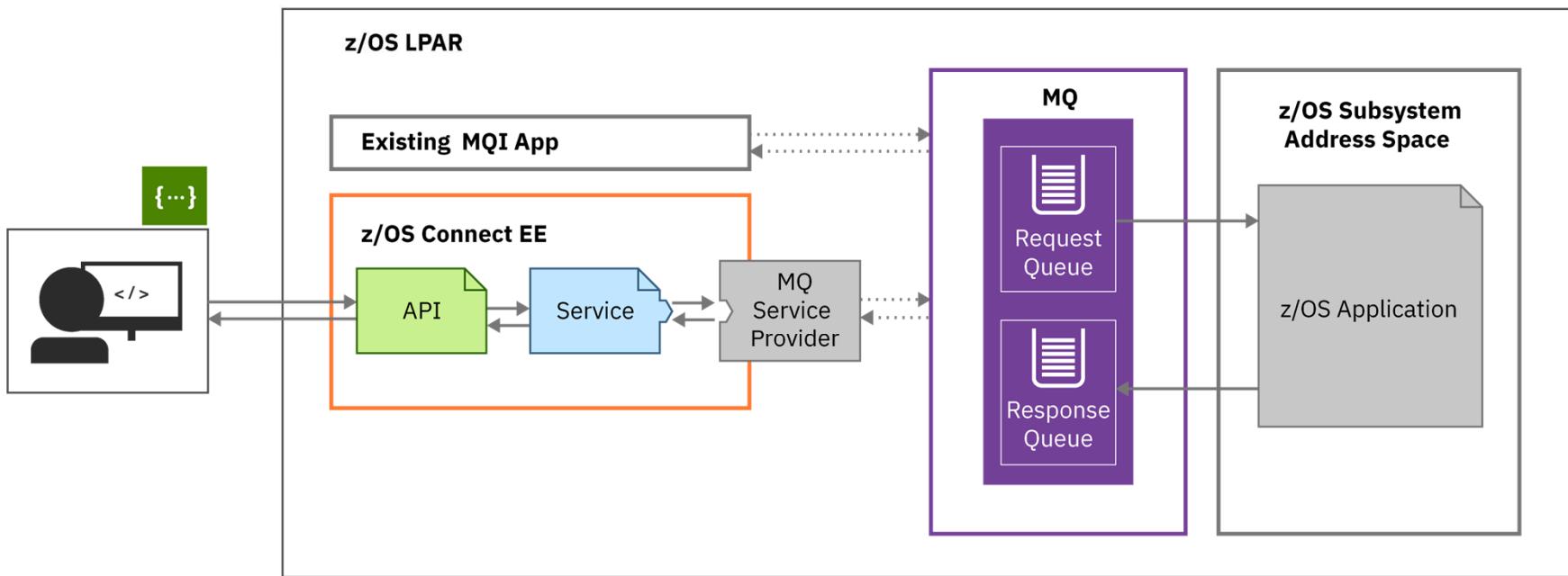
Connection to the Application Delivery Foundation for z (ADFz) common server is over TCP/IP

A File Manager Template is required .

Connect to MQ



Topology (Two-way service example)



You can also configure one-way services.

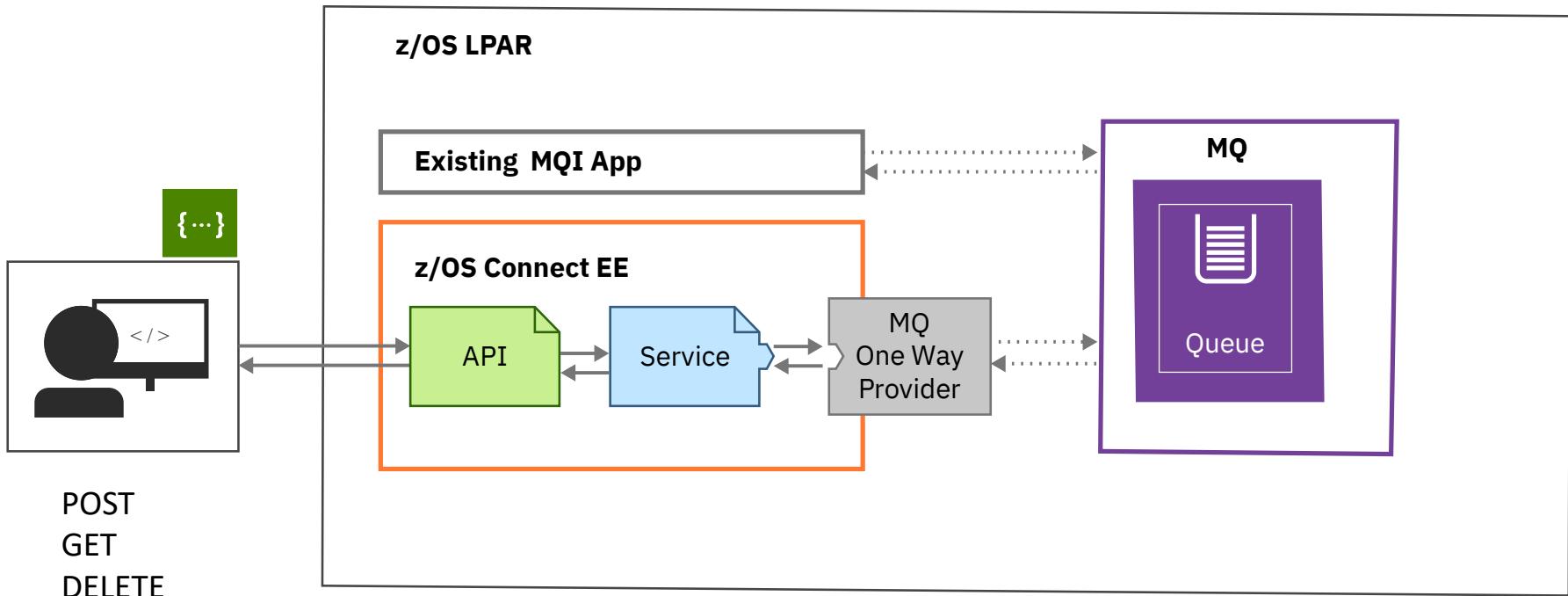
ⓘ ibm.biz/zosconnect-mq-service-provider

