



IBM z/OS Connect Enterprise Edition

Introduction and Overview

Mitch Johnson

mitchj@us.ibm.com

Washington System Center

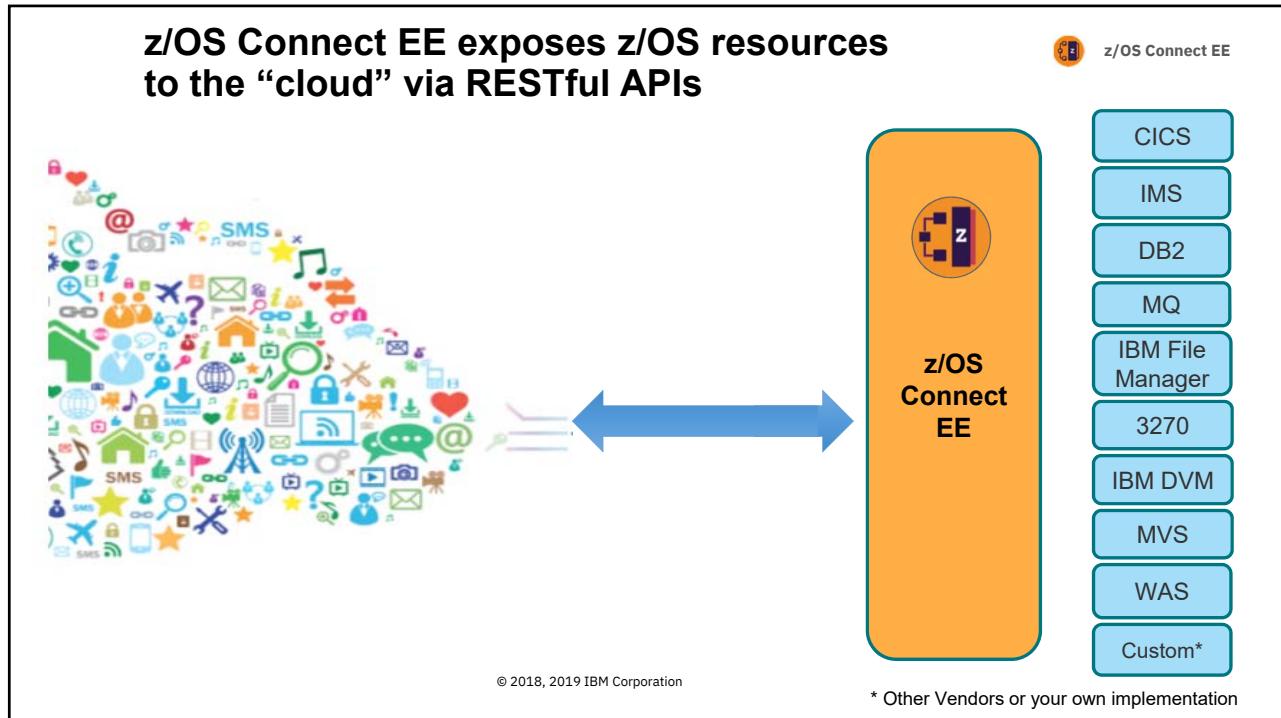


© 2018, 2019 IBM Corporation

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

© 2018, 2019 IBM Corporation



/but_first, what_is_REST?

What makes an API “RESTful”?

© 2018, 2019 IBM Corporation

REST is an Architectural Style



z/OS Connect EE

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.

© 2018,2019 IBM Corporation

Key Principles of REST



z/OS Connect EE

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

GET
POST
PUT
DELETE

```

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" }
  
```

Request/Response Body is used to represent the data object

Path and Query parameters are used for refinement of the request

URIs represent things (or lists of things)

© 2018,2019 IBM Corporation

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

© 2018,2019 IBM Corporation

7

RESTful Examples



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>

© 2018,2019 IBM Corporation

Not every REST API is a RESTful API



z/OS Connect EE

(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" }
```

© 2018,2019 IBM Corporation

Why is REST popular?



z/OS Connect EE

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.

© 2018,2019 IBM Corporation

How do we describe a REST API?

© 2018, 2019 IBM Corporation



/swagger/open_api

The industry standard framework for describing RESTful APIs.

© 2018, 2019 IBM Corporation

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/>

Example: <https://developer PSA-peugeot-citroen.com/Inc/>

13



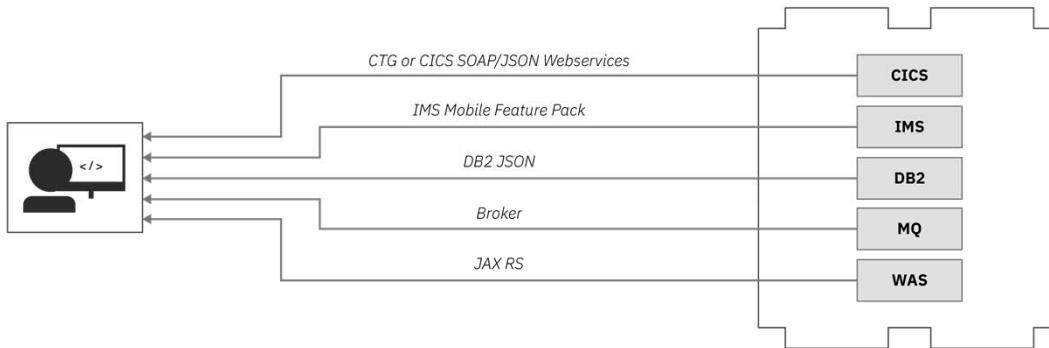
Why /zos_connect_ee?

Truly RESTful APIs to and from your mainframe.

© 2018 , 2019 IBM Corporation

Can't we do REST and JSON today?

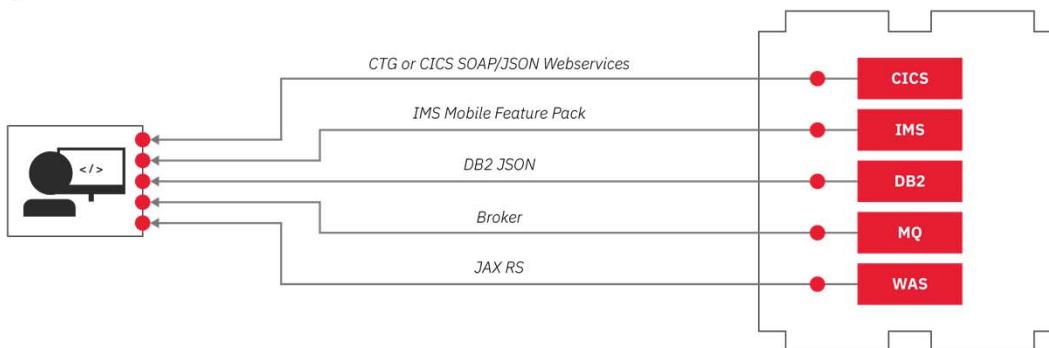
 z/OS Connect EE



© 2018, 2019 IBM Corporation

Yes, but....

 z/OS Connect EE



Completely different configuration and management.

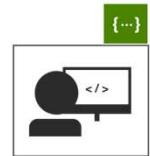
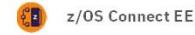
Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!

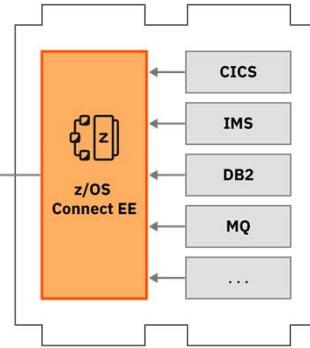
© 2018, 2019 IBM Corporation

A single entry point is needed

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.

© 2018, 2019 IBM Corporation



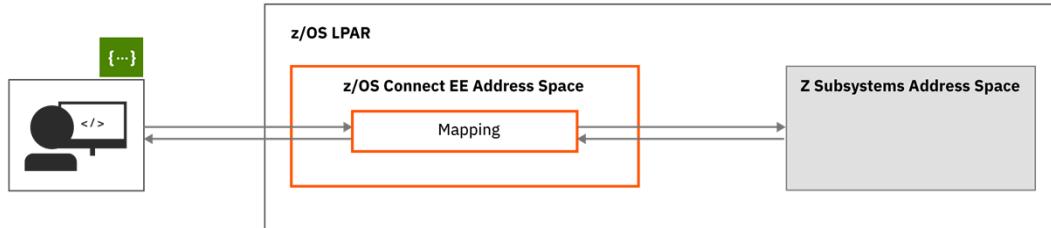
**Other than a RESTful interface,
what does z/OS Connect provide?**

© 2018, 2019 IBM Corporation

Let's Start with Data mapping



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

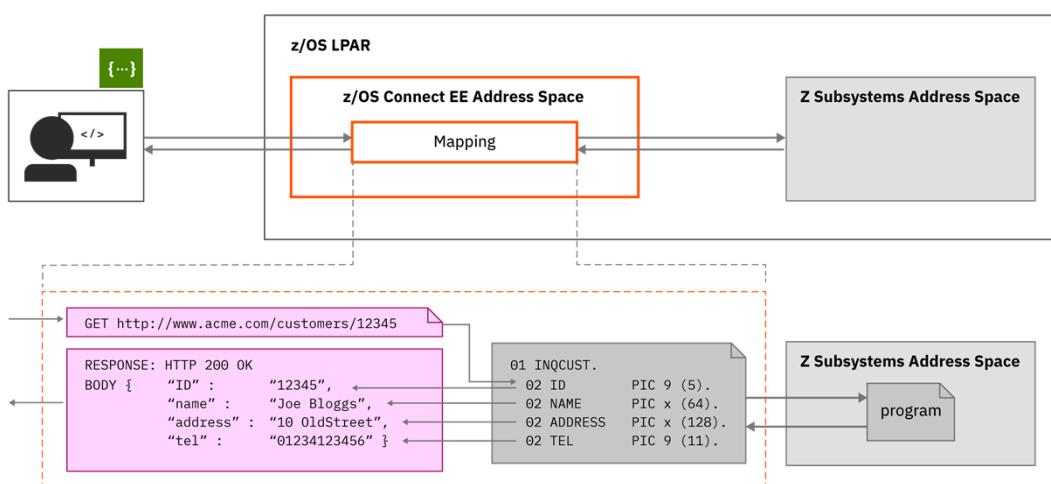


© 2018, 2019 IBM Corporation

Data mapping Example



A closer look



© 2018, 2019 IBM Corporation

COBOL versus JSON Example



COBOL Source v JSON

