



IBM z/OS Connect Enterprise Edition

Introduction and Overview

Mitch Johnson

mitchj@us.ibm.com

Washington System Center

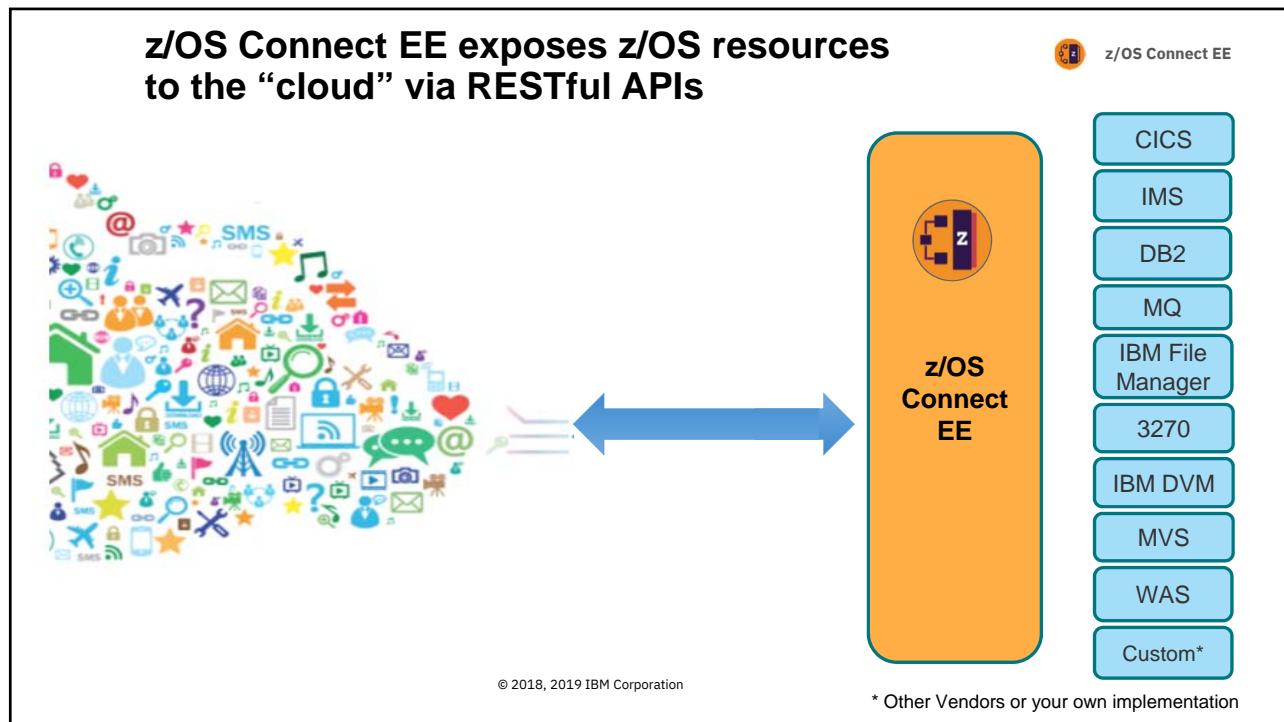


© 2018, 2019 IBM Corporation

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”?

REST is an Architectural Style

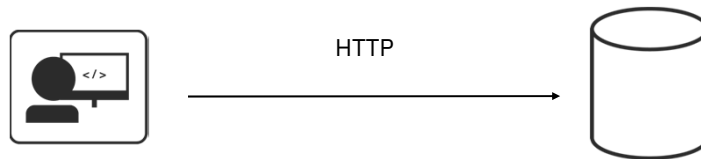


REST stands for **R**epresentational **S**tate **T**ransfer.

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

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

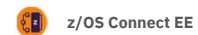
Simple and intuitive for the end consumer (**the developer**).



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.

© 2018,2019 IBM Corporation

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)

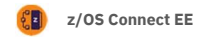
Request/Response Body is used to represent the data object

```

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" }
  
```

© 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

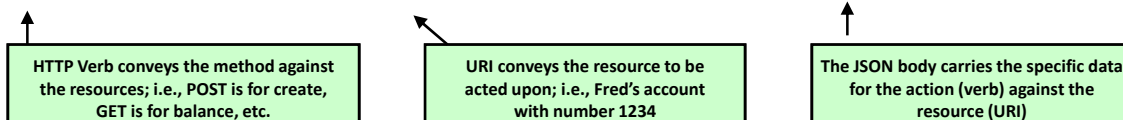


z/OS Connect Enterprise Edition:

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

GET /account?number=1234

PUT /account?number=1234 +  (JSON with dollar amount of deposit)



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



(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?



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



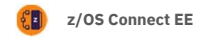
/swagger/open_api

The industry standard framework for describing RESTful APIs.

© 2018, 2019 IBM Corporation

Why use Swagger?

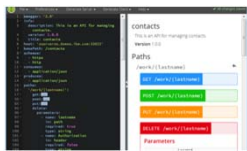
It is more than just an API framework



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/>

© 2018, 2019 IBM Corporation

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

13

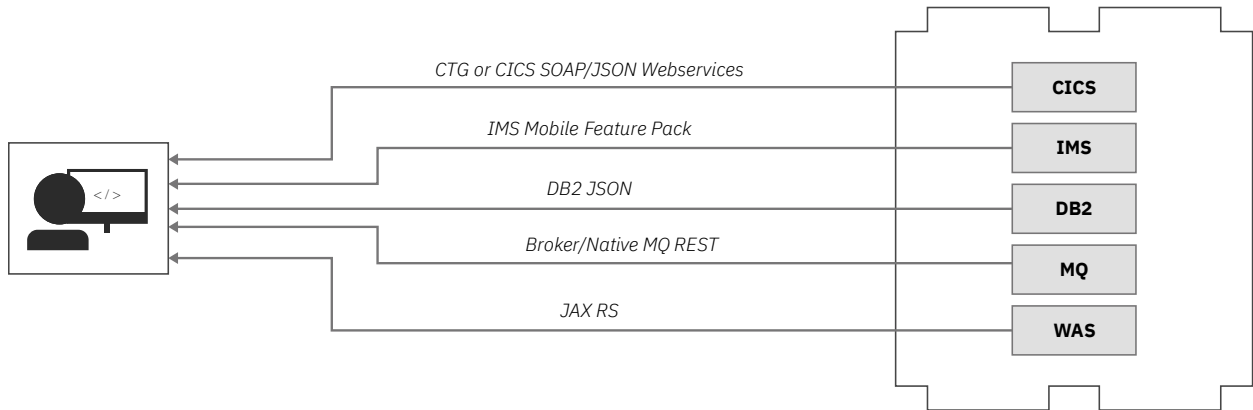
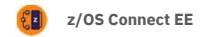


Why /zos_connect_ee?

Truly RESTful APIs to and from your mainframe.

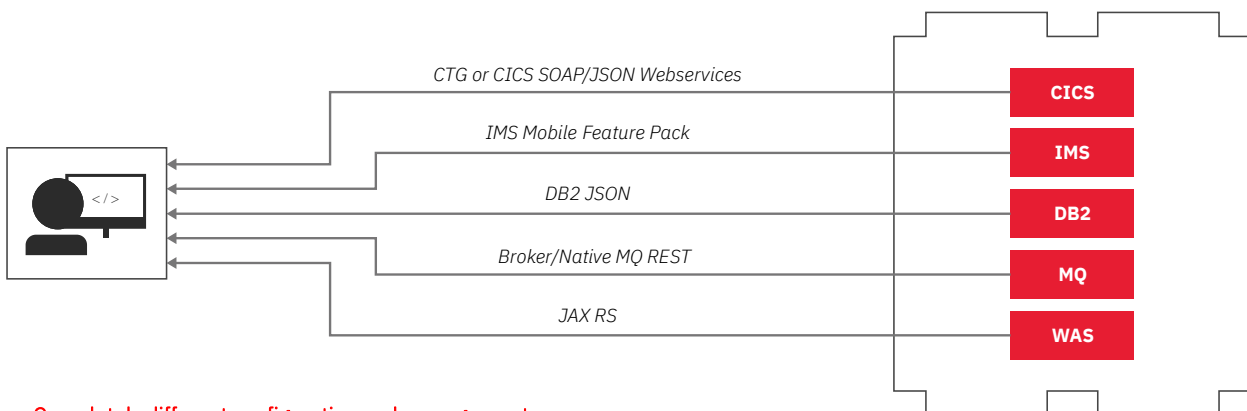
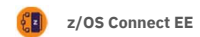
© 2018 , 2019 IBM Corporation

First, can't we already do REST and JSON today?



© 2018, 2019 IBM Corporation

Yes, but....



Completely different configuration and management.