Connect to MQ



z/OS Connect EE

Topology (One-way service example)



ibm.biz/zosconnect-mq-service-provider

© 2018, 2019 IBM Corporation



z/OS Connect EE

The server.xml File (MQ)

mq.xml

Design Source

Server

Feature Manager

- Feature jms-2.0
- Feature mqzosconnect:zosConnectMQ-2.0
- Feature wmqJmsClient-2.0
- Feature zosTransaction-1.0

Variable Declaration wmqJmsClient.rar.location

WebSphere MQ Messaging

z/OS Connect Endpoint filequeue

z/OS Connect Endpoint miniloan

IBM MQ for z/OS service provider for IBM MQ

IBM MQ for z/OS service provider for IBM MQ

Connection Manager ConMgr1

JMS Connection Factory qmgrCf

JMS Queue q1

JMS Connection Factory
Defines a JMS connection factory configuration.

Add child Remove

ID
qmgrCf
A unique configuration ID.

Connection manager reference
ConMgr1
Connection manager for a connection factory.

Container managed authentication data reference
(no value)
Default authentication data for container managed authentication that applies when bindings do not specify an authentication-alias for a resource reference with res-auth=CONTAINER.

JNDI name
jms/qmgrCf
JNDI name for a resource.

Features related to JMS Support

JMS Connection Factories,

MQ V9.1.1 Added support for remote queue managers.

The server.xml File (one-way MQ service)



z/OS Connect EE

```
12 <wmqJmsClient nativeLibraryPath= /usr/rpp/mqm/v9R1MI/Java/110 />
13
14 <zosconnect_zosConnectService id="fileaqueue"
15   invokeURI="/FileaQueue"
16   dataXformRef="xformJSON2Byte"
17   serviceName="FileaQueue"
18   serviceDescription="MQ Oneway Service"
19   serviceRef="FileaQueue" />
20
21 <mqzosconnect_mqzOSConnectService id="FileaQueue"
22   connectionFactory="jms/qmgrCF"
23   destination="jms/default" />
24
25 <jmsQueue id="q1" jndiName="jms/default">
26   <properties.wmqJms
27     baseQueueName="ZCONN2.DEFAULT.MQZCEE.QUEUE"
28     CCSID="37"/>
29 </jmsQueue>
30
```

z/OS Connect service

MQ z/OS Connect service

JMS Destination (queue)

The server.xml File (two-way MQ service)



z/OS Connect EE

The diagram illustrates the architecture of a two-way MQ service. It shows three main components: a **z/OS Connect service**, an **MQ z/OS Connect service**, and **JMS Destinations (queue)**. The **z/OS Connect service** is connected to the **MQ z/OS Connect service**. The **MQ z/OS Connect service** is connected to the **JMS Destinations (queue)**. Arrows indicate the flow of data between these components. The **MQ z/OS Connect service** is highlighted in a light blue box.

```
<zosconnect_zosConnectService id="miniloan">
    invokeURI="/Miniloan"
    dataXformRef="xformJSON2Byte"
    serviceName="Miniloan"
    serviceDescription="MQ Reply/Response Service"
    serviceRef="Miniloan" />

<mqzosconnect_mqzOSConnectService id="Miniloan">
    connectionFactory="jms/qmgrCF"
    waitInterval="30000"
    destination="jms/request"
    replyDestination="jms/response"/>

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

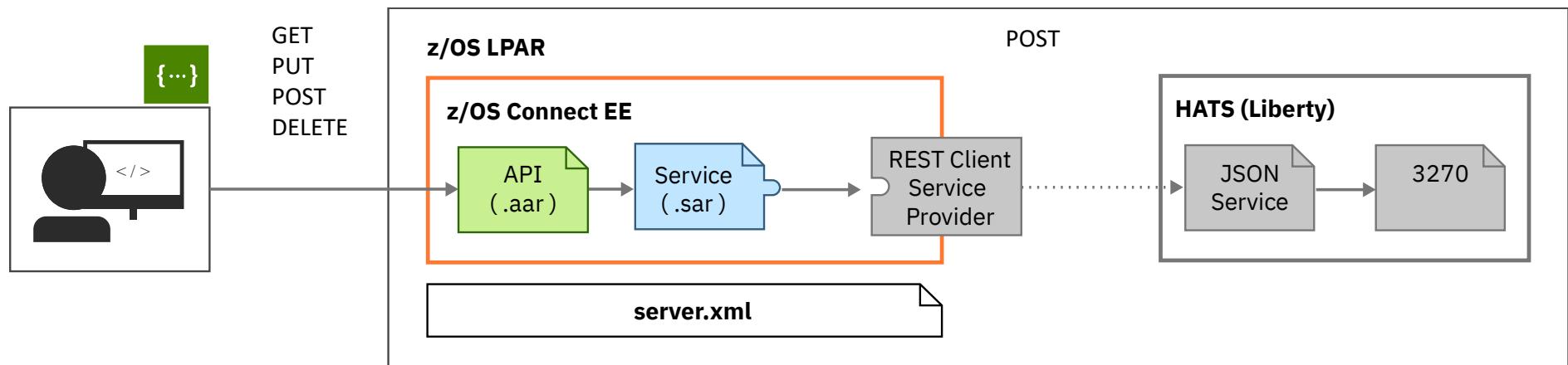
<jmsQueue id="response" jndiName="jms/response">
    <properties.wmqJms>
        baseQueueName="ZCONN2.TRIGGER.RESPONSE"
        targetClient="MQ"
        CCSID="37"/>
    </properties.wmqJms>
</jmsQueue>
```

Connect to HATS



z/OS Connect EE

Topology



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

A REST service must be configured in the HATS.