```
01 MINTLOAN-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_commarea": {
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "maxLength": 20
    },
    "creditScore": {
      "type": "number",
      "format": "decimal",
      "multipleOf": 0.01,
      "maximum": 99999999999999.99,
      "minimum": 0
    }
  }
},
```

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

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

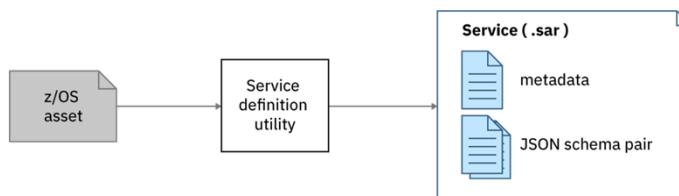
© 2018, 2019 IBM Corporation

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

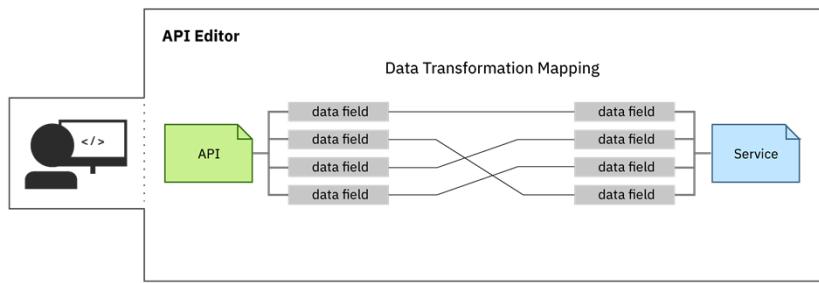
ibm.biz/zosconnect-sar-creation

© 2018,2019 IBM Corporation

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.

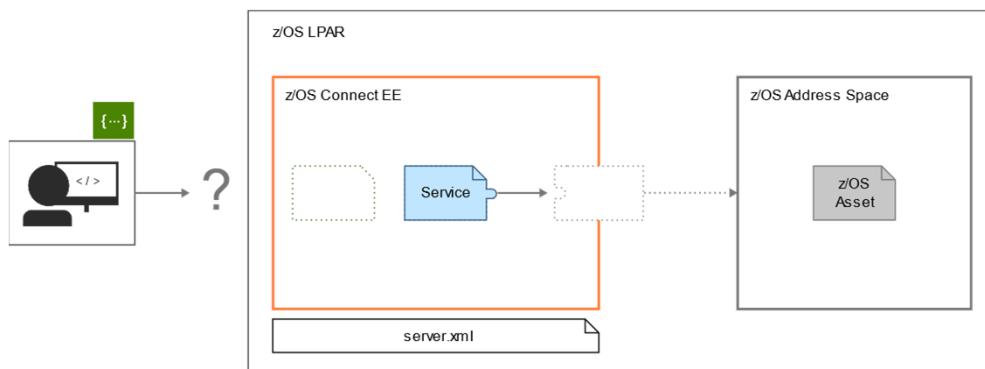
<ibm.biz/zosconnect-create-api>

© 2018,2019 IBM Corporation

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.

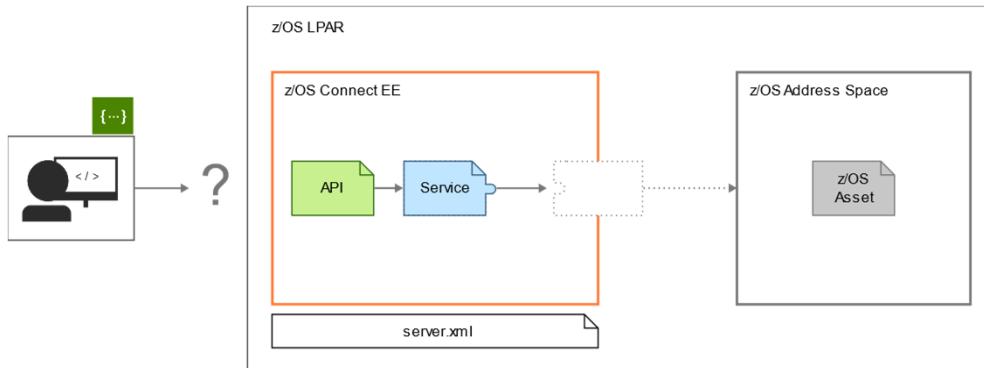
<ibm.biz/zosconnect-define-services>

© 2018, 2019 IBM Corporation

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.

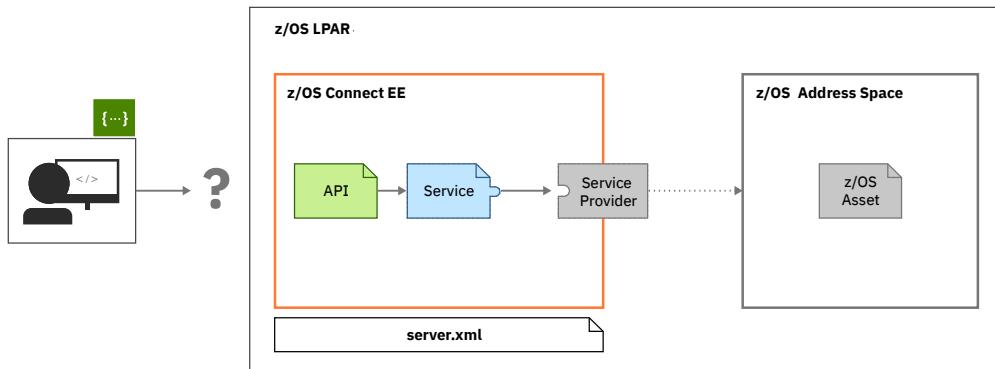
<ibm.biz/zosconnect-deploy-api>

© 2018, 2019 IBM Corporation

Steps to expose a z/OS application



5. Configure your service provider

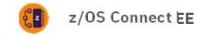


Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.

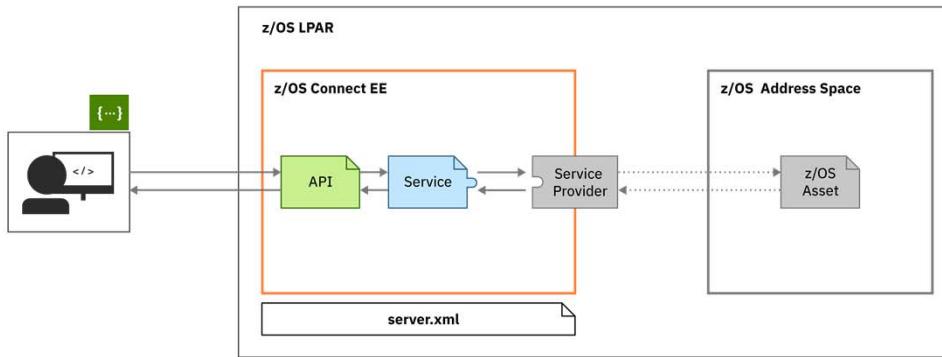
<ibm.biz/zosconnect-configuring>

© 2018, 2019 IBM Corporation

Steps to expose a z/OS application



6. Done



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

© 2018, 2019 IBM Corporation

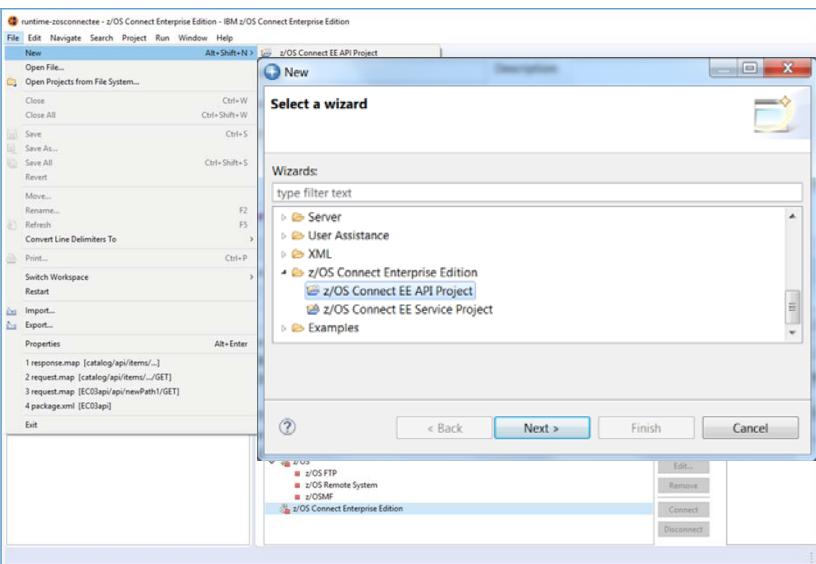


/api_toolkit/services

Simple service creation.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



The screenshot shows the 'Select a wizard' dialog box within the Eclipse-based tooling interface. The 'Wizards:' list includes 'Server', 'User Assistance', 'XML', 'z/OS Connect Enterprise Edition' (which is expanded to show 'z/OS Connect EE API Project' and 'z/OS Connect EE Service Project'), and 'Examples'. Below the dialog is a list of connected subsystems: z/OS FTP, z/OS Remote System, z/OSMF, and z/OS Connect Enterprise Edition (which is selected). Buttons for 'Edit...', 'Remove', 'Connect', and 'Disconnect' are visible.

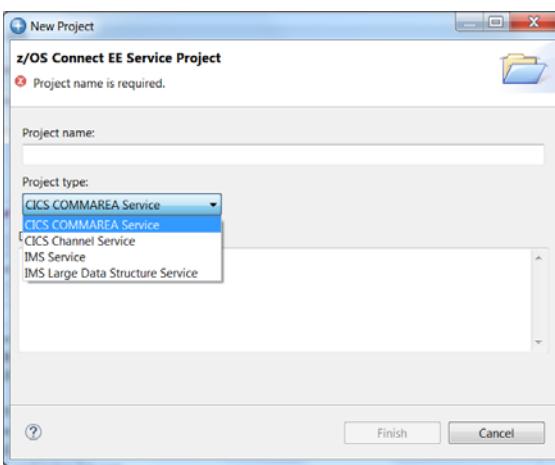
Use the **API toolkit** to create services through Eclipse-based tooling.

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

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS

Service creation – a common interface



The screenshot shows the 'New Project' dialog for 'z/OS Connect EE Service Project'. A red error message 'Project name is required.' is displayed above the 'Project name:' input field. The 'Project type:' dropdown menu is open, showing 'CICS COMMAREA Service' (which is selected), 'CICS COMMAREA Service', 'CICS Channel Service', 'IMS Service', and 'IMS Large Data Structure Service'. Buttons for '?', 'Finish', and 'Cancel' are at the bottom.

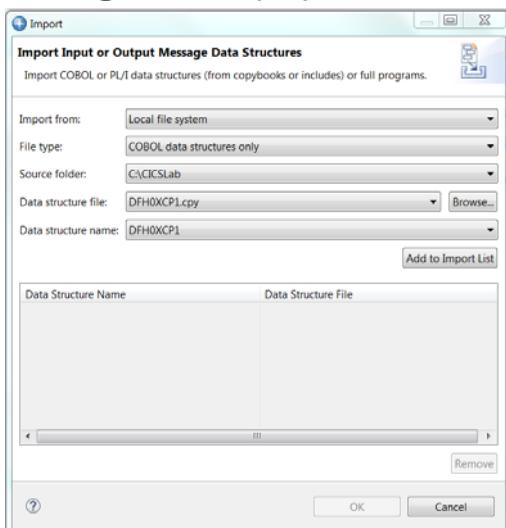
A common interface for service creation, agnostic of back end subsystem.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS

 z/OS Connect EE

Creating a service project



The dialog shows the following settings:

- Import from: Local file system
- File type: COBOL data structures only
- Source folder: C:\CICSLab
- Data structure file: DFHOXCP1.cpy
- Data structure name: DFHOXCP1

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

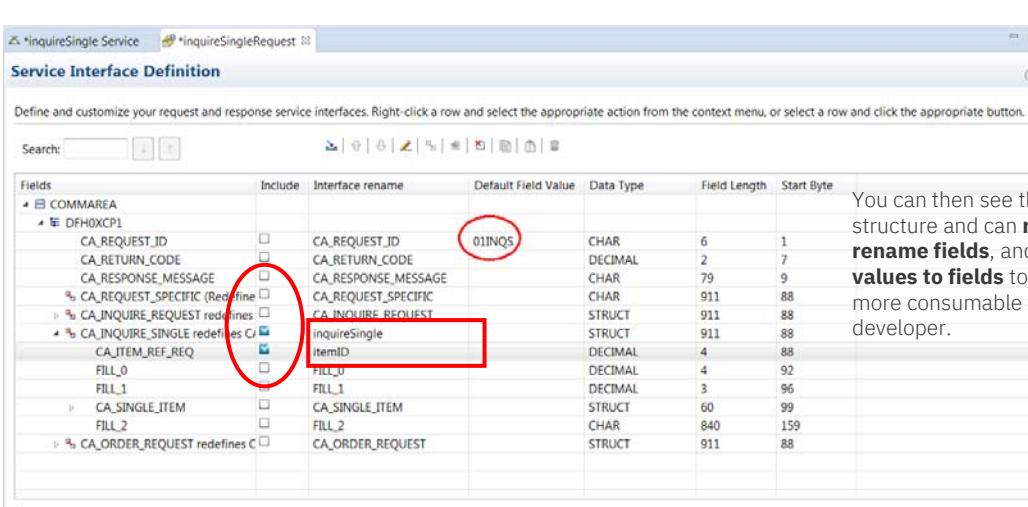
The service interface supports complex data structures, including OCCURS DEPENDING ON and REDEFINES clauses.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS

 z/OS Connect EE

Creating a service interface definition



The table displays the following fields and their properties:

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFHOXCP1						
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 (Req fine)	<input type="checkbox"/>	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines CA_INQUIRE_REQUEST	<input checked="" type="checkbox"/>	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines CA_INQUIRE_SINGLE	<input checked="" type="checkbox"/>	inquireSingle		STRUCT	911	88
CA_ITEM_REF_REQ	<input 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_ORDER_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.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS

Creating a service – response message

 z/OS Connect EE

The Service Interface Definition (SID) tool allows you to 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						