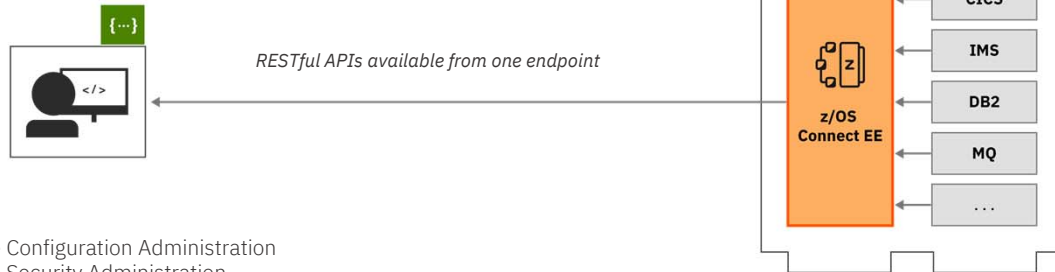
Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!

© 2018, 2019 IBM Corporation

A single entry point is needed

Expose z/OS resources without writing any code.



- ❑ 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



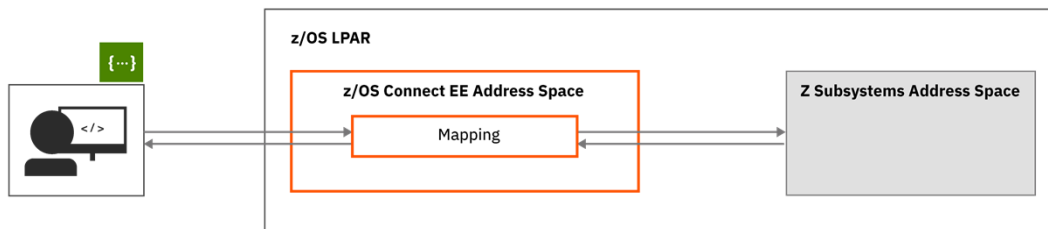
**Other than a RESTful interface,
what does z/OS Connect provide?**

© 2018, 2019 IBM Corporation

Let's Start with Data mapping

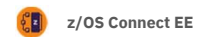


Converting from JSON to the target's subsystem's format

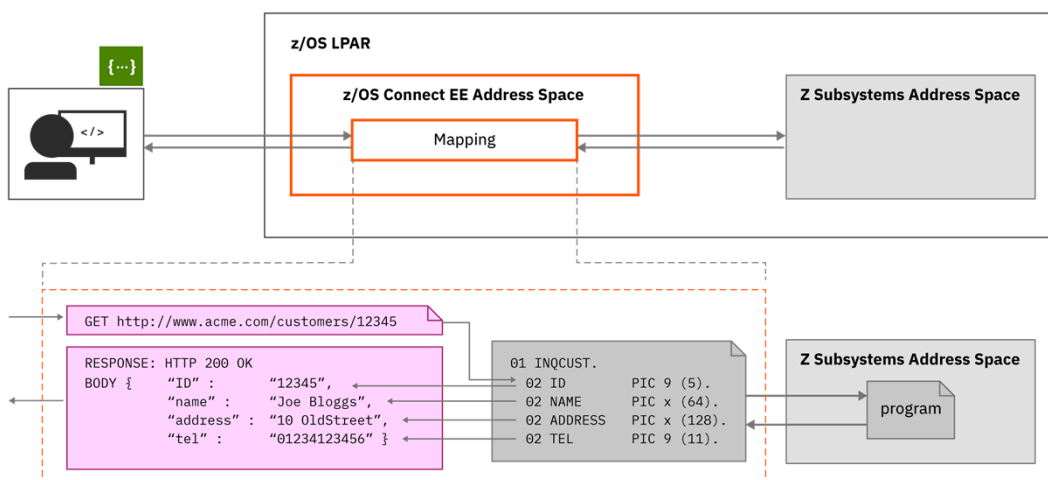


© 2018, 2019 IBM Corporation

Data mapping Example

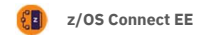


A closer look



© 2018, 2019 IBM Corporation

COBOL versus JSON Example



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

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

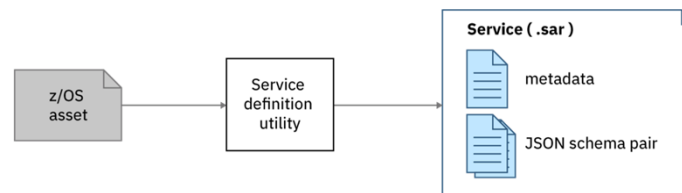
© 2018, 2019 IBM Corporation

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)
- z/OS Connect EE Build Toolkit (Db2, MQ, MVS Batch, IBM File Manager and HATS)
- DVM Toolkit

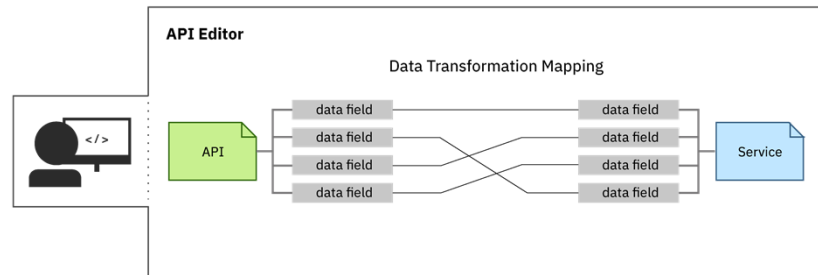
ibm.biz/zosconnect-sar-creation

© 2018,2019 IBM Corporation

Steps to expose a z/OS application



2. Create an API



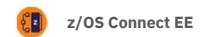
Import your `.sar` file into the **API toolkit**, and start designing your API.

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

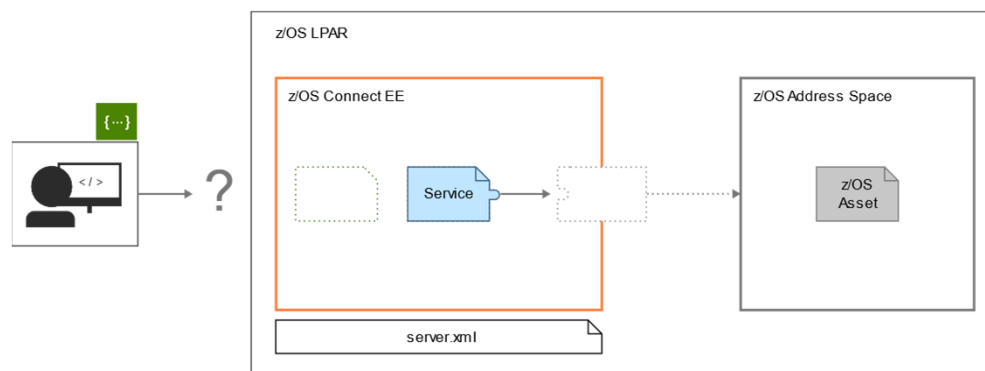
ibm.biz/zosconnect-create-api

© 2018,2019 IBM Corporation

Steps to expose a z/OS application



3. Deploy your service

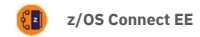


Deploy the `.sar` file generated in **Step 2** using the right-click deploy in the **API toolkit**, or by copying the `.sar` file to the services directory.

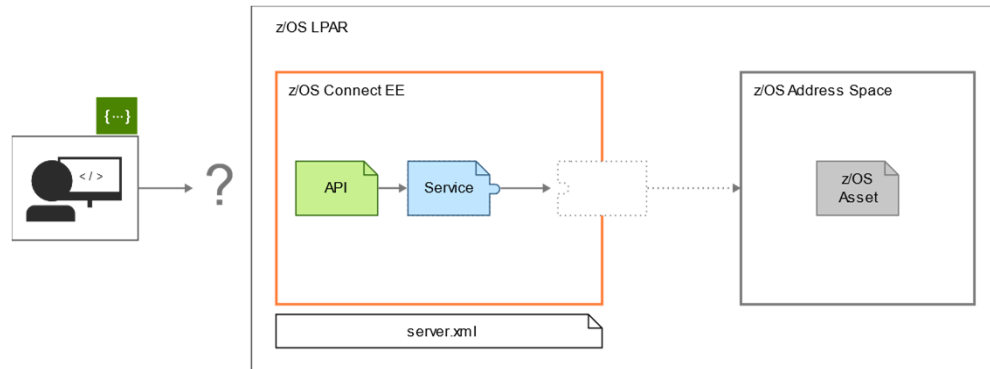
ibm.biz/zosconnect-define-services

© 2018, 2019 IBM Corporation

Steps to expose a z/OS application



4. Deploy your API

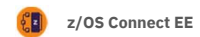


Deploy your API using the right-click deploy in **the API toolkit**, or by copying the `.aar` file to the `apis` directory.

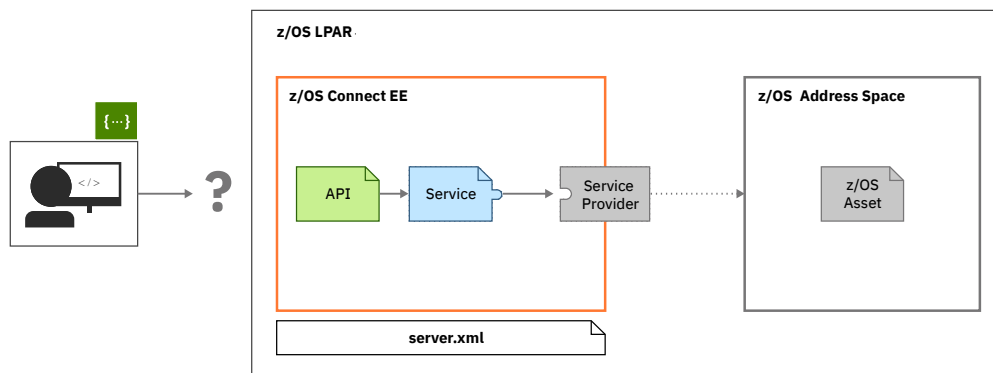
ibm.biz/zosconnect-deploy-api

© 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`.