ibm.biz/zosconnect-db2-rest-services



/miscellaneousTopics

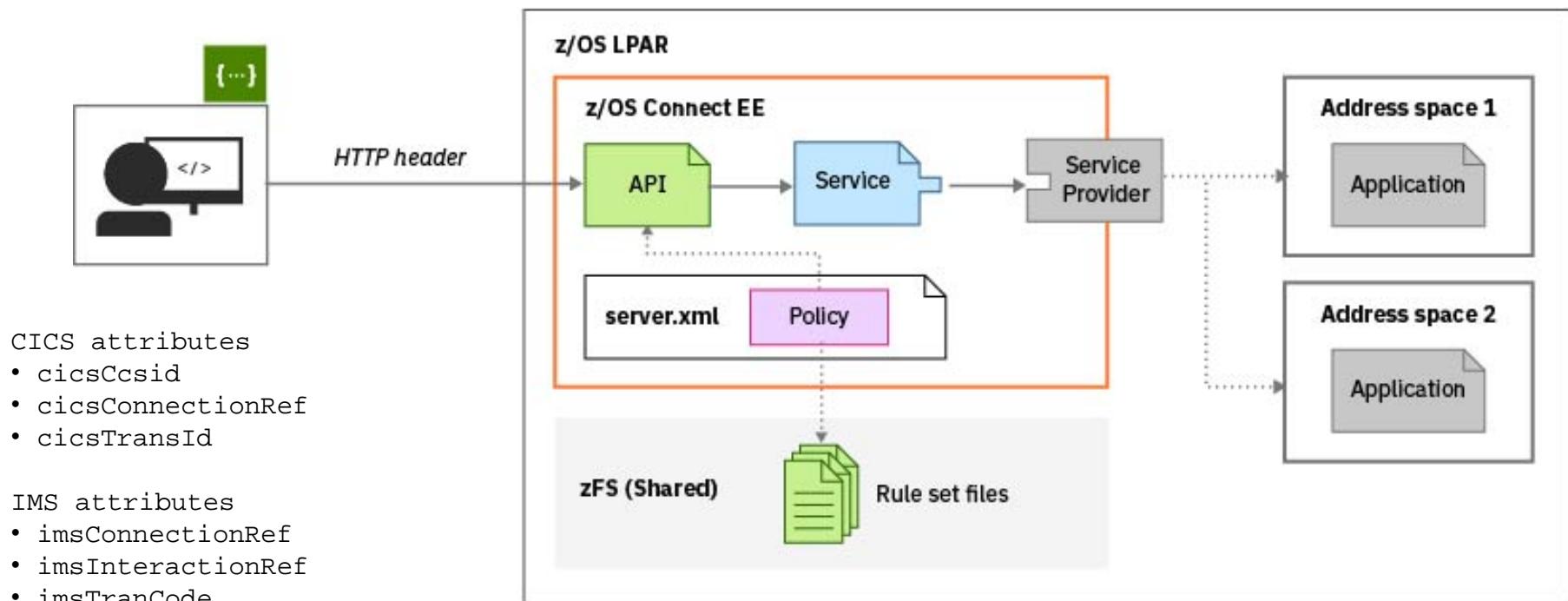
performance, high availability, Liberty



z/OS Connect EE

API Policies

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



Liberty's “adminCenter” Feature



z/OS Connect EE

Web browser interface to the server's configuration files

The screenshot shows the "Server Config" interface for the "server.xml" file. The left sidebar lists various configuration sections under "Remote File Access". The "z/OS Connect Manager" section is currently selected. In the main pane, there are four configuration groups:

- Global asynchronous request timeout:** Set to "(no value)".

Specifies the mode in which all z/OS Connect requests are processed.
- Global administrative group:** Set to "GADMIN".

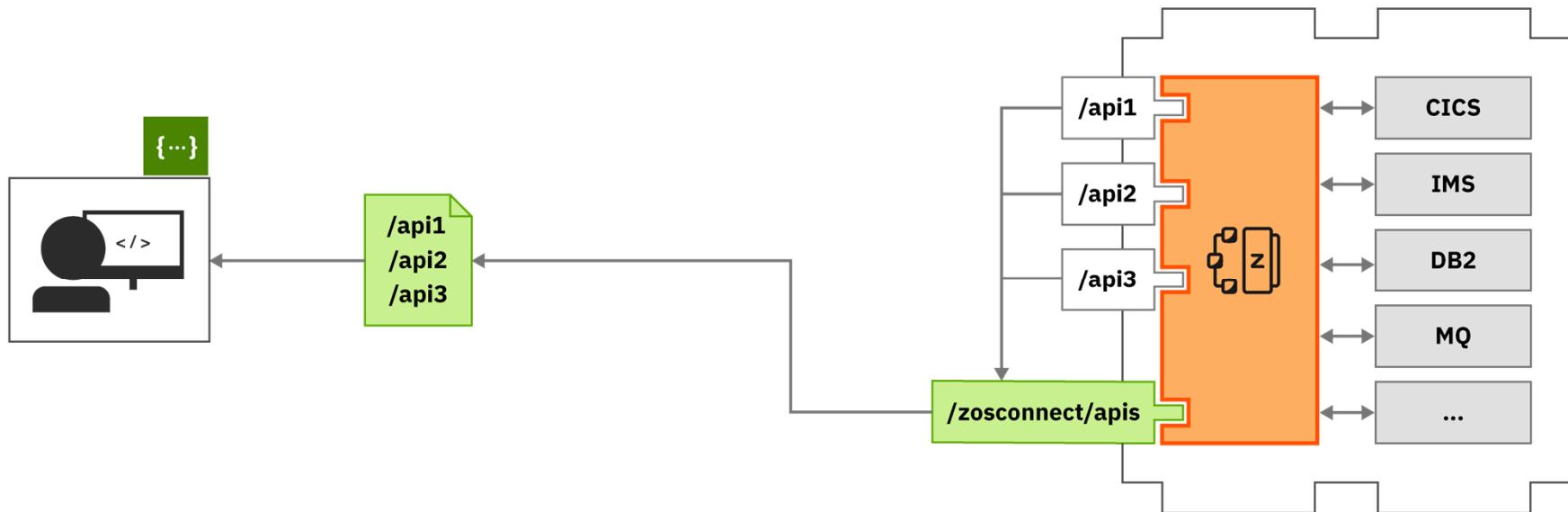
Administrative group name that is associated with all service endpoints. It is the name of the security group that the user needs to be in before administrative functions are permitted.
- Global operations group:** Set to "GOPER".