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 C	<input 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"/>	singlItem		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 type="checkbox"/>	inStock		DECIMAL	4	152
ON_SNGL_ORDER	<input type="checkbox"/>	onOrder		DECIMAL	3	156
FILL_2	<input type="checkbox"/>	FILL_2		CHAR	840	159
CA.ORDER.REQUEST redefines C	<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

© 2018, 2019 IBM Corporation

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

API toolkit – Creating Services for CICS and IMS

Creating a “GET” service interface request definition

 z/OS Connect EE

The Service Interface Definition (SID) tool allows you to 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
IVTNODisplayRequest						
Segment 1						
INPUT_MSG						
IN_LL	<input type="checkbox"/>	IN_LL		SHORT	2	1
IN_ZZ	<input type="checkbox"/>	IN_ZZ		SHORT	2	3
IN_TRANCODE	<input type="checkbox"/>	IN_TRANCODE		CHAR	10	5
IN_COMMAND	<input type="checkbox"/>	IN_COMMAND	IVTNO DISPLAY	CHAR	8	15
IN_LAST_NAME	<input checked="" type="checkbox"/>	lastName		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

© 2018, 2019 IBM Corporation

The service developer creates distinct services for each function.

DISPLAY
DELETE
ADD
UPDATE

API toolkit – Creating Services for CICS and IMS

Creating a “GET” service interface response definition

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
*ivtnoDisplayResponse						
Segment 1						
OUT_OUTPUT_AREA						
OUT_LL	<input checked="" type="checkbox"/>	OUT_LL		SHORT	2	1
OUT_ZZ	<input checked="" type="checkbox"/>	OUT_ZZ		SHORT	2	3
message	<input checked="" type="checkbox"/>	message		CHAR	40	5
OUT_COMMAND	<input checked="" 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 checked="" type="checkbox"/>	OUT_SEGMENT_NO		CHAR	4	90

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS

Creating a service for CICS and IMS

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

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS

Creating a service for CICS and IMS

The screenshot shows the z/OS Connect Enterprise Edition interface. On the left is the Project Explorer with a service named 'InquireSingle'. In the center, the 'Overview' tab of the 'InquireSingle Service' properties window is displayed. It shows the service type as 'CICS COMMAREA Service', version '13.0', and a description 'Coding example Inquire Single Service'. Below this are sections for 'Program' (set to 'DHCICLMM') and 'Define Request and Response Service Interface'. A right-click context menu is open over the service entry in the Project Explorer, with the option 'Deploy Service to z/OS Connect EE Server...' highlighted. To the right, a 'Deploy Service' dialog box is open, showing the service name 'inquireSingle', version '13.00', and type 'CICS COMMAREA Se...'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

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

© 2018, 2019 IBM Corporation

Creating Services without the Toolkit - 1

For MQ and MVS Batch use the supplied conversion utility BAQLS2JS

The diagram illustrates the conversion process. On the left, two rectangular boxes represent 'Request COPYBOOK' and 'Response COPYBOOK'. An arrow points from the Request COPYBOOK to a yellow rounded rectangle labeled 'baqls2js'. Another arrow points from the Response COPYBOOK to the same 'baqls2js' box. Below this, a box labeled 'PARMS for baqls2js' has an arrow pointing to the 'baqls2js' box. From the 'baqls2js' box, four arrows point to the right, each leading to a cylinder icon representing a file. These files are labeled: 'BIND Files', 'JSON Schema Files', 'Service Archive (SAR) File', and another unlabeled cylinder. A note at the bottom left states: 'This is a "bottom up" pattern (language structure to JSON). There is also a "top down" pattern (JSON to language structure). That is baqls2ls utility.'

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.

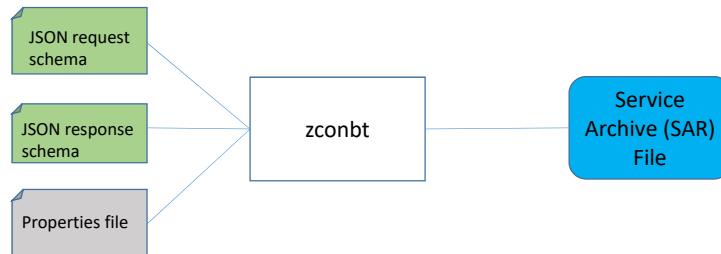
Service Archive (SAR) File
This is a ZIP-format file that contains the JSON schema and some meta-data. This is input to the API Editor (next unit) to create the APIs.
This is placed in a USS file system location based on input parms you specify.

© 2018, 2019 IBM Corporation

Creating Services without the Toolkit – 2a



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



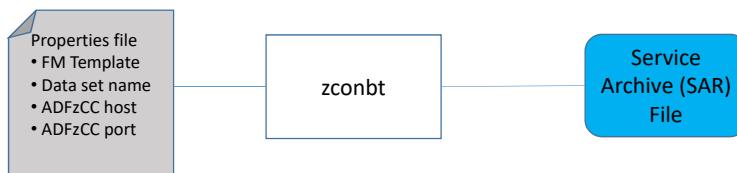
Generate the service archive file

© 2018, 2019 IBM Corporation

Creating Services without the Toolkit – 2b



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



Generate the service archive file

© 2018, 2019 IBM Corporation

Creating Services without the Toolkit – Part 3



z/OS Connect EE

For DVM use the DVM Studio