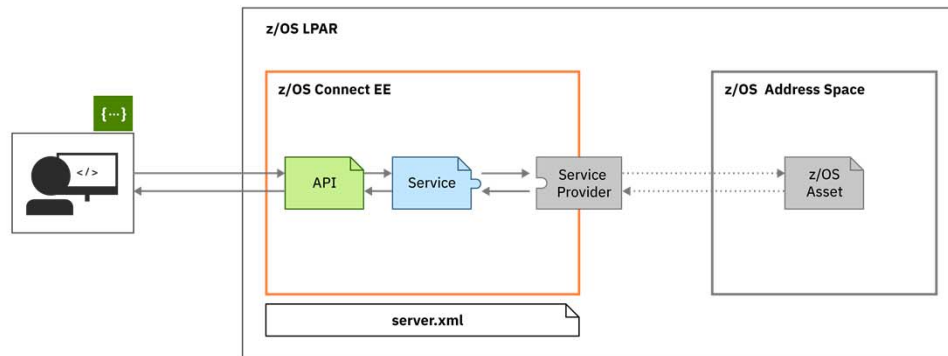
ibm.biz/zosconnect-configuring

© 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

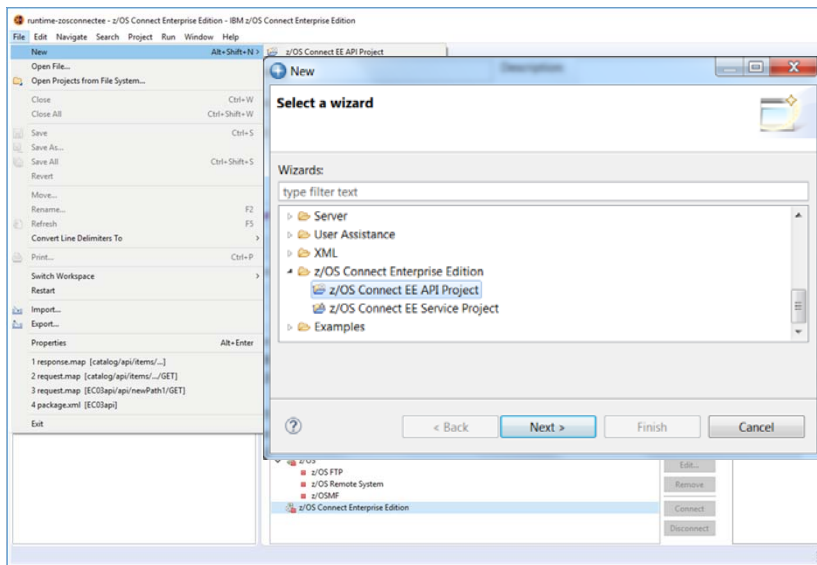
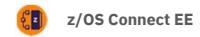


/api_toolkit/services

Simple **service** creation.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS

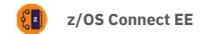


Use the **API toolkit** to create services through Eclipse-based tooling.

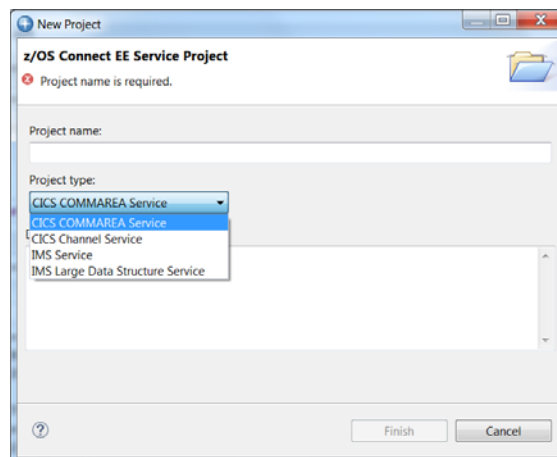
Services are described as **Projects**, so They can be easily managed in source control.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



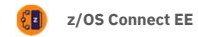
Service creation – a common interface



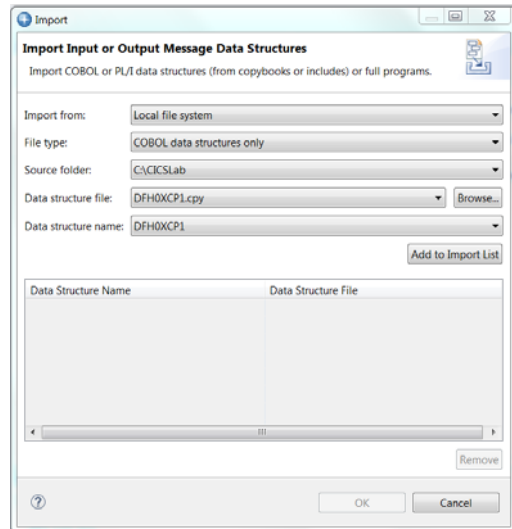
A common interface for service creation, agnostic of back end subsystem.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



Creating a service project

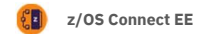


© 2018, 2019 IBM Corporation

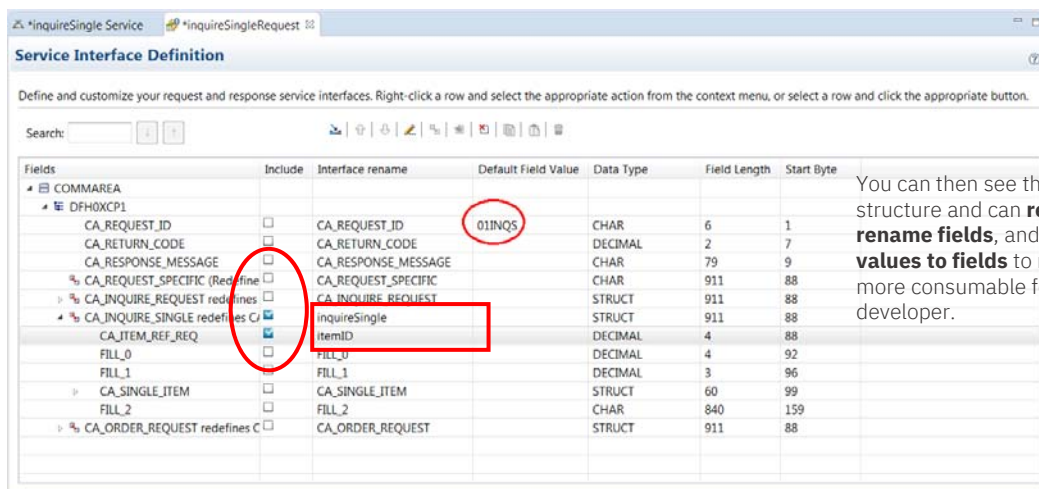
You start by importing data structures into the service interface from the local file system or the workspace.

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

API toolkit – Creating Services for CICS and IMS



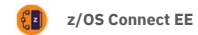
Creating a service interface definition



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

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.

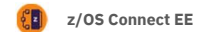
Search:

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH00CP1						
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 checked="" type="checkbox"/>	InquireSingle		STRUCT	911	88
CA_ITEM_REF_REQ	<input checked="" 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_SINGL_ITEM_REF	<input checked="" type="checkbox"/>	itemReference		DECIMAL	4	99
CA_SINGL_DESCRIPTION	<input checked="" type="checkbox"/>	description		CHAR	40	103
CA_SINGL_DEPARTMENT	<input checked="" type="checkbox"/>	department		DECIMAL	3	143
CA_SINGL_COST	<input checked="" type="checkbox"/>	cost		CHAR	6	146
IN_SINGL_STOCK	<input checked="" type="checkbox"/>	inStock		DECIMAL	4	152
ON_SINGL_ORDER	<input checked="" type="checkbox"/>	onOrder		DECIMAL	3	156
FILL_2	<input type="checkbox"/>	FILL_2		CHAR	840	159
% CA_ORDER_REQUEST redefines C	<input checked="" 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**

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



Creating a service – request messages

Service Interface Editor

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

Search:

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
Channel					
Container1		CSCCINContainer			
REQUEST_CONTAINER	<input type="checkbox"/>	REQUEST_CONTAINER			
ACTION	<input type="checkbox"/>	ACTION			
USERID	<input type="checkbox"/>	USERID			
FILE_AREA	<input checked="" type="checkbox"/>	FILE_AREA			
STAT	<input type="checkbox"/>	STAT			
NUMB	<input checked="" type="checkbox"/>	NUMB			
NAME	<input type="checkbox"/>	NAME			
ADDRX	<input type="checkbox"/>	ADDRX			
PHONE	<input type="checkbox"/>	PHONE			
DATEX	<input type="checkbox"/>	DATEX			
AMOUNT	<input type="checkbox"/>	AMOUNT			
COMMENT	<input type="checkbox"/>	COMMENT			