Operations group name that is associated with all service endpoints. It is the name of the security group that the user needs to be in before operations such as start, stop, or status is permitted.
- Global invoke group:** Set to "GINVOKE".

Invoke group name that is associated with all service endpoints. It contains the name of the security group that the user needs to be in before invoke calls are permitted.

Buttons at the top right include "Save" and "Close".

API Documentation



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



z/OS Connect EE

RESTful Administrative Interface for Services

The administration interface for services is available in paths under /zosConnect/services.

Most administration tasks are supported by the RESTful administration interface

Method	Administrative Task
GET	Get details of a service
	Get the status of a service
	Get the request schema of a service
	Get the response schema of a service
POST	Deploy a service*
PUT	Update a service
	Change the status of a service
DELETE	Delete a service

POST /zosConnect/services inquireSingle.sar

PUT /zosConnect/services/{serviceName}?status=started|stopped

*Useful for deploying DB2 and HATS
service archive files

PUT /zosConnect/services inquireSingle.sar

GET /zosConnect/services

GET /zosConnect/services/{serviceName}

DELETE /zosConnect/services/{serviceName}

© 2018, 2019 IBM Corporation



z/OS Connect EE

RESTful Administrative Interface for APIs

The administration interface for services is available in paths under /zosConnect/apis.

Most administration tasks are supported by the RESTful administration interface

Method	Administrative Task
GET	Get a list of APIs
	Get the details of an API
POST	Deploy an API
PUT	Update an API
	Change the status of an API
DELETE	Delete an API

```
POST  /zosConnect/apis CatalogManager.aar
PUT   /zosConnect/apis/{apiName}?status=started|stopped
PUT   /zosConnect/apis CatalogManager.aar
GET   /zosConnect/apis
GET   /zosConnect/apis/{apiName}
DELETE /zosConnect/apis/{apiName}
```

© 2018, 2019 IBM Corporation

RESTful Administrative Interface for API Requesters



z/OS Connect EE

The administration interface for services is available in paths under `/zosConnect/apisRequesters`.

Most administration tasks are supported by the RESTful administration interface

Method	Administrative Task
GET	Get a list of API Requesters
	Get the details of an API Requester
POST	Deploy an API Requester
PUT	Update an API Requester
	Change the status of an API Requester
DELETE	Delete an API Requester

```
GET /zosConnect/apiRequesters cscvinc.aar
PUT /zosConnect/apiRequesters/{apiRequesterName}?status=started|stopped
PUT /zosConnect/apiRequesters cscvinc.aar
GET /zosConnect/apiRequesters
GET /zosConnect/apiRequesters/{apRequesterName}
DELETE /zosConnect/apiRequesters
```