```
-- 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.EXPLAPP.EMPCAT
SELECT WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTMENT, WS_COST, WS_IN_STOCK,
WS_ON_ORDER
FROM CATALOG LIMIT 1000;
```

WS_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
all Pens ...	10	002.90	80	0
encil wits...	10	001.78	83	0
total lines...	10	000.84	47	0

© 2018, 2019 IBM Corporation



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

Quick and easy API mapping.

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

© 2018, 2019 IBM Corporation



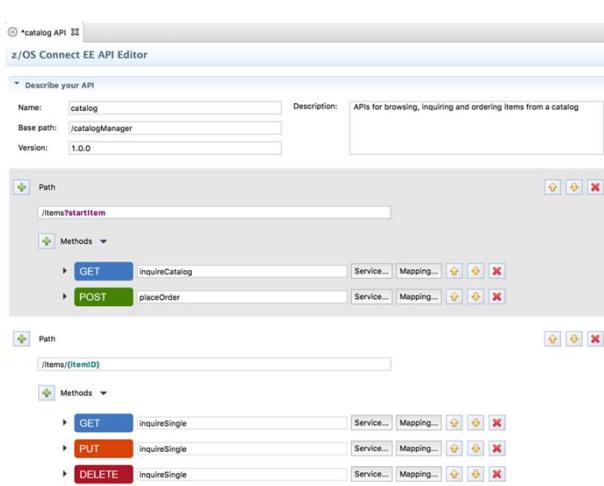
/api_toolkit/api_editor

Quick and easy API mapping.

© 2018, 2019 IBM Corporation

API toolkit – API Editor

API definition



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

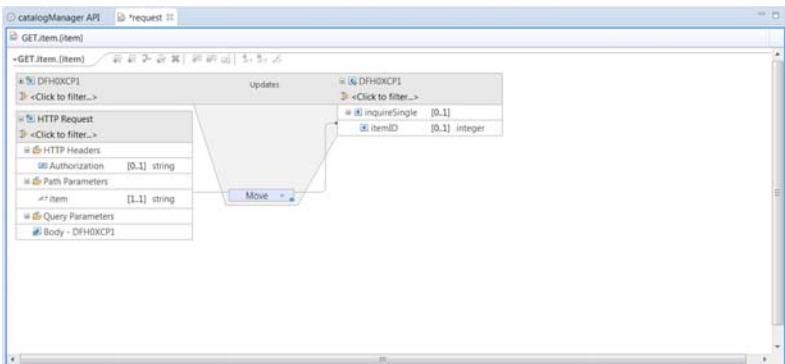
Once you define your API, you can map backend services to each request.

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

© 2018, 2019 IBM Corporation

API toolkit – API Editor

API mapping: Point-and-click interface



The screenshot shows the API Editor interface. On the left, a tree view lists the request structure: catalogManager API > GET.item.(Item) > DFH0XCP1 > Click to filter... > HTTP Request > Click to filter... > item > itemID. On the right, a tree view shows the response structure: DFH0XCP1 > Click to filter... > inquireSingle [0..1] > itemID [0..1] integer. A 'Move' button is visible between the two trees.

Map both the request and response for each API.

Map path and query parameters to native data structures.

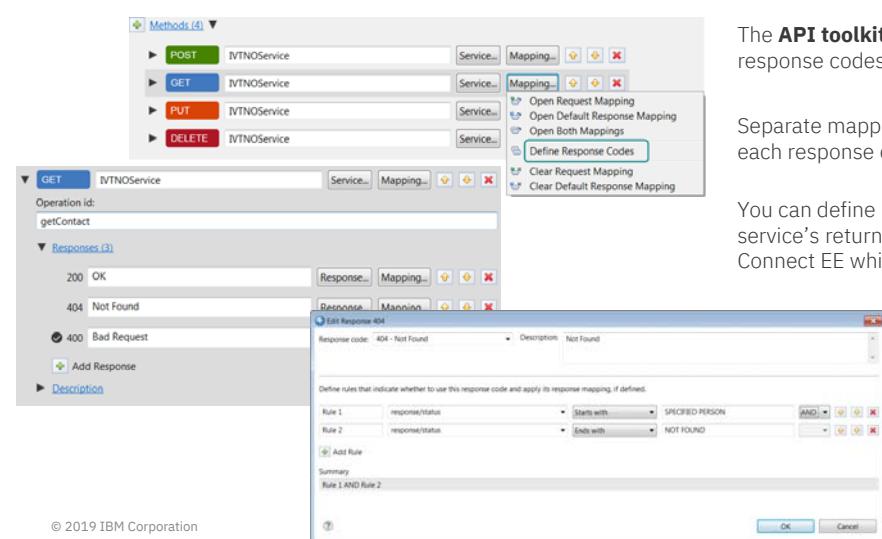
Assign static values to fields, useful for Op codes.

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

© 2018, 2019 IBM Corporation

API toolkit

API definition with multiple response codes



The screenshot shows the API toolkit interface. In the top navigation bar, 'Methods (4)' is selected. Under 'GET', there are four entries: POST IVTNOService, GET IVTNOService, PUT IVTNOService, and DELETE IVTNOService. For the GET entry, the 'Mapping...' button is highlighted. A context menu is open with options: Open Request Mapping, Open Default Response Mapping, Open Both Mappings, and Define Response Codes. The 'Define Response Codes' option is selected. Below this, under 'Responses (3)', there are three entries: 200 OK, 404 Not Found, and 400 Bad Request. The 404 entry is selected, and its 'Mapping...' button is also highlighted. A detailed dialog box titled 'Edit Response 404' is open, showing a table with rules for defining when to use the response code. Rule 1: response/status Starts with SPECIFIED PERSON AND Rule 2: response/status Ends with NOT FOUND.

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

© 2019 IBM Corporation

API toolkit – API Editor

API mapping: Point-and-click interface

Allows the API Developer to remove fields to simplify the API

© 2018, 2019 IBM Corporation

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.

© 2018, 2019 IBM Corporation

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.

© 2018, 2019 IBM Corporation

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

?

Distributed system

z/OS system

Cloud service

RESTful endpoint

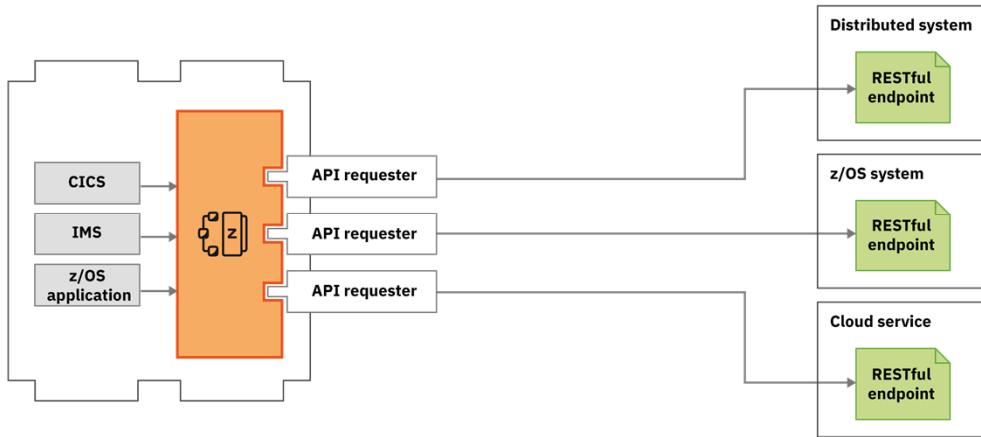
RESTful endpoint

RESTful endpoint

© 2018, 2019 IBM Corporation

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

 z/OS Connect EE

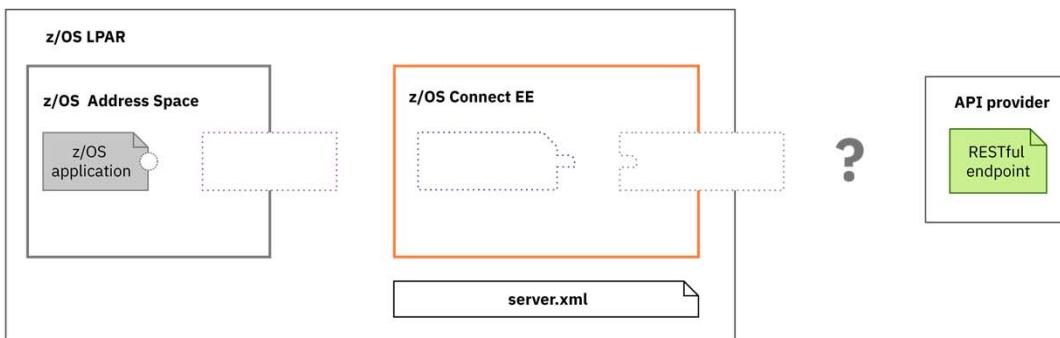


© 2018, 2019 IBM Corporation

Steps to calling an external API

 z/OS Connect EE

Starting point

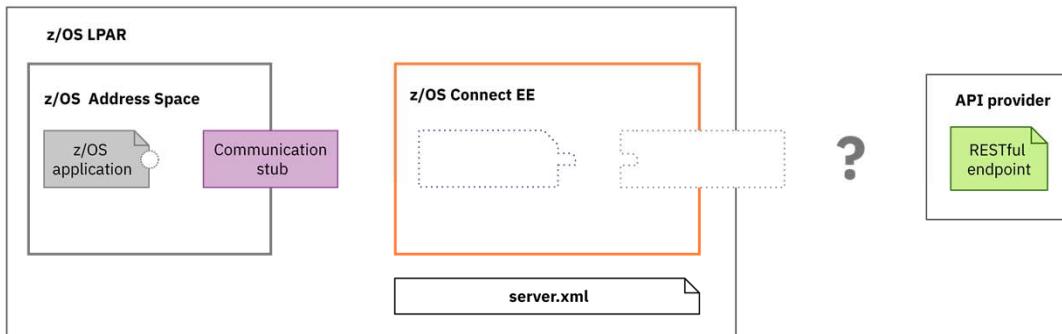


© 2018, 2019 IBM Corporation

Steps to calling an external API



Step 1. Configure communication stub



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

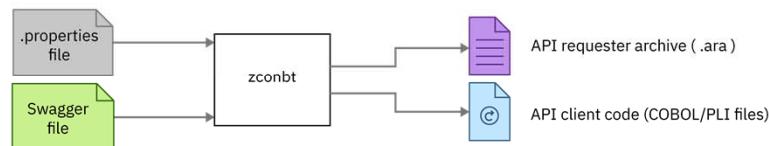
ibm.biz/zosconnect-configure-comms-stub

© 2018, 2019 IBM Corporation

Steps to calling an external API



Step 2. Generate API requester archive from Swagger



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

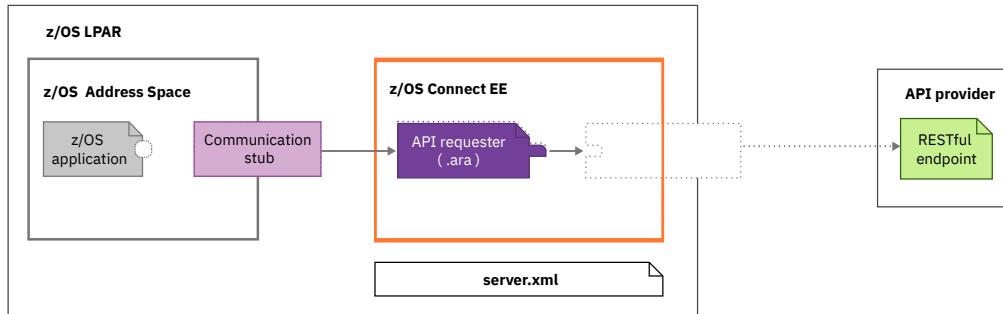
ibm.biz/zosconnect-generate-ara

© 2018, 2019 IBM Corporation

Steps to calling an external API



Step 3. Deploy API requester (.ara) archive



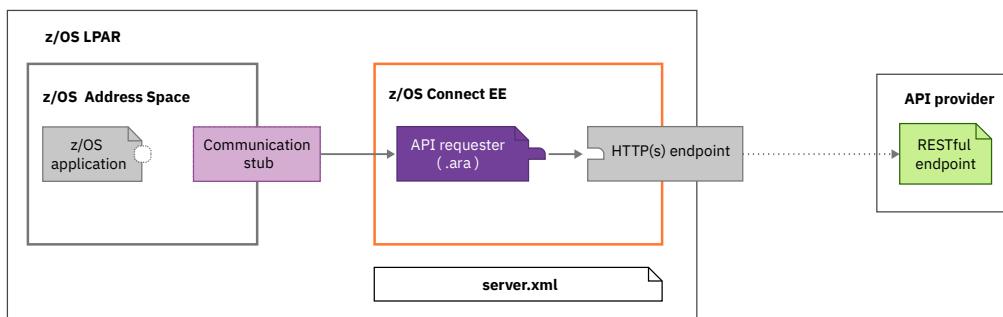
Deploy your API requester archive to the *apiRequester* directory.

© 2018, 2019 IBM Corporation

Steps to calling an external API



Step 4. Configure HTTP(S) endpoint



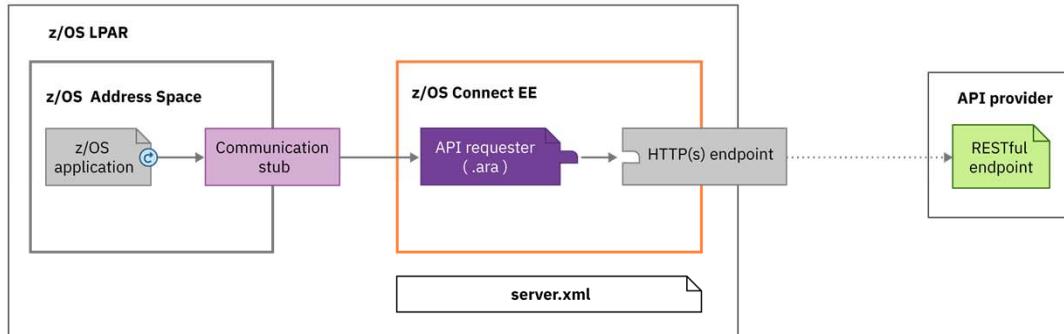
Configure the connection between z/OS Connect EE and the external API.

ibm.biz/zosconnect-configure-endpoint-connection

© 2018, 2019 IBM Corporation

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

© 2018, 2019 IBM Corporation

Steps to calling an external API

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

The screenshot shows the **GETAPI** editor with several windows open:

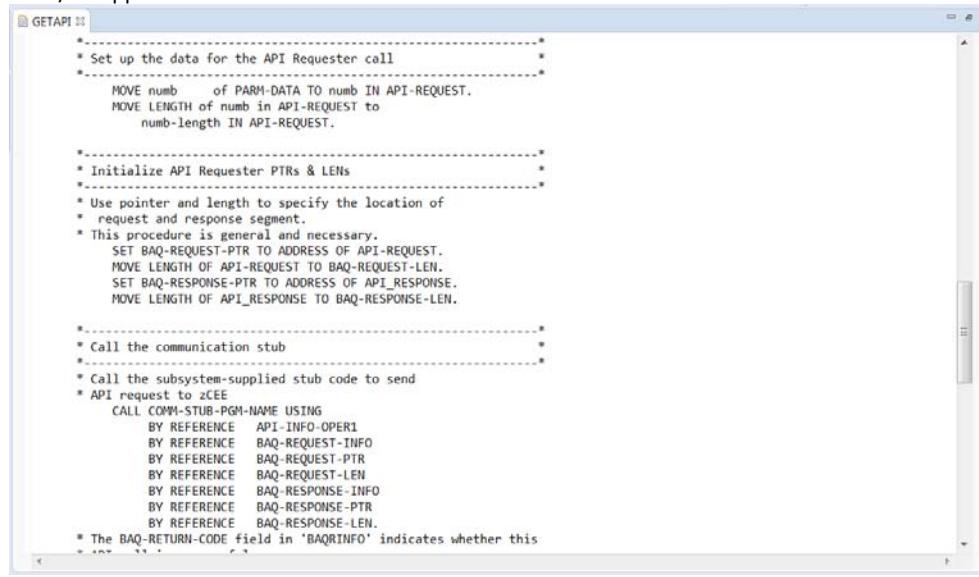
- GETAPI**: Shows the **ERROR MESSAGE STRUCTURE** with fields like **01 ERROR-MSG.**, **03 EM-ORIGIN**, **03 EM-CODE**, and **03 EM-DETAIL**.
- CSC02I01**: Shows the **STRUCTURE WITH THE API** with fields like **03 BAQ-APINAME**, **03 BAQ-APINAME-LEN**, **03 BAQ-APIPATH**, **03 BAQ-APIPATH-LEN**, **03 BAQ-APIMETHOD**, and **03 BAQ-APIMETHOD-LEN**.
- apis.xml**: Shows the XML configuration for the API requester, including the **zosconnect_apiRequesters** section with a basicAuthData entry for user "fred" and password "fredpwd".
- apiDescriptionFile**: Shows the Swagger definition for the API, including **dataStructuresLocation**, **apiInfofileLocation**, **logFileDirectory**, **language** (COBOL), **connectionRef** (cscvincAPI), and **requesterPrefix** (csc).

A red arrow points to the **CSC02I01** window, indicating where the new copy book was added.

© 2018, 2019 IBM Corporation

Steps to calling an external API

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