Service Interface Editor

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

Search:

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
Channel					
Container1		cscvincinContainer			
REQUEST_CONTAINER	<input type="checkbox"/>	REQUEST_CONTAINER			
ACTION	<input type="checkbox"/>	ACTION		CHAR	1
USERID	<input type="checkbox"/>	USERID		CHAR	8
FILE_AREA	<input checked="" type="checkbox"/>	FILE_AREA		STRUCT	80
STAT	<input checked="" type="checkbox"/>	status		CHAR	1
NUMB	<input checked="" type="checkbox"/>	employeeNumber		CHAR	6
NAME	<input checked="" type="checkbox"/>	employeeName		CHAR	20
ADDRX	<input checked="" type="checkbox"/>	address		CHAR	20
PHONE	<input checked="" type="checkbox"/>	phoneNumber		CHAR	8
DATEX	<input checked="" type="checkbox"/>	startDate		CHAR	8
AMOUNT	<input checked="" type="checkbox"/>	amount		CHAR	8
COMMENT	<input checked="" type="checkbox"/>	comment		CHAR	9

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



Creating a service interface definition

Service Interface Editor

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

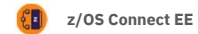
Search:

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length	Start
COMMAREA						
MINILOAN_COMMAREA						
NAME	<input checked="" type="checkbox"/>	loan application		CHAR	20	1
CREDITSCORE	<input checked="" type="checkbox"/>	name		DECIMAL	18	21
YEARLYINCOME	<input checked="" type="checkbox"/>	credit score		DECIMAL	18	39
AGE	<input checked="" type="checkbox"/>	yearly income		DECIMAL	10	57
AMOUNT	<input checked="" type="checkbox"/>	age		DECIMAL	18	67
APPROVED	<input checked="" type="checkbox"/>	loan amount		CHAR	1	85
EFFECTDATE	<input checked="" type="checkbox"/>	approved?		CHAR	8	86
YEARLYINTERESTRATE	<input checked="" type="checkbox"/>	effective date		DECIMAL	5	94
YEARLYREPAYMENT	<input checked="" type="checkbox"/>	yearly interest rate	00005	DECIMAL	18	99
UID	<input checked="" type="checkbox"/>	yearly payment		CHAR	8	117
MESSAGES_NUM	<input checked="" type="checkbox"/>	authorization identity		DECIMAL	9	125
MESSAGES (1..99)	<input checked="" type="checkbox"/>	MESSAGES_NUM		ARRAY	9940	134
	<input checked="" type="checkbox"/>	disapproval message				

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 “GET” service interface request definition

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:

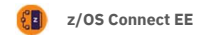
Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
ivtnoDisplayRequest						
Segment 1						
INPUT_MSG						
IN_LL	<input type="checkbox"/>	IN_LL		SHORT	2	1
IN_ZZ	<input type="checkbox"/>	IN_ZZ		SHORT	2	3
IN_TRANCODE	<input type="checkbox"/>	IN_TRANCODE		CHAR	10	5
IN_COMMAND	<input checked="" 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

The service developer creates distinct services for each function.

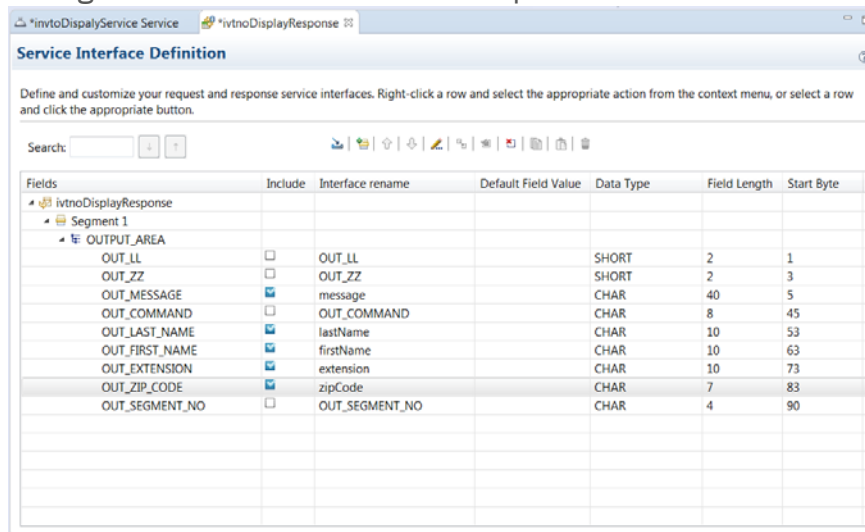
DISPLAY
DELETE
ADD
UPDATE

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



Creating a “GET” service interface response definition

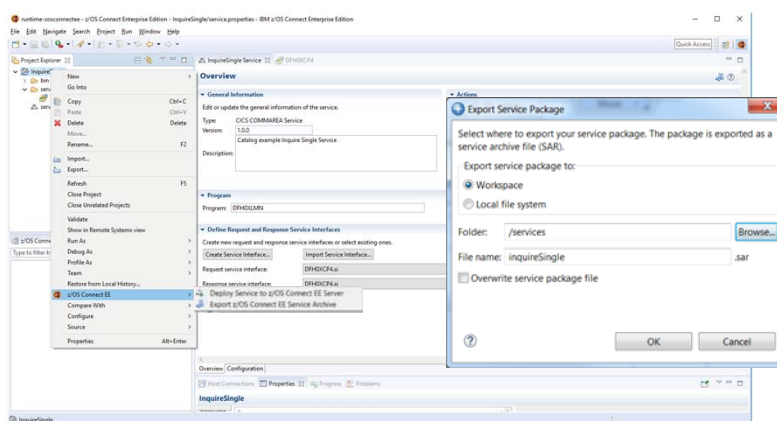


© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



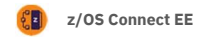
Creating a service for CICS and IMS



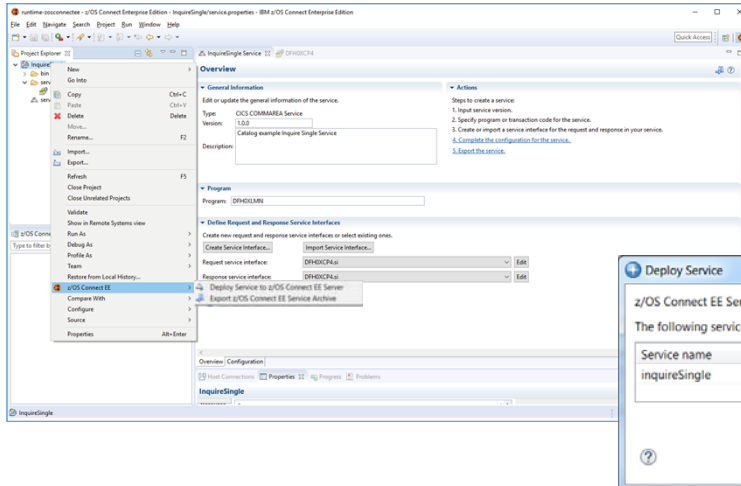
Finally, you can export the service project as a **Service Archive file (.sar)**.

© 2018, 2019 IBM Corporation

API toolkit – Creating Services for CICS and IMS



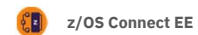
Creating a service for CICS and IMS



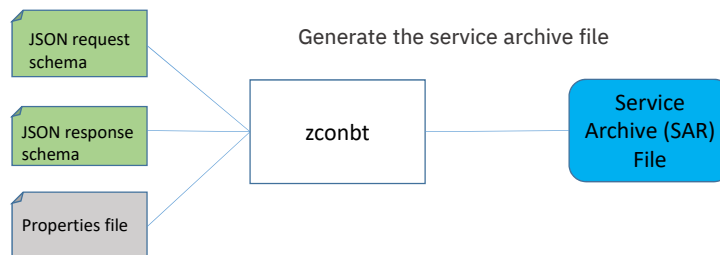
Finally, you can deploy the service project as a **Service Archive file (.sar)**

© 2018, 2019 IBM Corporation

Creating Services without the Toolkit – REST



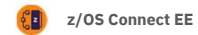
For DB2 and HATS REST Services use the z/OS Connect Build toolkit (zconbt)



```
provider=rest
name=selectEmployee
version=1.0
description=Select a row from USER1.EMPLOYEE
requestSchemaFile=selectEmployeeRequest.json
responseSchemaFile=selectEmployeeResponse.json
verb=POST
uri=/services/selectEmployee
connectionRef=Db2Conn
```

© 2018, 2019 IBM Corporation

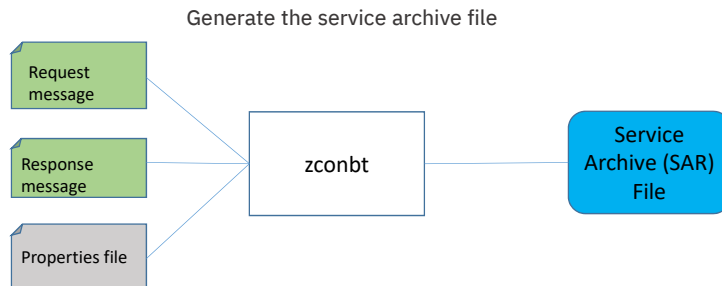
Creating Services without the Toolkit – MQ