© 2018, 2019 IBM Corporation

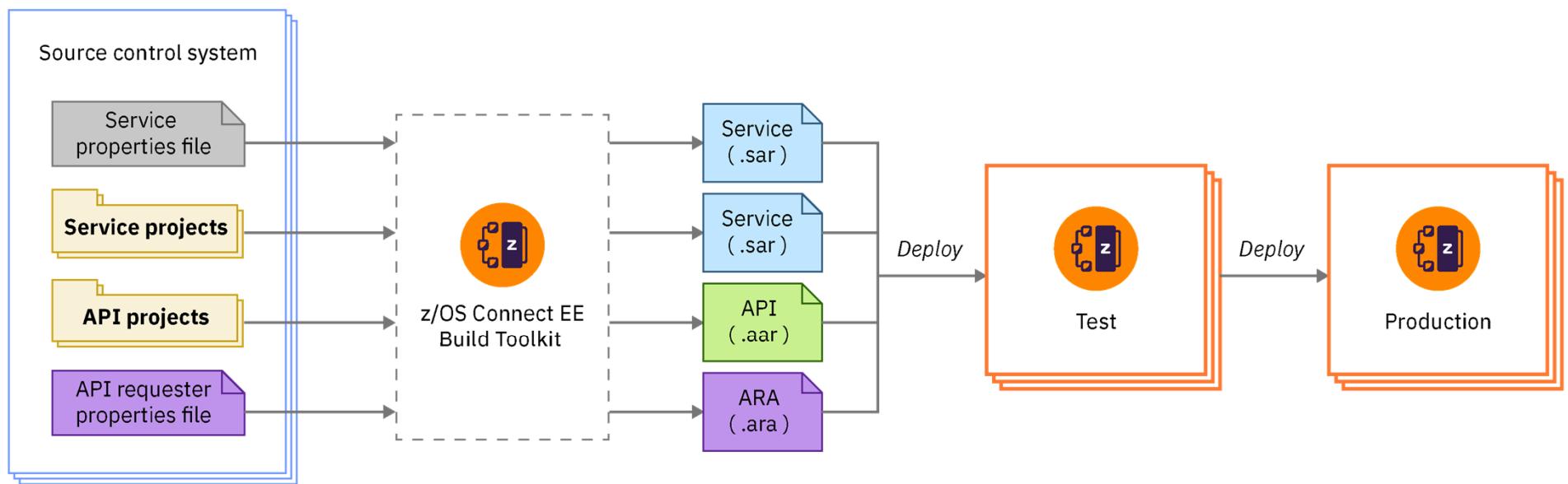


z/OS Connect EE

DevOps using z/OS Connect EE

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

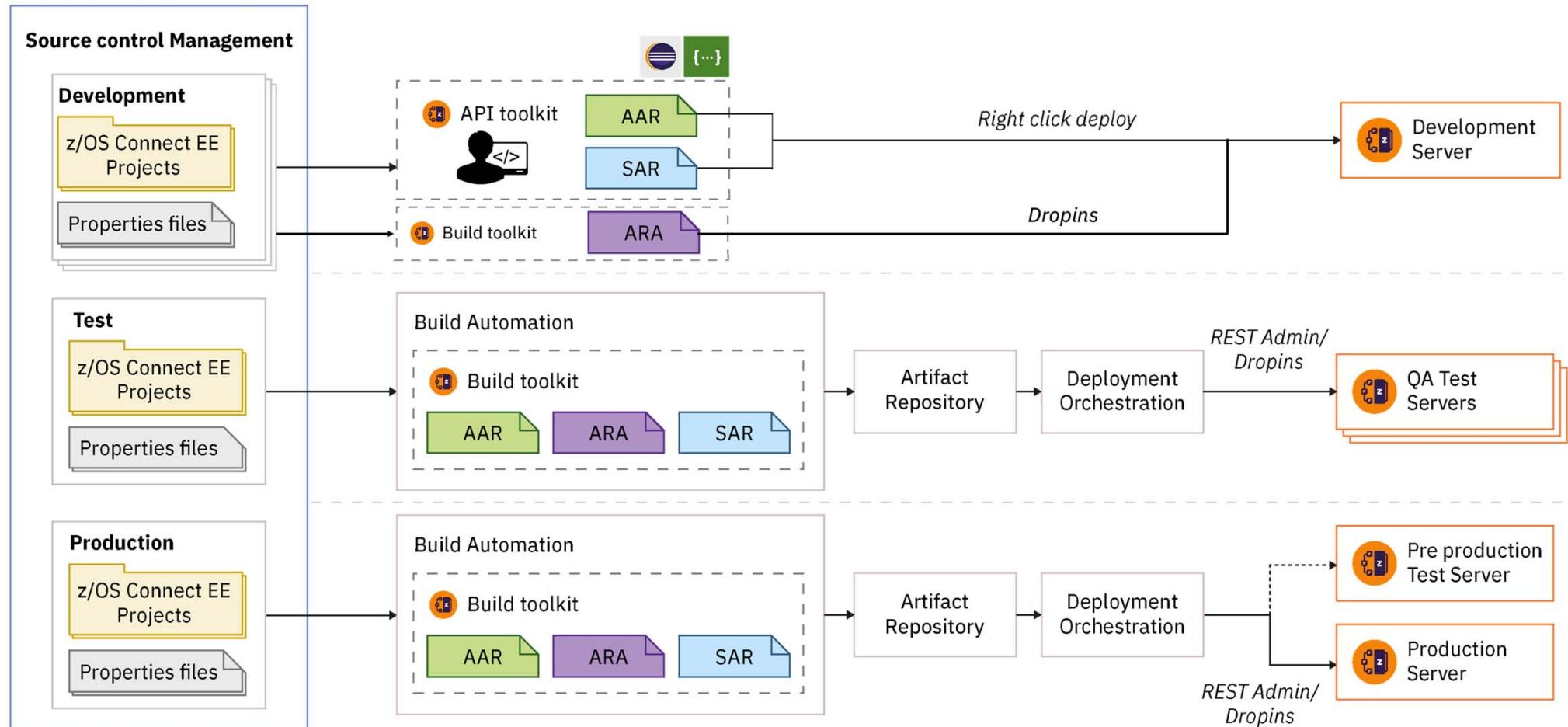
- The build toolkit supports the generation of service archives and API archives from projects created in the z/OS Connect EE API toolkit
- The build toolkit also supports the use of properties files to generate API requester archives
- Run the build toolkit from a build script to generate these archive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



DevOps Pipeline using z/OS Connect EE



z/OS Connect EE



z/OS Connect EE integration with Zowe

z/OS Connect EE and Zowe API Catalog – Administration API



The screenshot shows the API Catalog interface with a search bar and a sidebar for available API services. The services listed are:

- API Mediation Layer API**: Described as the API Mediation Layer for z/OS internal API services. It provides a single point of access to mainframe REST APIs and offers enterprise cloud-like feature... Status: All services are running.
- z/OS Connect EE Servers**: Described as z/OS Connect EE Administration APIs. Status: All services are running.
- z/OS Datasets services**: IBM z/OS Datasets REST services. Status: All services are running.
- z/OS Jobs services**: IBM z/OS Jobs REST services. Status: All services are running.

z/OS Connect administration API

API Version: 1.1.0

[Base URL: /zosConnect]

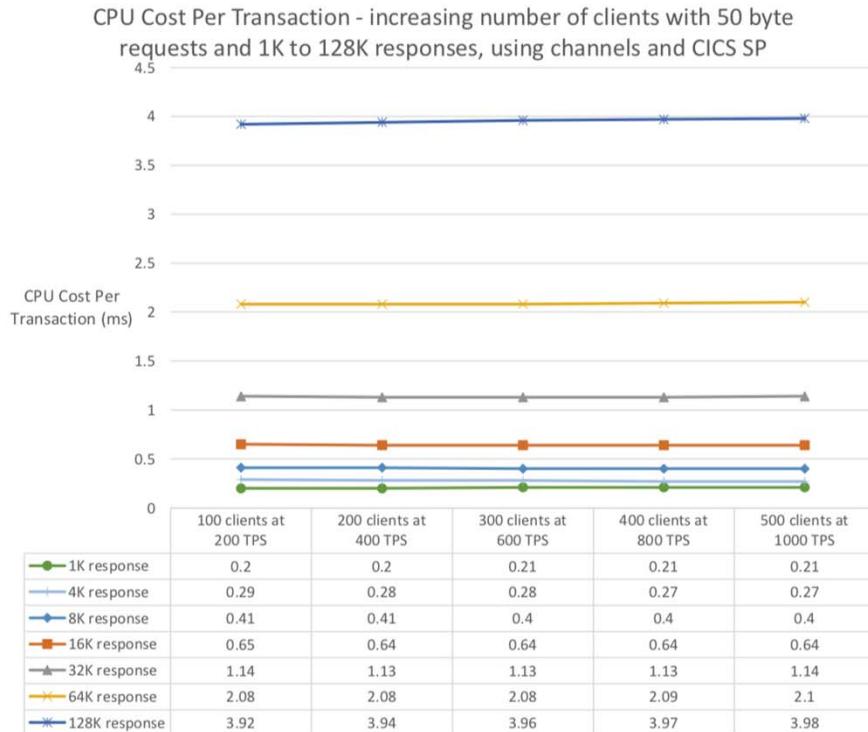
Interface providing meta-data and life-cycle operations for z/OS Connect services, APIs and API requesters.