```

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

-----*
* Initialize API Requester PTRs & LENS
*-----*
      MOVE LENGTH of API-REQUEST to BAQ-REQUEST-LEN.
      SET BAQ-REQUEST-PTR TO ADDRESS OF API-REQUEST.
      MOVE LENGTH of BAQ-REQUEST-LEN to BAQ-REQUEST-LEN.
      SET BAQ-RESPONSE-PTR TO ADDRESS OF API_RESPONSE.
      MOVE LENGTH of API_RESPONSE to BAQ-RESPONSE-LEN.

-----*
* 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 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.

-----*
* 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.

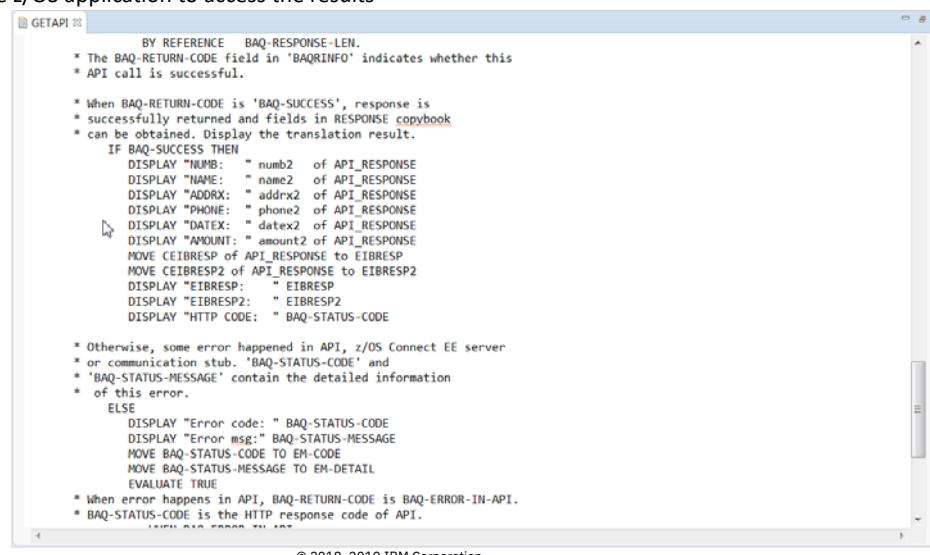
-----*
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
* API call is successful.

```

© 2018, 2019 IBM Corporation

Steps to calling an external API

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



```

GETAPI :>
-----*
* BY REFERENCE BAQ-RESPONSE-LEN.
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
* API call is successful.
-----*
* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is
* successfully returned and fields in RESPONSE copybook
* can be obtained. Display the translation result.
      IF BAQ-SUCCESS THEN
        DISPLAY "NUMB: " numb2 of API_RESPONSE
        DISPLAY "NAME: " name2 of API_RESPONSE
        DISPLAY "ADDRX: " addrx2 of API_RESPONSE
        DISPLAY "PHONE: " phone2 of API_RESPONSE
        DISPLAY "DATEX: " datex2 of API_RESPONSE
        DISPLAY "AMOUNT: " amount2 of API_RESPONSE
        MOVE CEIBRESP of API_RESPONSE to EIBRESP
        MOVE CEIBRESP2 of API_RESPONSE to EIBRESP2
        DISPLAY "EIBRESP: " EIBRESP
        DISPLAY "EIBRESP2: " EIBRESP2
        DISPLAY "HTTP CODE: " BAQ-STATUS-CODE

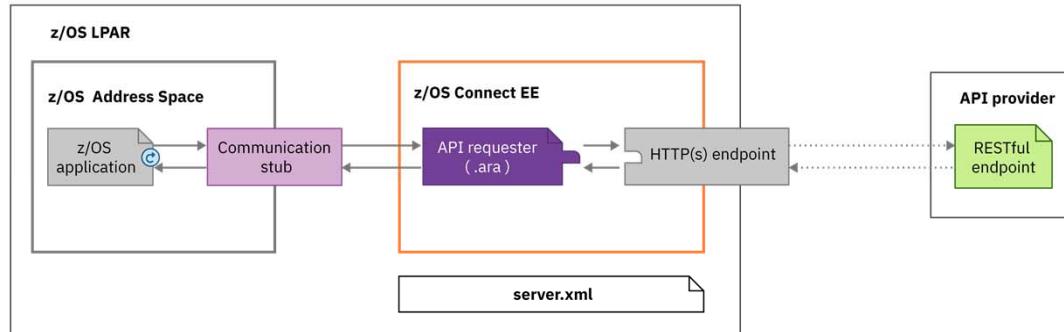
-----*
* Otherwise, some error happened in API, z/OS Connect EE server
* or communication stub. 'BAQ-STATUS-CODE' and
* 'BAQ-STATUS-MESSAGE' contain the detailed information
* of this error.
      ELSE
        DISPLAY "Error code: " BAQ-STATUS-CODE
        DISPLAY "Error msg: " BAQ-STATUS-MESSAGE
  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

```

© 2018, 2019 IBM Corporation

Steps to calling an external API

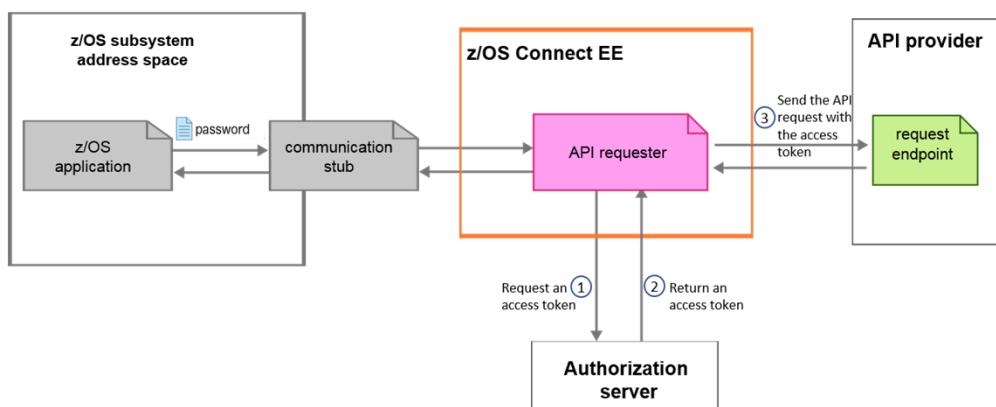
Done



© 2018, 2019 IBM Corporation

API endpoint secured by OAuth 2.0?

Not a problem



More information ibm.biz/zosconnect-oauth2

© 2019 IBM Corporation

63



/common_scenarios

Typical connection patterns to different subsystems.

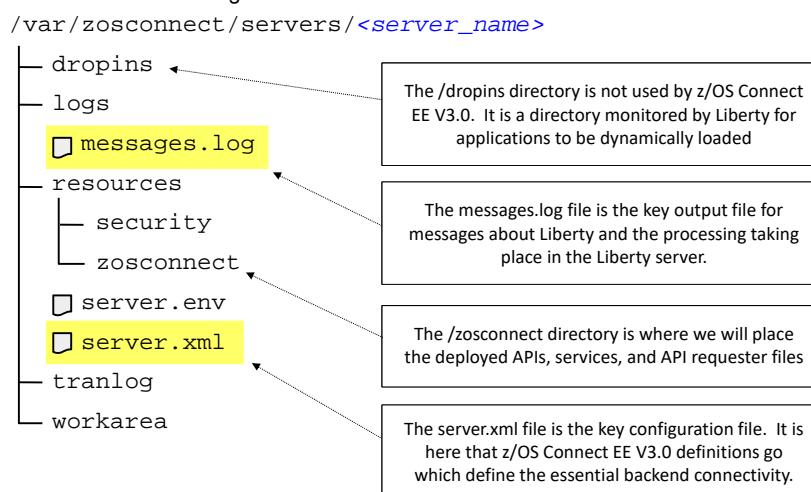
© 2018, 2019 IBM Corporation

Tour of Server Configuration Directories and Files



z/OS Connect EE

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

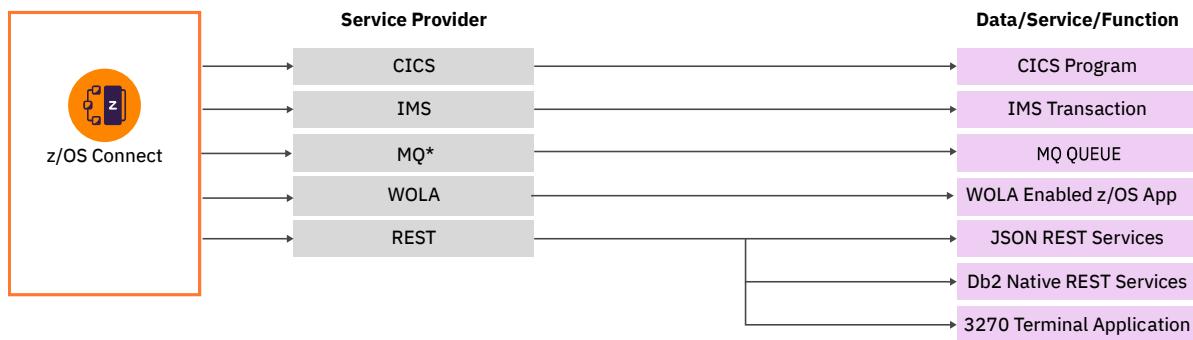


© 2018, 2019 IBM Corporation

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.

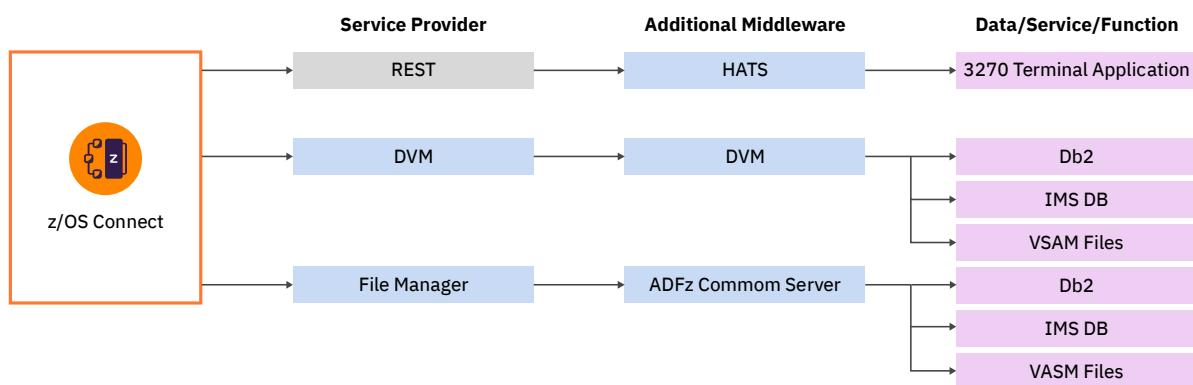
*The MQ service provided is provided by IBM MQ or from Fix Central

© 2019 IBM Corporation

Additional Middleware



Additional value from the ecosystem

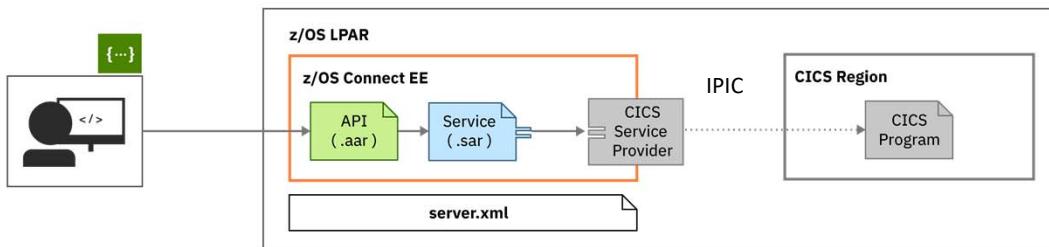


z/OS Connect EE is **pluggable** and **extensible** allowing the use of additional middleware to expand the list of z/OS assets you can expose as APIs

© 2018, 2019 IBM Corporation

Connections to CICS

Topology



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

An IPIC connection must be configured in CICS.

ibm.biz/zosconnect-scenarios

© 2018, 2019 IBM Corporation

CICS IPIC (server.xml)

The `server.xml` file is the key configuration file:

The screenshot shows the IBM Rational Application Developer interface for configuration. On the left, there's a configuration panel for 'inquireSingle Service' with sections for 'Required Configuration' (CCSID: 37, Connection reference: catalog) and 'Optional Configuration' (Transaction ID: catalog, Transaction ID usage: catalog). A callout box points to this panel with the text: "Define IPIC connection to CICS". On the right, the 'catalog.xml' file is displayed in a code editor. The 'Source' tab shows the XML configuration for the IPIC connection. A callout box points to this section with the text: "Features are functional building blocks. When configured here, that function becomes available to the Liberty server". The XML code is as follows:

```

<!-- Enable features -->
<featureManager>
  <feature>zosconnect:cicsService-1.0</feature>
</featureManager>
<zosconnect_cicsIpicConnection id="catalog">
  <host>wg31.washington.ibm.com</host>
  <port>1491</port>
  <transid>CSMI</transid>
  <transidUsage>EIB_AND_MIRROR</transidUsage>
</zosconnect_cicsIpicConnection>

```

Below the XML editor, there's a detailed configuration panel for the 'z/OS Connect CICS IPIC connection'. It includes fields for 'ID' (catalog), 'Host' (wg31.washington.ibm.com), 'Port' (1491), 'Shared port' (false), and 'z/OS Connect APPLID' (no value).

© 2018, 2019 IBM Corporation

The IMS server.xml file ...

Connections to IMS

Topology