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

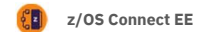
```
provider=mq
name=mqPut
version=1.0
description=MQ put one-way
destination=jms/default
connectionFactory=jms/qmgrCf
language=COBOL
requestStructure=FILEAMQ.CPY
operationName=mqPut
messagingAction=mqput/mqget
```

```
provider=mq
name=miniloan
version=1.0
description=MQ two-way
destination=jms/request
replyDestination=jms/response
connectionFactory=jms/qmgrCf
language=COBOL
requestStructure=miniloan.cpy
responseStructure=miniloan.cpy
operationName=miniloan
waitInterval=10
```



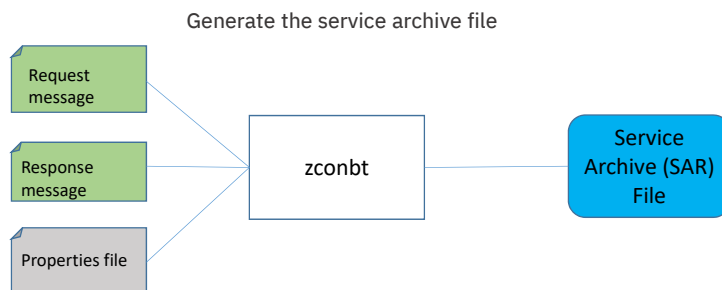
© 2018, 2019 IBM Corporation

Creating Services without the Toolkit – Batch



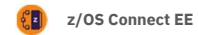
For MQ Services use the z/OS Connect Build toolkit (zconbt)

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



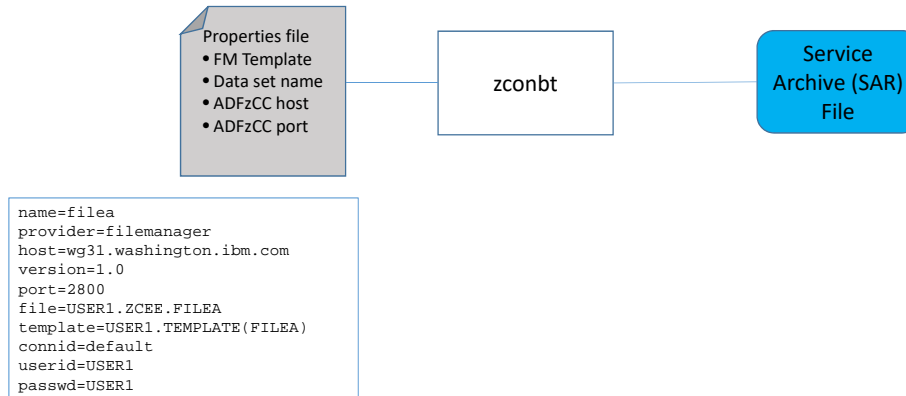
© 2018, 2019 IBM Corporation

Creating Services without the Toolkit – FM



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

Generate the service archive file

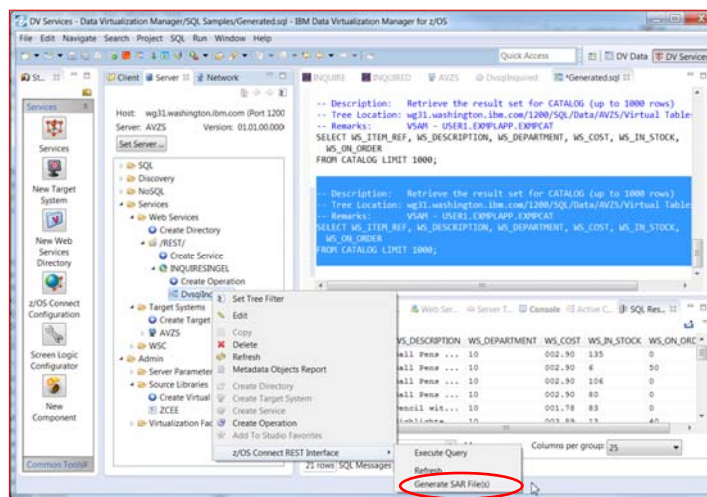


© 2018, 2019 IBM Corporation

Creating Services without the Toolkit



For DVM use the DVM Studio



© 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

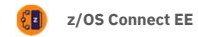


/api_toolkit/api_editor

Quick and easy **API mapping**.

© 2018, 2019 IBM Corporation

API toolkit – API Editor



API definition

The screenshot shows the 'z/OS Connect EE API Editor' window. It has a tab for 'catalog API'. Under 'Describe your API', there are fields for Name (catalog), Base path (/catalogManager), Version (1.0.0), and Description (APIs for browsing, inquiring and ordering items from a catalog). Below this, there are two sections for defining API methods. The first section has a Path (/items?startItem) and two methods: GET (InquireCatalog) and POST (placeOrder). The second section has a Path (/items/{itemID}) and three methods: GET (InquireSingle), PUT (InquireSingle), and DELETE (InquireSingle). Each method has buttons for 'Service...' and 'Mapping...'.

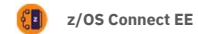
The **API toolkit** is designed to encourage RESTful API design.

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 mapping' interface for the 'GET .item.item' endpoint. On the left, there is a tree view of the API definition with expandable sections for HTTP Request, HTTP Headers, Authorization, Path Parameters, Query Parameters, and Body. On the right, there is a 'Updates' panel showing the mapping of the request to a service. A 'Move' button is visible between the two panels.

Map both the request and response for each API.

Map path and query parameters to native data structures.

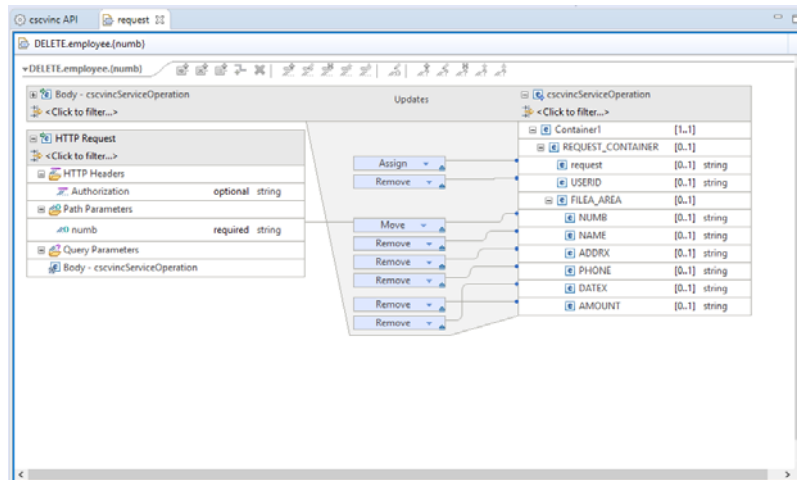
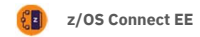
Assign static values to fields, useful for Op codes.

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

© 2018, 2019 IBM Corporation

API toolkit – API Editor

API mapping: Point-and-click interface

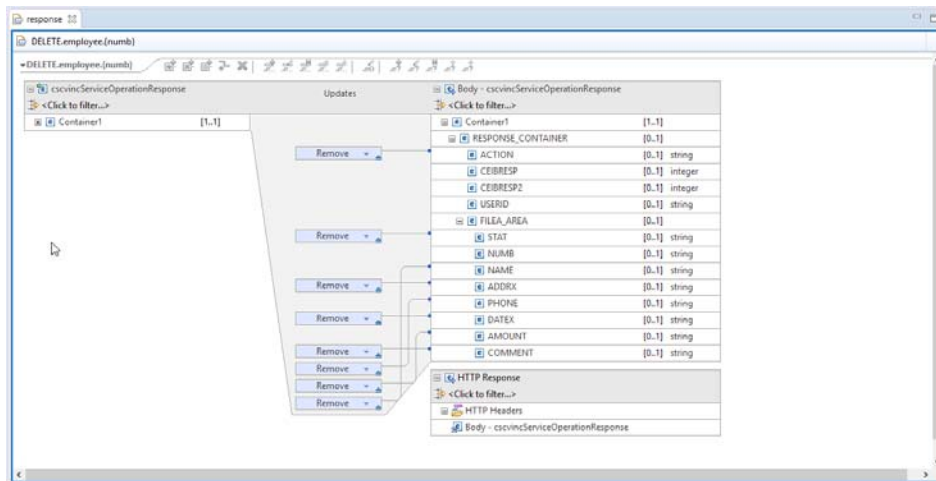
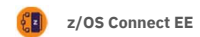


© 2018, 2019 IBM Corporation

API toolkit – API Editor

API mapping: Point-and-click interface

Allows the API Developer to remove fields to simplify the API

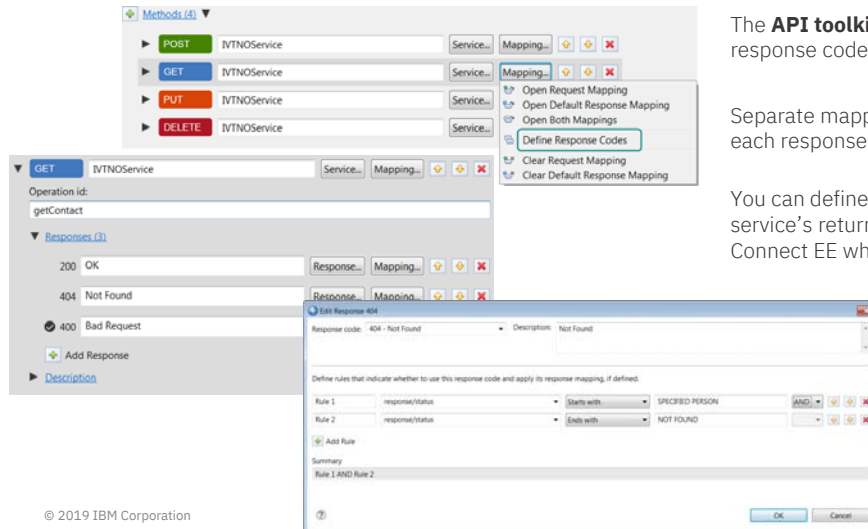


© 2018, 2019 IBM Corporation

API toolkit



API definition with multiple response codes



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

Separate mappings can be defined for each response code.