APIs Operations for working with APIs

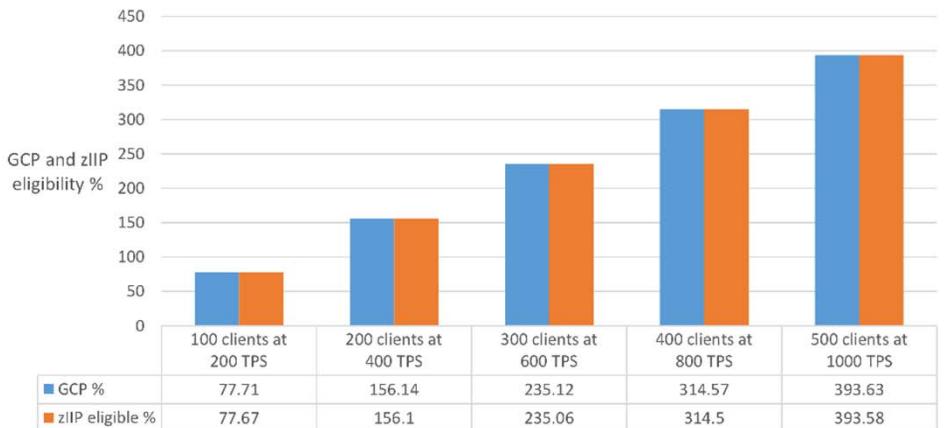
GET	/apis	Returns a list of all the deployed z/OS Connect APIs
POST	/apis	Deploys a new API into z/OS Connect
GET	/apis/{apiName}	Returns detailed information about a z/OS Connect API
PUT	/apis/{apiName}	Updates an existing z/OS Connect API
DELETE	/apis/{apiName}	Undeploys an API from z/OS Connect

Performance: API Provider

High Speed, High Throughput, Low Cost



zIIP eligibility - increasing number of clients with 50 byte requests and 128K responses, using channels and CICS SP



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



z/OS Connect EE

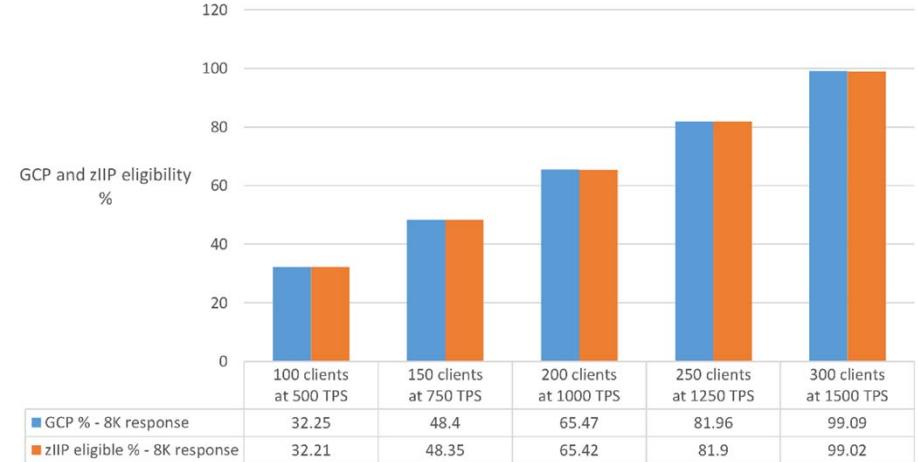
Performance: API Requester

High Speed, High Throughput, Low Cost

CPU Cost Per Transaction - increasing number of clients with API requester returning 1K, 4K and 8K API responses



zIIP eligibility - increasing number of clients with API requester returning 8K API responses



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



IBM z Omegamon for JVM



WG31 - 3270

File Edit View Communication Actions Window Help

File Edit View Tools Navigate Help 04/02/2019 19:10:36
Command ==> KJJZCSA z/OS Connect Request Summary
Auto Update : Off
SMF ID : WG31 Coll ID : KJJ1

APIName	Service	SoR ID	Reference	Resource
1. Last 30 Minute(s)	(HH:MM:SS.mmm)	(MM/DD/YYYY)		
2. Last 1 Hour(s)	Start Time 18:40:36.491 Date 04/02/2019			
3. Date/Time Range	End Time 19:10:36.491 Date 04/02/2019			

Columns 3 to 6 of 17 ← → ↑ ↓ Rows 1 to 8 of 8

ΔAPI VName	ΔHTTP VMethod	ΔRequest VCount	ΔError VCount	ΔTimeout VCount	ΔResp Time VAvg
ADMIN	GET	28	0	0	.000887s
catalog	GET	1	0	0	.019334s
cscvinc	GET	1	0	0	.008006s
db2employee	GET	2	0	0	.021797s
filea	GET	1	0	0	.005971s
filequeue	GET	1	0	0	.016206s
ADMIN	POST	1	0	0	.777651s
phonebook	GET	1	0	0	.345265s

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002
Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

WG31 - 3270

File Edit View Communication Actions Window Help 04/02/2019 19:09:41
Command ==> KJJZCSS Requests by Service Name
Auto Update : Off
SMF ID : WG31 Coll ID : KJJ1