```

graph LR
    Client[Client Application] --> ZOSLPAR[z/OS LPAR]
    subgraph ZOSLPAR [z/OS LPAR]
        direction TB
        API[API (.aar)] --> Service[Service (.sar)]
        Service --> Provider[IMS Service Provider]
        Provider -.-> Connect[IMS Connect]
        Connect -.-> IMS[IMS Address Space]
        IMS -.-> Program[IMS program]
        serverxml[server.xml]
    end
    ZOSLPAR -.-> Client

```

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

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>

```

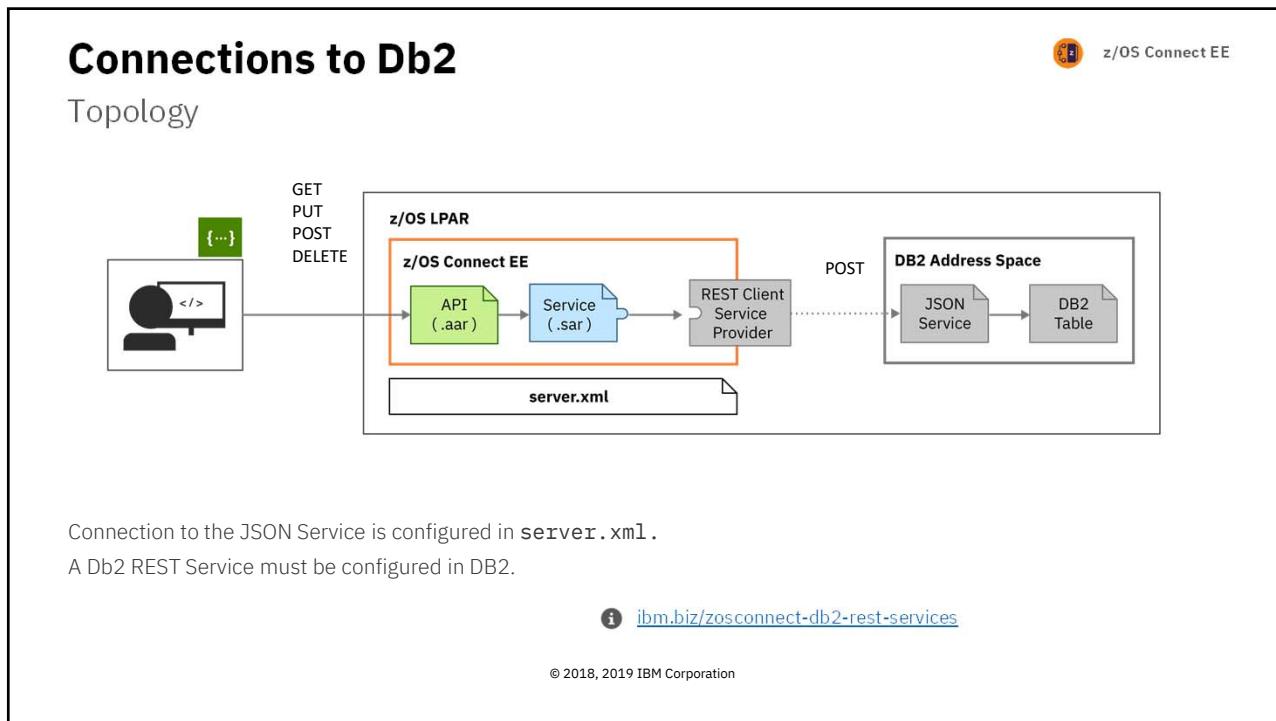
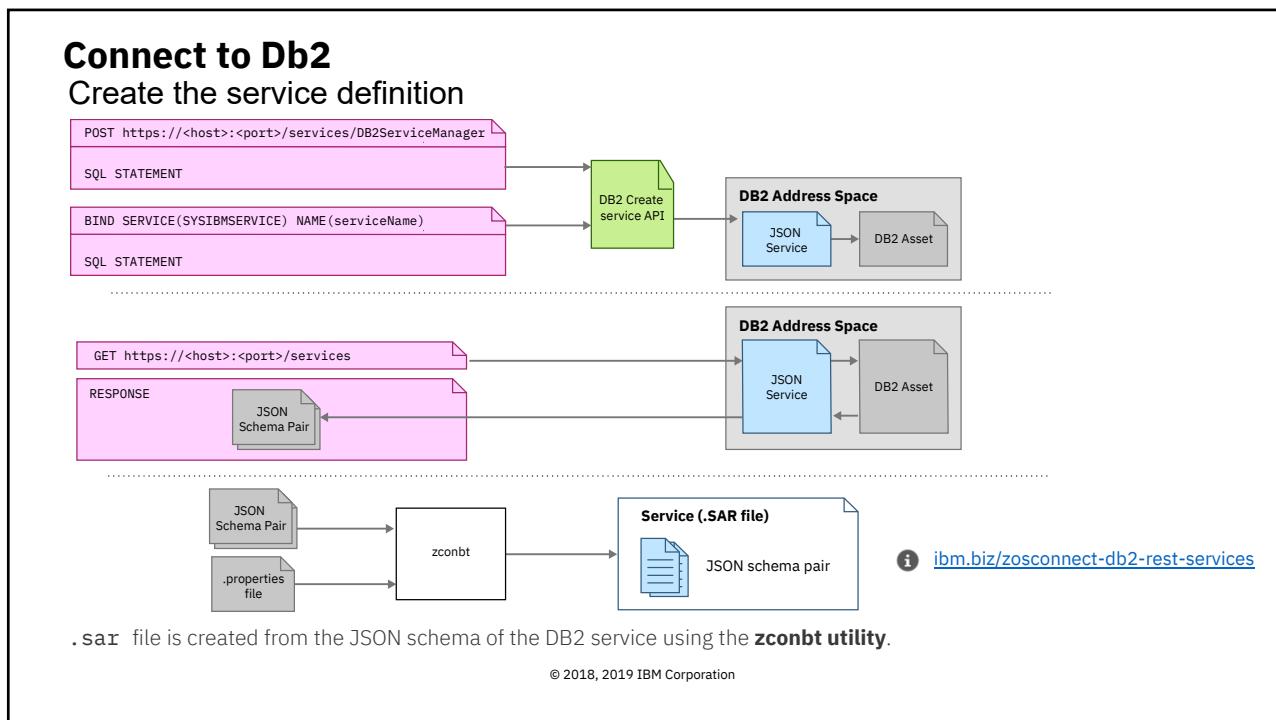
Interaction

```

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

```

© 2018, 2019 IBM Corporation



The server.xml File (Db2)



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

```

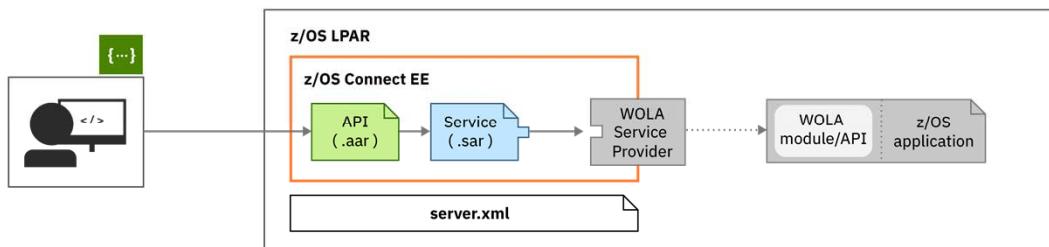
DSNL004I	-DSN2 DDF START
COMPLETE	LOCATION
DSN2LOC	LU
USIBMWZ.DSN2APPL	GENERICLU -NONE
WG31.WASHINGTON.IBM.COM	DOMAIN
TCPPORT 2446	
SECPORT 2445	
RESPORT 2447	

© 2018, 2019 IBM Corporation

Connections to a MVS batch application



Topology



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

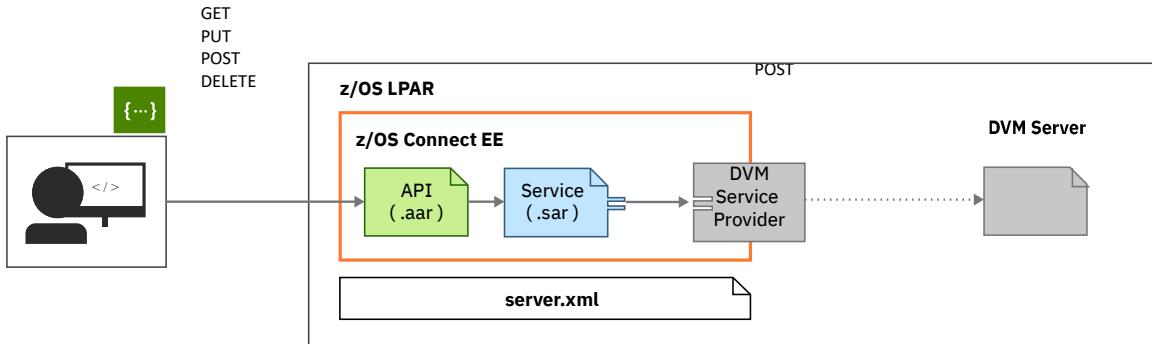
The z/OS application must be WOLA-enabled.

© 2018, 2019 IBM Corporation

Connections to DVM



Topology



The DVM service provider uses WOLA

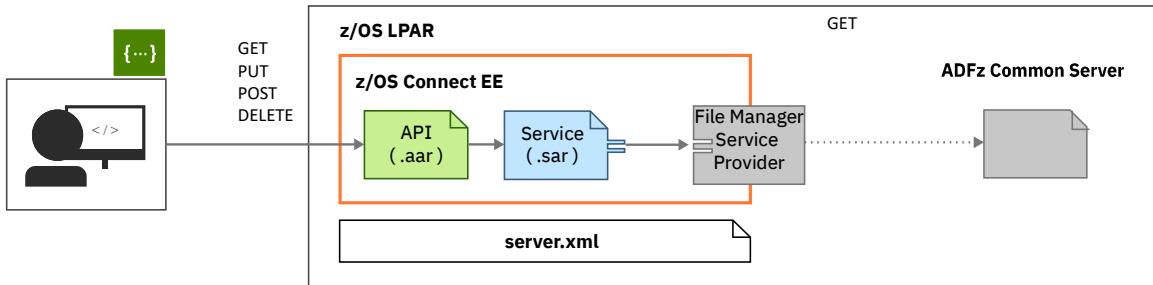
ibm.biz/zosconnect-db2-rest-services

© 2018, 2019 IBM Corporation

Connections to File Manager



Topology



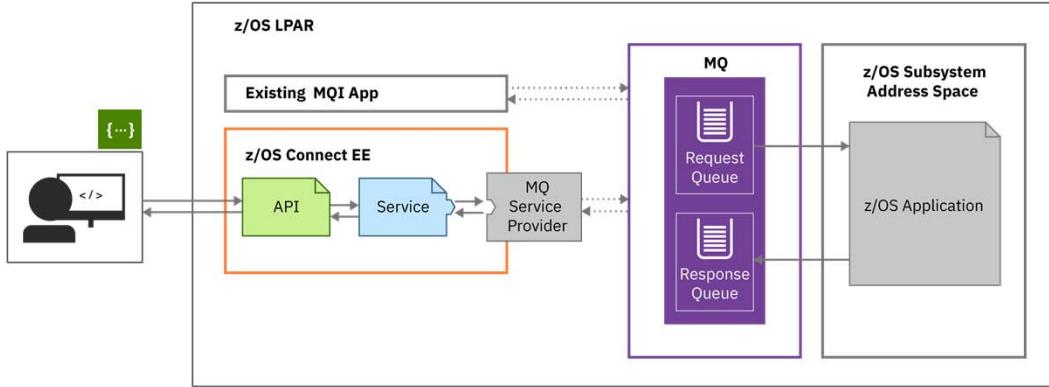
Connection to the Application Delivery Foundation for z (ADFz) common server is over TCP/IP

A File Manager Template is required .

© 2018, 2019 IBM Corporation

Connections to MQ

Topology (Two-way service example)



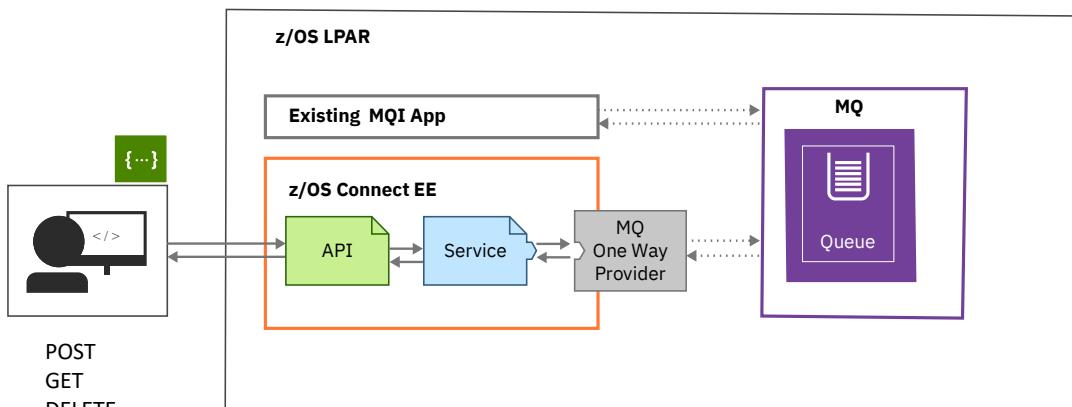
You can also configure one-way services.

ibm.biz/zosconnect-mq-service-provider

© 2018, 2019 IBM Corporation

Connections to MQ

Topology (One-way service example)



ibm.biz/zosconnect-mq-service-provider

© 2018, 2019 IBM Corporation

The server.xml File (MQ)

The screenshot shows the Liberty Admin Center interface with the mq.xml configuration file open. The left pane shows a tree view of the configuration, with the 'JMS Connection Factory' node selected. The right pane displays the details for this node, including its ID ('qmgrCf'), connection manager reference ('ConMgr1'), and JNDI name ('jms/qmgrCf'). A callout box points to the 'JMS Connection Factories' section in the documentation.

JMS Connection Factory
Defines a JMS connection factory configuration.

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.

MQ V9.1.1 Added support for remote queue managers.

© 2018, 2019 IBM Corporation

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

The screenshot shows the Liberty Admin Center interface with the mq.xml configuration file open. The left pane shows a tree view of the configuration, with the 'FileQueue' node selected. The right pane displays the details for this node, including its invokeURI ('/FileQueue'), dataForm ('xformJSON2Byte'), service ('Filequeue'), serviceDescription ('MQ OneWay Service'), and serviceRef ('Filequeue'). Callout boxes point to the 'z/OS Connect service' and 'MQ z/OS Connect service' sections in the documentation.