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

© 2019 IBM Corporation

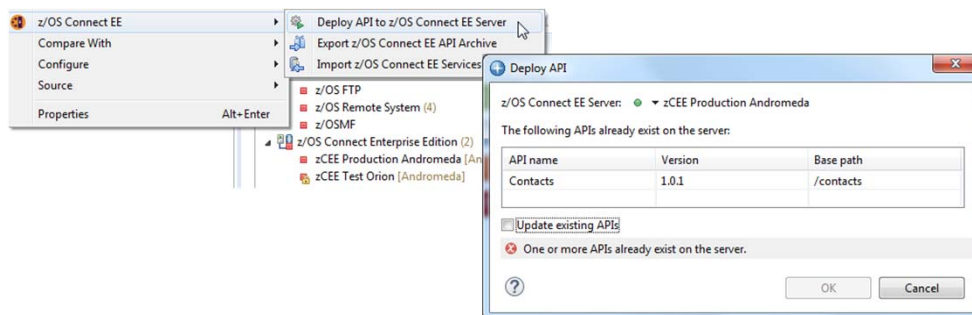
52

API toolkit – API Editor



Server connection and API deployment

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



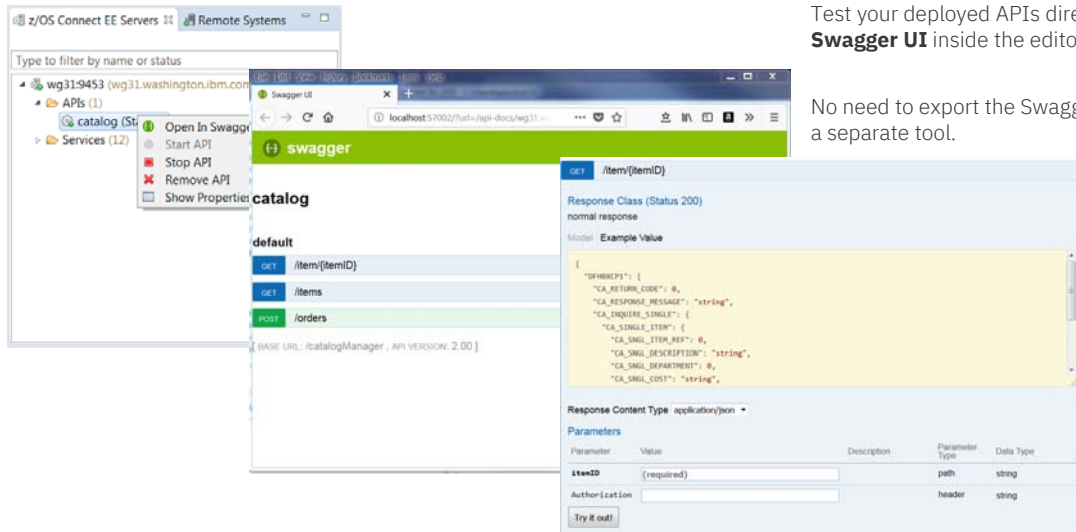
Right-click deploy to server enables developers to quickly deploy, test, and iterate on their APIs.

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

© 2018, 2019 IBM Corporation

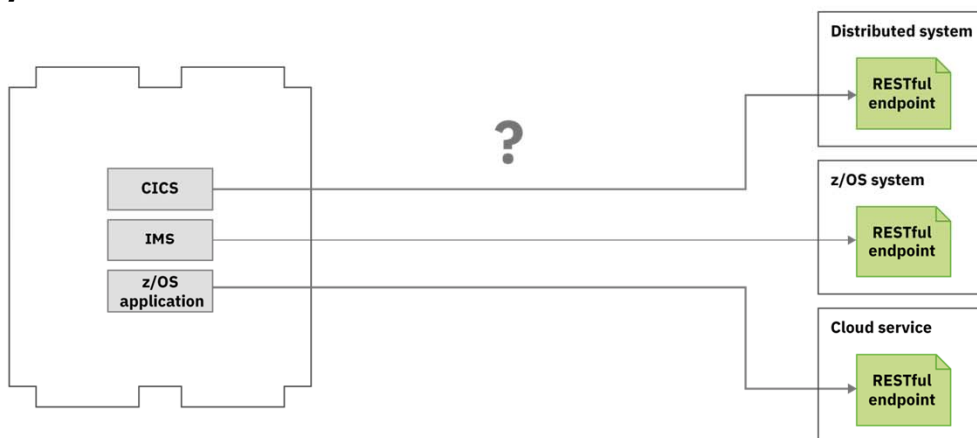
API toolkit – API Editor

Testing with Swagger UI




© 2018, 2019 IBM Corporation

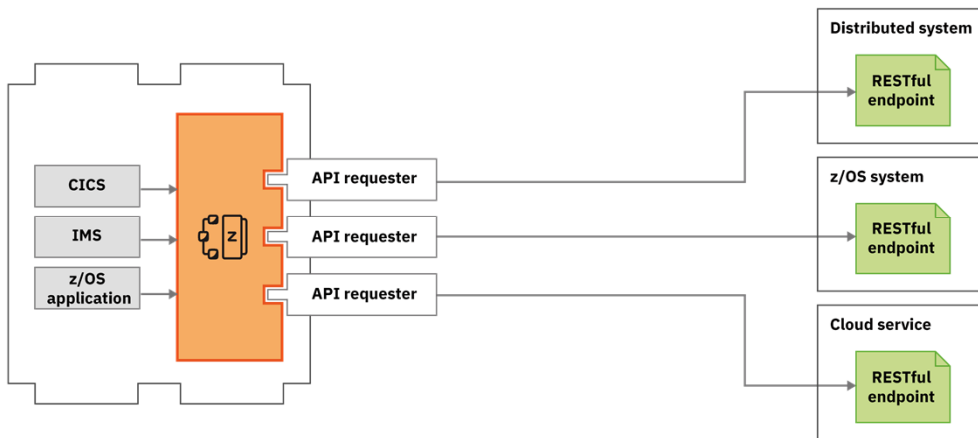
What about calling external APIs from my z/OS assets?