APIName	Service	SoR ID	Reference	Resource
1. Last 30 Minute(s)	(HH:MM:SS.mmm)	(MM/DD/YYYY)		
2. Last 1 Hour(s)	Start Time 18:39:41.193 Date 04/02/2019			
3. Date/Time Range	End Time 19:09:41.193 Date 04/02/2019			

Columns 2 to 6 of 16 ← → ↑ ↓ Rows 1 to 6 of 6

ΔService VName	ΔRequest VCount	ΔError VCount	ΔTimeout VCount	ΔResp Time VAvg	ΔzOS Conne VAvg
inquireSingle	1	0	0	.019334s	.017995s
cscvincService	1	0	0	.008006s	.005515s
selectEmployee	2	0	0	.021797s	.021797s
Filea	1	0	0	.005971s	.005971s
FileaQueue	1	0	0	.016206s	.016206s
ivtnoService	1	0	0	.345265s	.163460s

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002
Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

WG31 - 3270

File Edit View Communication Actions Window Help 04/02/2019 18:48:18.198
Command ==> KJJZCSA z/OS Connect Event Summary
Auto Update : Off
SMF ID : WG31 Coll ID : KJJ1

ΔEvent VTime	ΔProvider VName	ΔUser ID V	ΔQuery VString
04/02/19 18:48:18.198	restclient-1.0	Fred	
04/02/19 18:48:34.790	restclient-1.0	Fred	
04/02/19 18:48:57.786	WOLA-1.0	Fred	
04/02/19 18:49:14.525	IBM MQ for z/OS	Fred	

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 12/034
Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

© 2018, 2019 IBM Corporation

IBM z Omegamon for JVM



WG31 - 3270

File Edit View Communication Actions Window Help

File Edit View Tools Navigate Help 04/02/2019 18:59:29
Auto Update : Off
SMF ID : WG31
Coll ID : KJJ1

Command ==> z/OS Connect Request Detail

Event time... 04/02/19 18:49:14.525
Request Type... API
API name... filequeue
Request URI... /filequeue/mq
Query String...
Method... GET
Port... 9453
HTTP code... 200 (OK)
Timeout... No
Service Name... FileaQueue
Total Red Time... 0.016206s
z/OS Conn Time... 0.016206s
SoR Resp Time... 0.000000s
SoR ID... NONE
SoR Ref... NONE
SoR Resource... NONE
Remote Address... 192.168.0.141
Request Length... 0
Response Length... 191
Correlator... e6e2d3d7d3c5e7400011000010d5ea51
Operation... getFilea
Provider... IBM MQ for z/OS
User ID... Fred

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

Request Type... API
API name... db2employee
Request URI... /db2/employee/000020
Query String...
Method... GET
Port... 9453
HTTP code... 200 (OK)
Timeout... No
Service Name... selectEmployee
Total Red Time... 0.022592s
z/OS Conn Time... 0.022592s
SoR Resp Time... 0.000000s
SoR ID... NONE
SoR Ref... NONE
SoR Resource... NONE
Remote Address... 192.168.0.141
Request Length... 0
Response Length... 326
Correlator... e6e2d3d7d3c5e7400011000010d5ea50
Operation... getSelectEmployee
Provider... restclient-1.0
User ID... Fred

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

WG31 - 3270

File Edit View Communication Actions Window Help

File Edit View Tools Navigate Help 04/02/2019 19:00:52
Auto Update : Off
SMF ID : WG31
Coll ID : KJJ1

Command ==> z/OS Connect Request Detail

Event time... 04/02/19 18:47:54.267
Request Type... API
API name... cscvinc
Request URI... /cscvinc/employee/444444
Query String...
Method... GET
Port... 9453
HTTP code... 200 (OK)
Timeout... No
Service Name... cscvincService
Total Red Time... 0.000006s
z/OS Conn Time... 0.005515s
SoR Resp Time... 0.002491s
SoR ID... USIBMWZ.CICS532
SoR Ref... cscvinc
SoR Resource... CSM1,CSCVINC
Remote Address... 192.168.0.141
Request Length... 0
Response Length... 302
Correlator... e6e2d3d7d3c5e7400011000010d5ea50
Operation... getGscvincService
Provider... CICS-1.0
User ID... Fred

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

Request Type... API
API name... phonebook
Request URI... /phonebook/contacts/LAST1
Query String...
Method... GET
Port... 9453
HTTP code... 200 (OK)
Timeout... No
Service Name... ivtnoService
Total Red Time... 0.345265s
z/OS Conn Time... 0.163460s
SoR Resp Time... 0.181805s
SoR ID... IVP1
SoR Ref... IMSCONN
SoR Resource... IVTNO
Remote Address... 192.168.0.141
Request Length... 0
Response Length... 158
Correlator... e6e2d3d7d3c5e7400011000010d5ea55
Operation... getPhoneBookService1
Provider... imsmobile-2.0
User ID... Fred

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

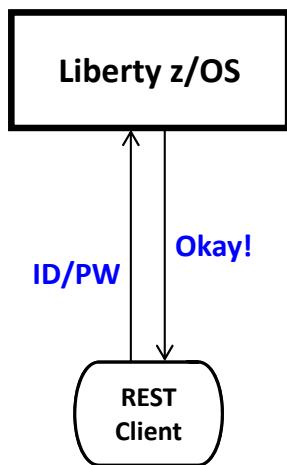
Authentication



z/OS Connect EE

Several different ways this can be accomplished:

Basic Authentication



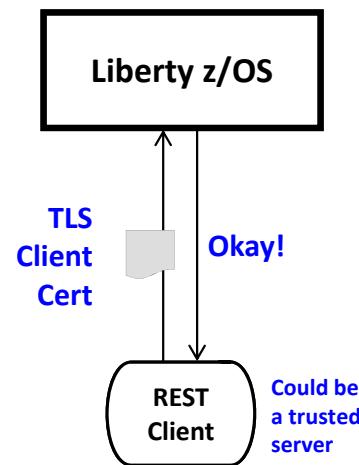
Server prompts for ID/PW

Client supplies ID/PW

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

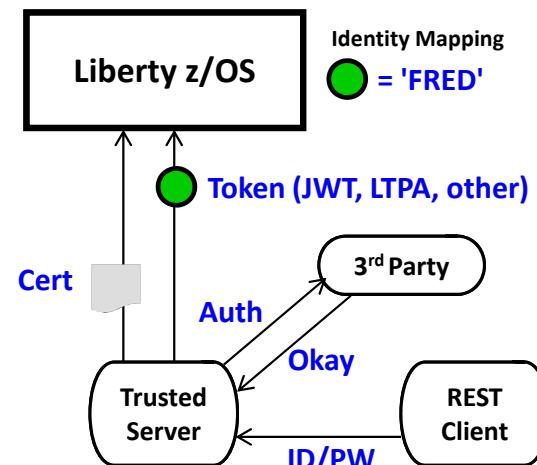
Client supplies certificate

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:

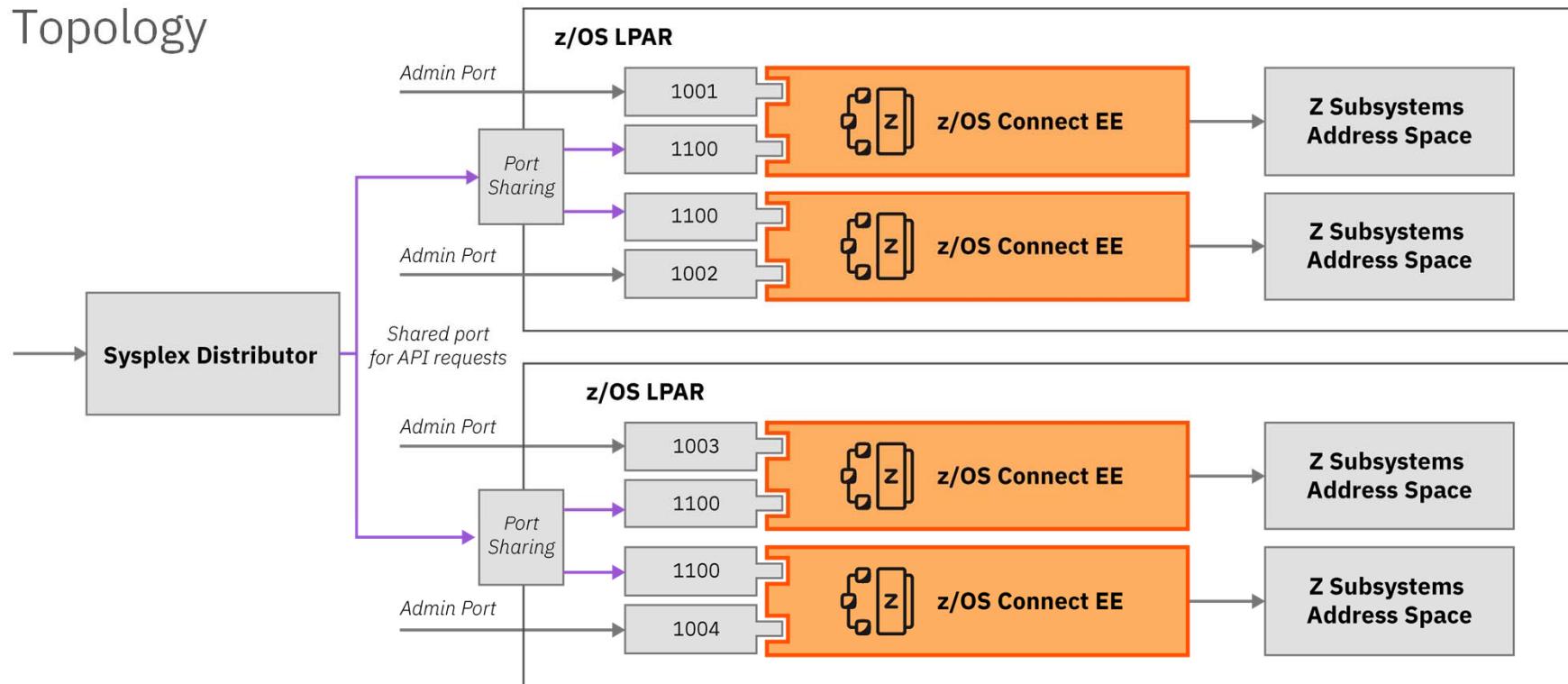
- LDAP
- SAF

High Availability



z/OS Connect EE

Topology



ibm.biz/zosconnect-ha-concepts

ibm.biz/zosconnect-scenarios



/questions?thanks=true

Thank you for listening.



/exercises

basic security; exercise paths



z/OS Connect EE

Exercises – Two paths or options

- ❑ Basic Configuration Hands-on Lab
 - ❑ Configure a z/OS Connect Server
 - ❑ Develop and deploy a Service
 - ❑ Develop and deploy an API
 - ❑ Test using Swagger UI
 - ❑ Enable Security (SAF and SSL)

Or one or more of the following:

- ❑ Developing APIs Hands-on Labs
 - ❑ CICS Container/COMMAREA
 - ❑ DB2
 - ❑ IMS Transaction
 - ❑ MQ
 - ❑ MVS Batch
 - ❑ Outbound RESTful applications

- Material can be downloaded from:
<http://tinyurl.com/y28fsezs>
- [z/OS Connect EE Users Group](https://www.linkedin.com/groups/8731382/)
<https://www.linkedin.com/groups/8731382/>
- Copy/Paste files on desktop
 - Basic Configuration CopyPaste
 - Developing APIs CopyPaste
- Identities:
 - RACF identity: USER1→ Password: USER1
 - zCEE identity: Fred → Password: fredpwd
- 3270 Key Sequences
 - Clear screen: Fn-P
 - Enter key: right CTRL