```

<!-- Native library path -->
<wmqJMSClient nativeLibraryPath= /usr/lpp/mqm/v9r1m1/java/lib />
<!-- ZOS Connect configuration -->
<zosconnect_zosConnectService id="filequeue">
  <invokeURI>/FileQueue</invokeURI>
  <dataFormRef>xformJSON2Byte</dataFormRef>
  <serviceName>Filequeue</serviceName>
  <serviceDescription>MQ OneWay Service</serviceDescription>
  <serviceRef>Filequeue</serviceRef>
</zosconnect_zosConnectService>
<mqzosconnect_mqzOSConnectService id="FileQueue">
  <connectionFactory>jms/qmgrCf</connectionFactory>
  <destination>jms/default</destination>
</mqzosconnect_mqzOSConnectService>
<jmsQueue id="q1" jndiName="jms/default">
  <properties.wmqJMS>
    <baseQueueName>ZCONN2.DEFAULT.MQZCEE.QUEUE</baseQueueName>
    <CCSID>37</CCSID>
  </properties.wmqJMS>
</jmsQueue>

```

© 2018, 2019 IBM Corporation

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

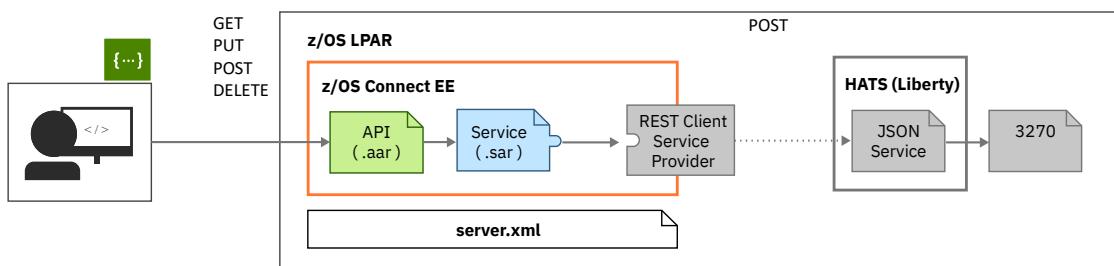
The screenshot shows the Liberty Admin Center interface with the 'Server Config' tab selected. A specific configuration file, 'mq.xml', is open in 'Design' mode. The code editor displays XML configuration for a 'zosconnect_zosConnectService' and 'mqzosconnect_mqzOSConnectService'. Annotations point from the XML code to various components in a diagram on the right:

- zosconnect_zosConnectService id="miniloan"**: Points to the 'z/OS Connect service' box.
- mqzosconnect_mqzOSConnectService id="Miniloan"**: Points to the 'MQ z/OS Connect service' box.
- <jmsQueue id="request" jndiName="jms/request">**: Points to the 'JMS Destinations (queue)' box.
- <jmsQueue id="response" jndiName="jms/response">**: Points to the 'JMS Destinations (queue)' box.

© 2018, 2019 IBM Corporation

Connection to HATS

Topology



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

ibm.biz/zosconnect-db2-rest-services

© 2018, 2019 IBM Corporation



/miscellaneousTopics

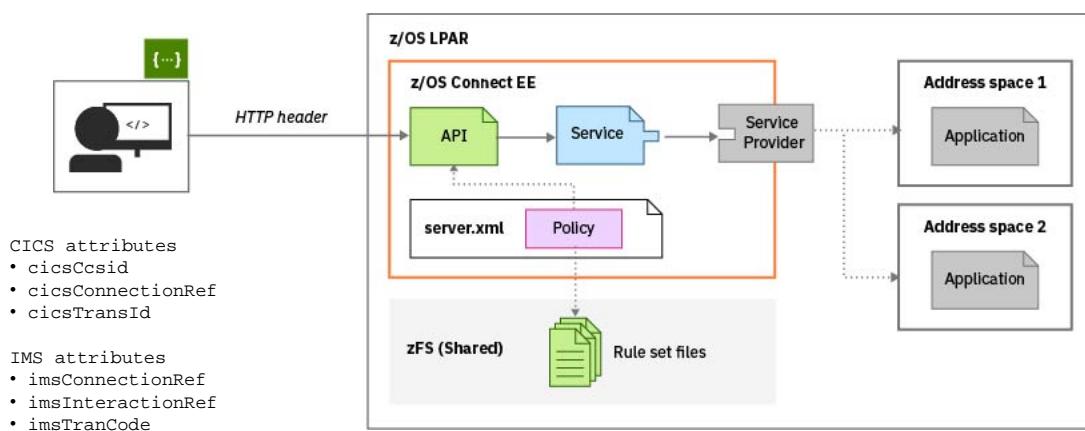
performance, high availability, Liberty

© 2018, 2019 IBM Corporation

API Policies

z/OS Connect EE

- 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)



© 2018, 2019 IBM Corporation

Liberty's “adminCenter” Feature



Web browser interface to the server's configuration files

The screenshot shows the 'Server Config' interface for 'server.xml'. The left sidebar lists various configuration sections: z/OS Connect Manager, z/OS Connect Data Transformer, WOLA, Connection Factory (wolaCF), Feature Manager, HTTP Endpoint (defaultHttpEndpoint), Cross-Origin Resource Sharing (default...), Configuration Management, z/OS Connect Manager (selected), z/OS Connect APIs, and z/OS Connect Services.

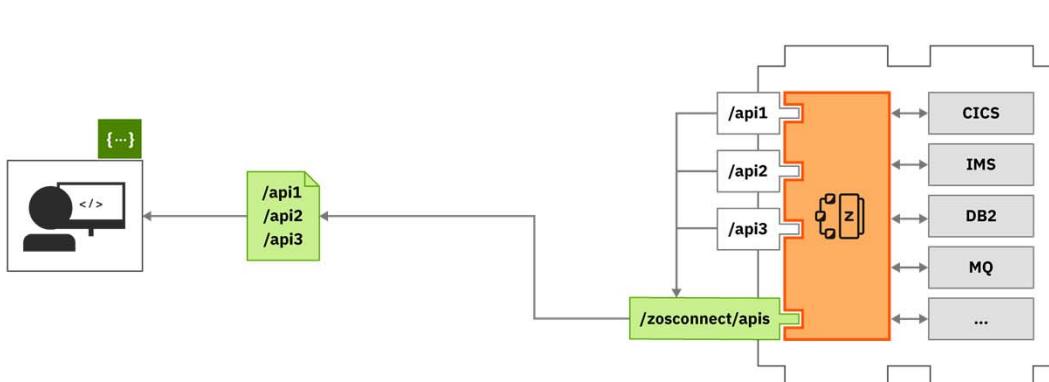
The main panel displays the 'Global asynchronous request timeout' configuration, which is currently set to '(no value)'. A note explains that this timeout specifies the time in milliseconds for asynchronous requests. It also mentions that if 'asyncTimeoutDefault' is not configured, the timeout used is the 'syncTimeoutDefault' attribute's default value (e.g., 30 seconds). If 'asyncRequestTimeout' is configured, but the 'syncTimeoutDefault' attribute is, the 'asyncRequestTimeout's configured value is used.

Below this, there are sections for 'Global administrative group' (set to 'GADMIN'), 'Global operations group' (set to 'GOPER'), and 'Global invoke group' (set to 'GINVOKE').

At the bottom of the interface, there are 'Save' and 'Close' buttons.

© 2018, 2019 IBM Corporation

API Documentation



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

© 2018, 2019 IBM Corporation

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
PUT   /zosConnect/services inquireSingle.sar
GET   /zosConnect/services
GET   /zosConnect/services/{serviceName}
DELETE /zosConnect/services/{serviceName}

```

© 2018, 2019 IBM Corporation

*Useful for deploying DB2 and HATS
service archive files

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



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

Deploying Db2 Service Archive Options



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

Command:

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

Results:

```
{
  "zosConnect": {
    "serviceName": "selectEmployee",
    "serviceDescription": "Select a row from USER1_EMPLOYEE",
    "serviceProvider": "IBM_ZOS_CONNECT_SERVICE_REST_CLIENT-1.0",
    "serviceInvokeURL": "https://mpxm:9453/zosConnect/services/selectEmployee",
    "serviceInvokeURL": "http://wsg1.washington.ibm.com:9453/zosConnect/services/selectEmployee",
    "dataXformProvider": "DATA_UNAVAILABLE",
    "receiveTimeout": "0",
    "receiveTimeout": "0",
    "receiveTimeout": "0",
    "port": null,
    "method": "POST",
    "httpMethod": "POST",
    "connectionTimeout": "0",
    "url": "/zosConnect/services/selectEmployee"
  }
}
```