© 2018, 2019 IBM Corporation

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

 z/OS Connect EE

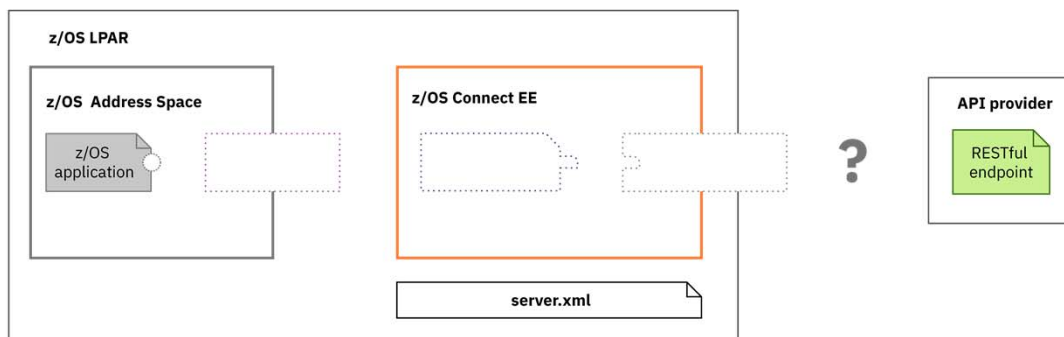


© 2018, 2019 IBM Corporation

Steps to calling an external API

 z/OS Connect EE

Starting point

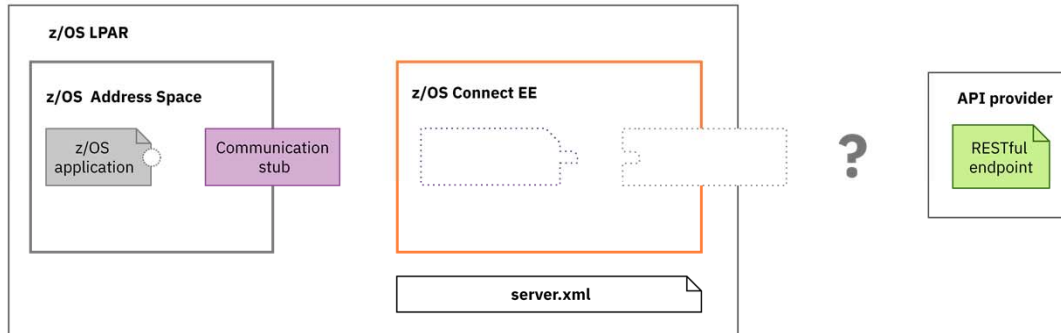


© 2018, 2019 IBM Corporation

Steps to calling an external API



Step 1. Configure communication stub



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

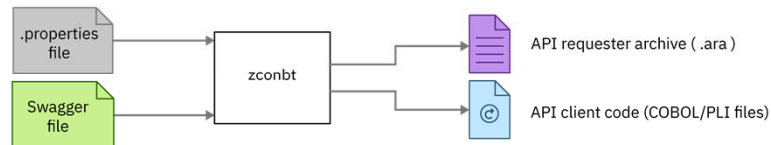
ibm.biz/zosconnect-configure-comms-stub

© 2018, 2019 IBM Corporation

Steps to calling an external API



Step 2. Generate API requester archive from Swagger

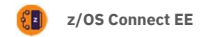


Generate your .ara file, and API client code.

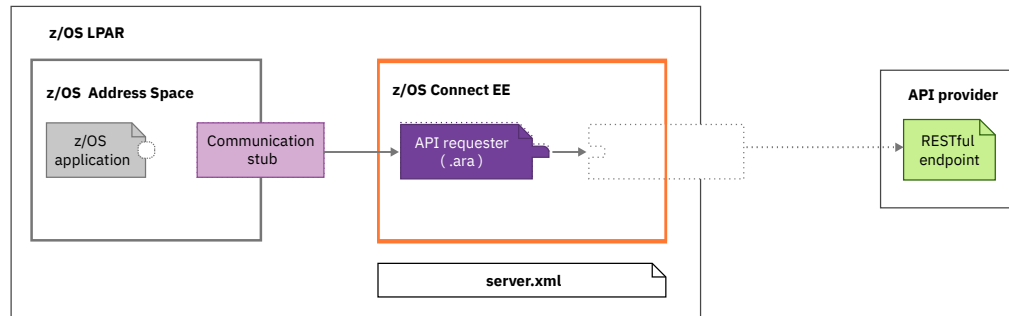
ibm.biz/zosconnect-generate-ara

© 2018, 2019 IBM Corporation

Steps to calling an external API



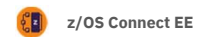
Step 3. Deploy API requester (.ara) archive



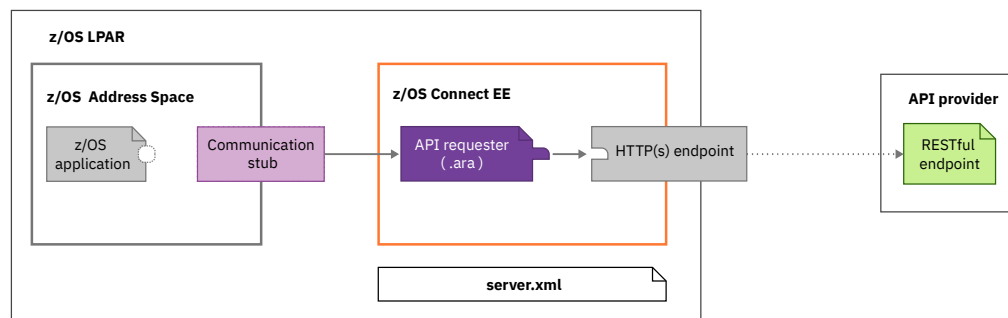
Deploy your API requester archive to the *apiRequester* directory.

© 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.

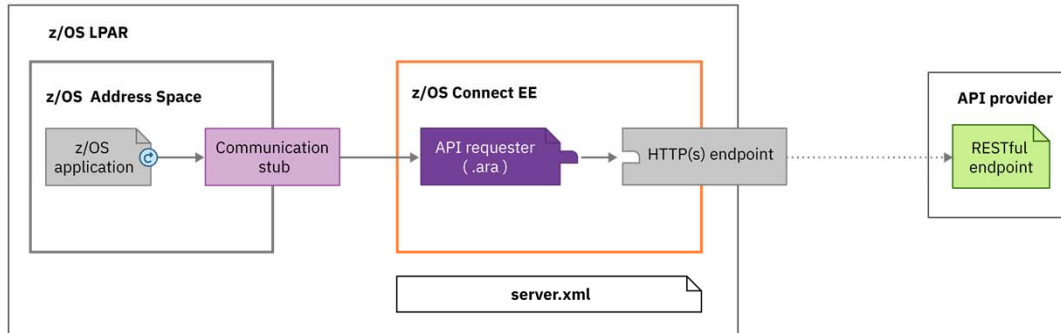
ibm.biz/zosconnect-configure-endpoint-connection

© 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

GETAPI

```

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

* Copy API Requester required copybook
COPY BAQRINFO.

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

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

* Request and Response segment
  
```

CSC02I01

```

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

apixml

```

<?xml version="1.0" encoding="UTF-8" standalone="yes">
<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_apiRequester>
    </zosconnect_apiRequesters>
  <zosconnect_endpointConnection id="cscvincAPI">
    host="http://wq31.washington.ibm.com"
    port="9120"
    basicAuthRef="myBasicAuth"
    connectionTimeout="10s"
    receiveTimeout="20s" />
  <zosconnect_authData id="myBasicAuth">
    user="Fred"
    password="fredmud" />
  </server>
  
```

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

© 2018, 2019 IBM Corporation

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
  
```

© 2018, 2019 IBM Corporation

Steps to calling an external API

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

```

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

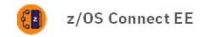
* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is
* successfully returned and fields in RESPONSE copybook
* can be obtained. Display the translation result.
IF BAQ-SUCCESS THEN
  DISPLAY "NUMB: " numb2 of API_RESPONSE
  DISPLAY "NAME: " name2 of API_RESPONSE
  DISPLAY "ADDRX: " addrx2 of API_RESPONSE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

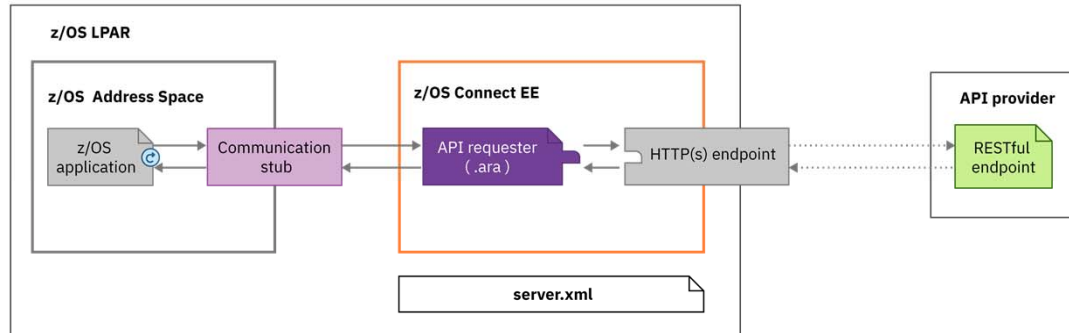
* When error happens in API, BAQ-RETURN-CODE is BAQ-ERROR-IN-API.
* BAQ-STATUS-CODE is the HTTP response code of API.
  
```

© 2018, 2019 IBM Corporation

Steps to calling an external API



Done



© 2018, 2019 IBM Corporation

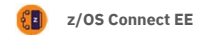


/common_scenarios

Typical connection patterns to different subsystems.