© 2018, 2019 IBM Corporation

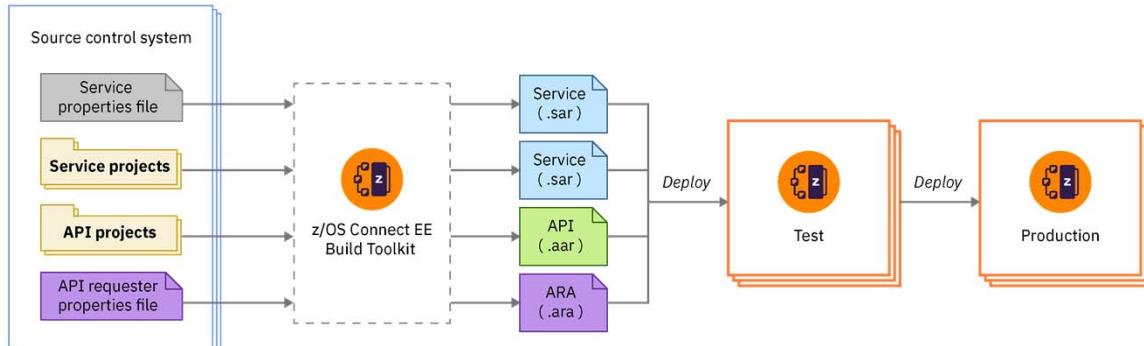
91

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

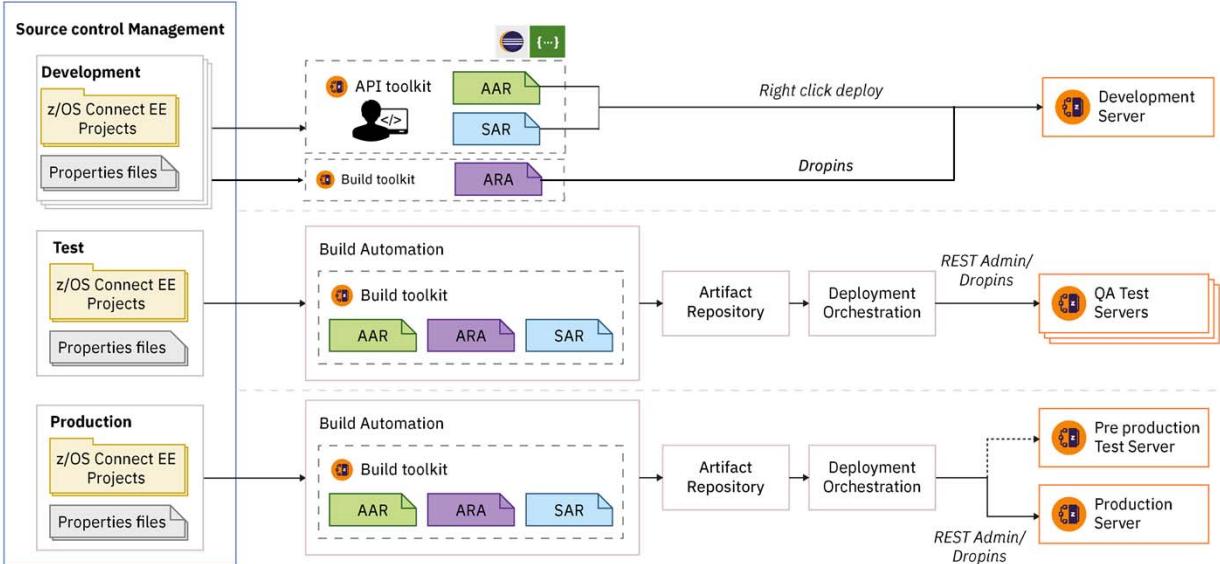


© 2019 IBM Corporation

ibm.biz/zosconnect-devops

92

DevOps Pipeline using z/OS Connect EE



© 2019 IBM Corporation

ibm.biz/zosconnect-devops

93

z/OS Connect EE integration with Zowe

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

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

© 2019 IBM Corporation [Blogpost: Expose z/OS Connect EE APIs in Zowe API mediation layer](#) 94

[developer.ibm.com/mainframe/2019/04/01/expose-z-os-connect-ee-apis-in-zowe-api-mediation-layer](#)

Performance: API Provider

High Speed, High Throughput, Low Cost

CPU Cost Per Transaction - increasing number of clients with 50 byte requests and 1K to 128K responses, using channels and CICS SP

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

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

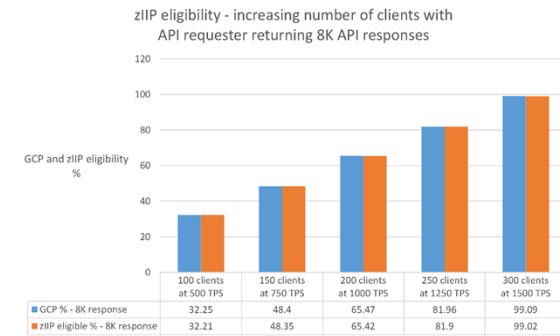
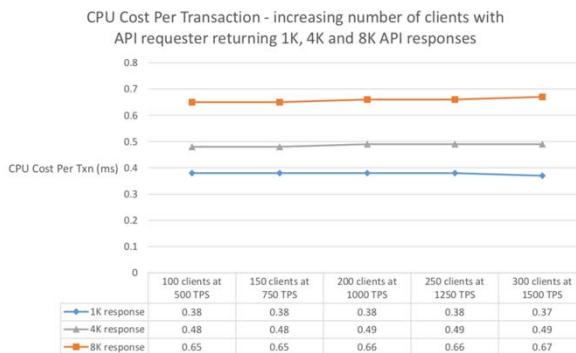
Clients	GCP %	ZIIP eligible %
100 clients at 200 TPS	77.71	77.67
200 clients at 400 TPS	156.14	156.1
300 clients at 600 TPS	235.12	235.06
400 clients at 800 TPS	314.57	314.5
500 clients at 1000 TPS	393.63	393.58

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

© 2019 IBM Corporation [ibm.biz/zosconnect-performance-report](#) 95

Performance: API Requester

High Speed, High Throughput, Low Cost



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

© 2019 IBM Corporation

ibm.biz/zosconnect-performance-report

96

IBM z Omegamon for JVM



The screenshots illustrate the following features:

- z/OS Connect Request Summary:** Shows a summary of requests over a specified time range (e.g., Last 30 Minutes, Last 1 Hour, Date/Time Range). It includes columns for API Name, Service, SoR ID, Reference, and Resource.
- Requests by Service Name:** A detailed view of requests grouped by service name. It shows metrics like API Name, Service, SoR ID, Reference, Resource, and various performance counters (e.g., Requests, Errors, Timeout, Keep Time, Avg).
- Event Timeline:** A log of events with columns for Event Type, Provider, User ID, and Query String.
- Summary Statistics:** A final summary table with columns for Rows, API Name, Method, Requests, Errors, Timeout, and Keep Time.

© 2018, 2019 IBM Corporation

IBM z Omegamon for JVM

The screenshots show the 'z/OS Connect Request Detail' window from the IBM z Omegamon for JVM tool. Each window displays detailed information about a specific API request, including:

- Request Type:** API
- Request URI:** /filequeue/mq, /cscvinc/employee/44444, or /phonebook/contacts/LASTI
- Method:** GET
- HTTP code:** 200 (OK) for all requests
- Timeout:** No
- Service Name:** filequeue, cscvincService, or phonebookService
- Total Resp Time:** 0.016206ms, 0.000006ms, or 0.000005ms
- z/OS Conn Time:** 0.016206ms, 0.000006ms, or 0.000005ms
- User ID:** KJ3ZCDD, KJ3ZCDD, or Fred
- Correlation ID:** 10d5ea51, 10d5ea50, or 10d5ea55
- Operation:** getFile, getEmployee, or getSelectedEmployee
- Provider:** IBM MQ for z/OS, CICS, or CICS
- Response Length:** 191, 302, or 0
- Correlation ID:** 10d5ea51, 10d5ea50, or 10d5ea55
- User ID:** KJ3ZCDD, KJ3ZCDD, or Fred

© 2018, 2019 IBM Corporation

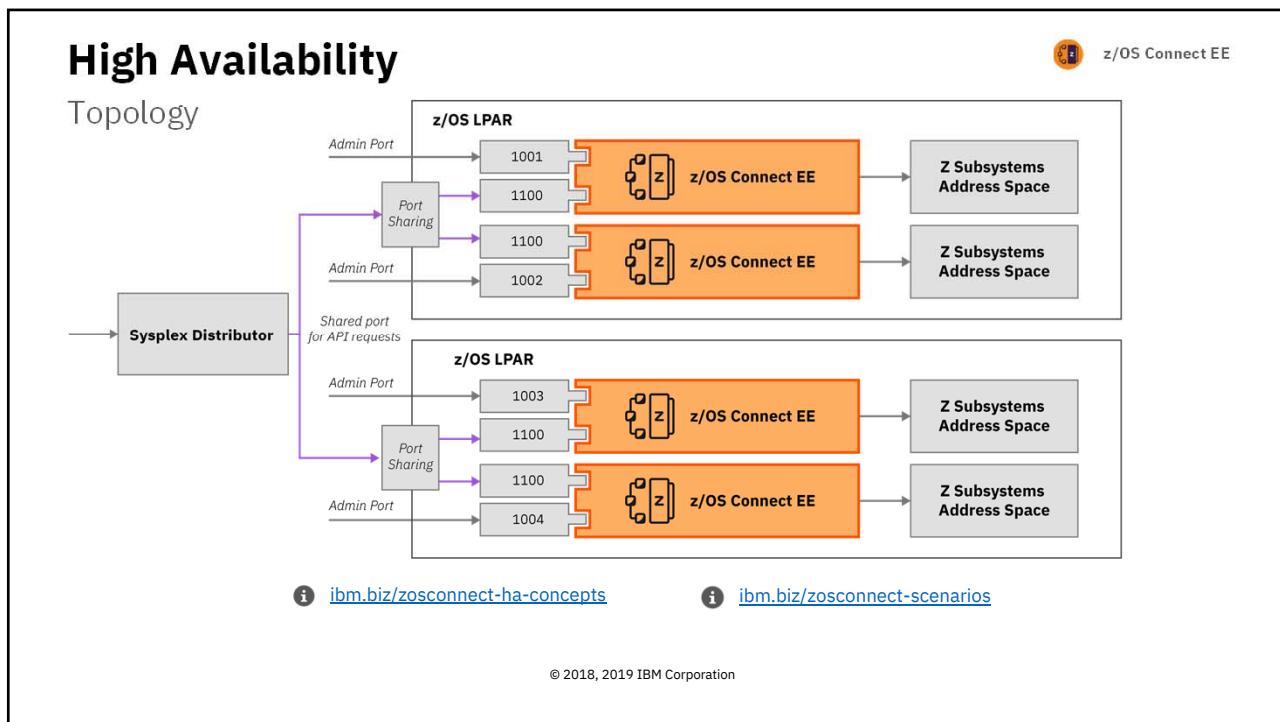
Authentication

z/OS Connect EE

Several different ways this can be accomplished:

Basic Authentication	Client Certificate	Third Party Authentication
<p>Liberty z/OS</p> <p>ID/PW</p> <p>REST Client</p> <p>Okay!</p>	<p>Liberty z/OS</p> <p>TLS Client Cert</p> <p>REST Client</p> <p>Okay!</p> <p>Could be a trusted server</p>	<p>Liberty z/OS</p> <p>Identity Mapping Green circle = 'FRED'</p> <p>Cert</p> <p>Auth</p> <p>3rd Party</p> <p>Token (JWT, LTPA, other)</p> <p>REST Client</p> <p>Okay!</p> <p>Trusted Server</p> <p>ID/PW</p>
Server prompts for ID/PW Client supplies ID/PW Server checks registry: <ul style="list-style-type: none"> • Basic (server.xml) • LDAP • SAF 	Server prompts for cert. Client supplies certificate Server validates cert and maps to an identity Registry options: <ul style="list-style-type: none"> • LDAP • SAF 	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: <ul style="list-style-type: none"> • LDAP • SAF

© 2018, 2019 IBM Corporation





/exercises

basic security; exercise paths

© 2018, 2019 IBM Corporation

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)
- Copy/Paste files on desktop
 - Basic Configuration CopyPaste
 - Developing APIs CopyPaste
- Identities:
 - RACF identity: USER1-> Password: USER1
 - zCEE identity: Fred → Password: fredpwd
- Or one or more of the following:**
- ❑ Developing APIs Hands-on Labs
 - ❑ CICS Container/COMMAREA
 - ❑ DB2
 - ❑ IMS Transaction
 - ❑ MQ
 - ❑ MVS Batch
 - ❑ Outbound RESTful applications
- 3270 Key Sequences
 - Clear screen: Fn-P
 - Enter key: right CTRL
- Material can be downloaded from:
<http://tinyurl.com/y28fsezs>
- z/OS Connect EE Users Group
<https://www.linkedin.com/groups/8731382/>

© 2018, 2019 IBM Corporation