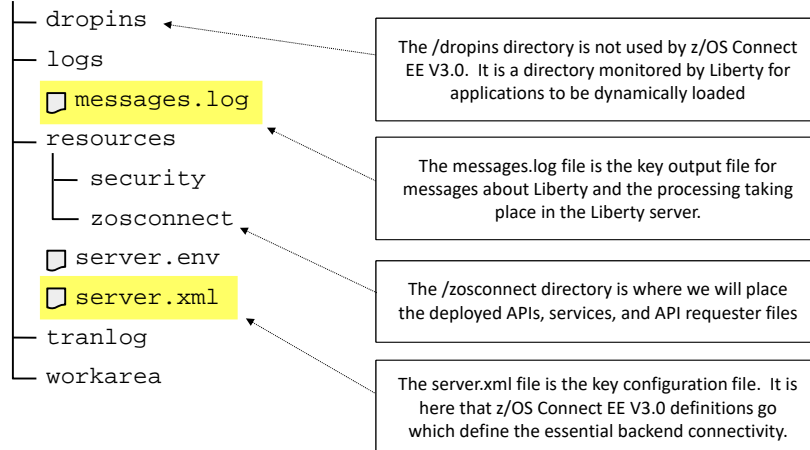
© 2018, 2019 IBM Corporation

Tour of Server Configuration Directories and Files



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

/var/zosconnect/servers/<server_name>

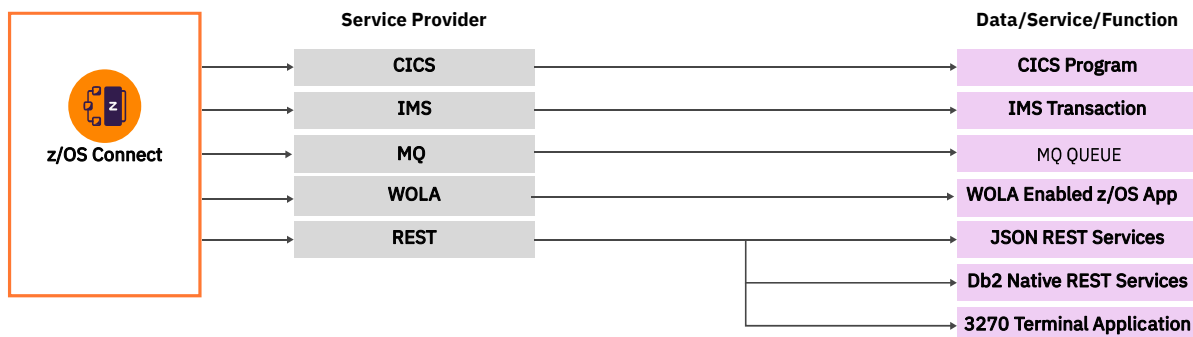


© 2018, 2019 IBM Corporation

What assets can z/OS Connect EE map to?



And which service provider could I use?

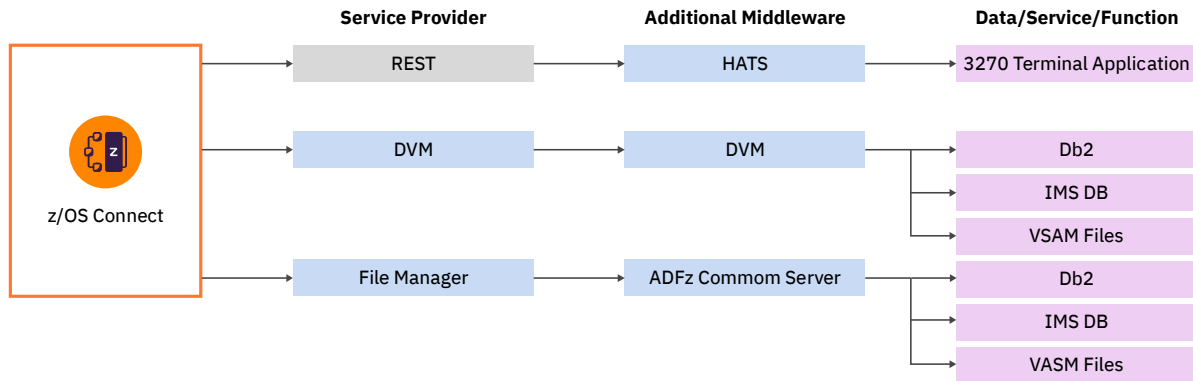


The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.

© 2019 IBM Corporation

Additional Middleware

Additional value from the ecosystem

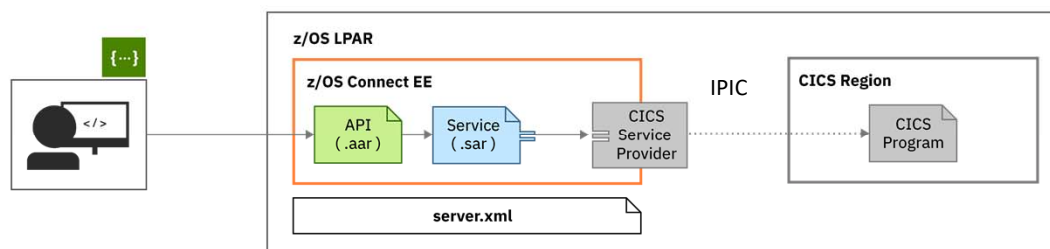


z/OS Connect EE is **pluggable** and **extensible** allowing the use of additional middleware to expand the list of z/OS assets you can expose as APIs

© 2018, 2019 IBM Corporation

Connections to CICS

Topology



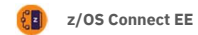
Connection to CICS is configured in `server.xml`.

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:

Define IPIC connection to CICS

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

```

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

```

z/OS Connect CICS IPIC connection

Defines a connection that enables requests to call CICS programs via an IPIC connection.

ID: catalog

Host: wg31.washington.ibm.com

Port: 1491

Shared port: false (default)

z/OS Connect APPLID: (no value)

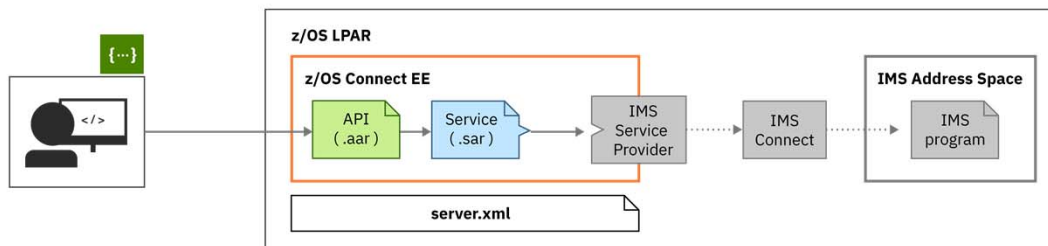
© 2018, 2019 IBM Corporation

The IMS server.xml file ...

Connections to IMS



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



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>
```

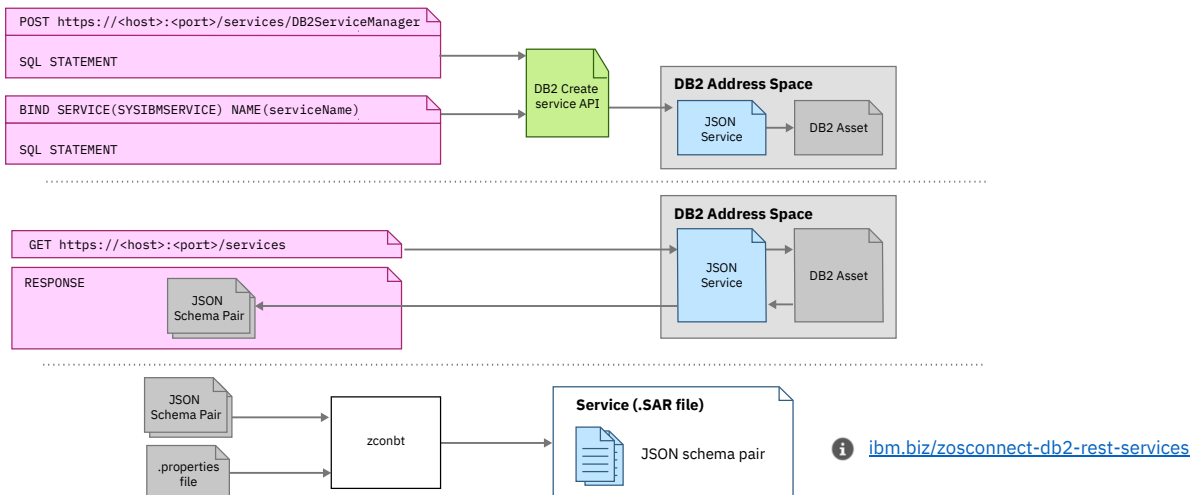
Interaction

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

© 2018, 2019 IBM Corporation

Connect to Db2

Create the service definition



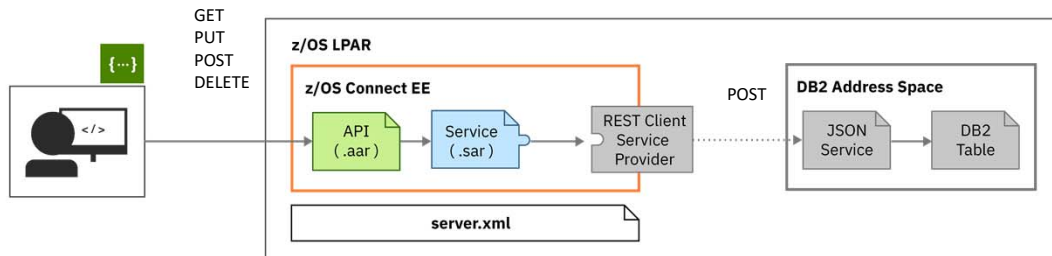
.sar file is created from the JSON schema of the DB2 service using the **zconbt** utility.

© 2018, 2019 IBM Corporation

Connections to Db2



Topology



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

A Db2 REST Service must be configured in DB2.

ibm.biz/zosconnect-db2-rest-services

© 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 <zoscconnect_zosConnectServiceRestClientConnection id="db2conn"
4   host="wg31.washington.ibm.com"
5   port="2446"
6   basicAuthRef="dsn2Auth" />
7
8 <zoscconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
9   applName="DSN2APPL"/>
10
11 </server>
12

```

```

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

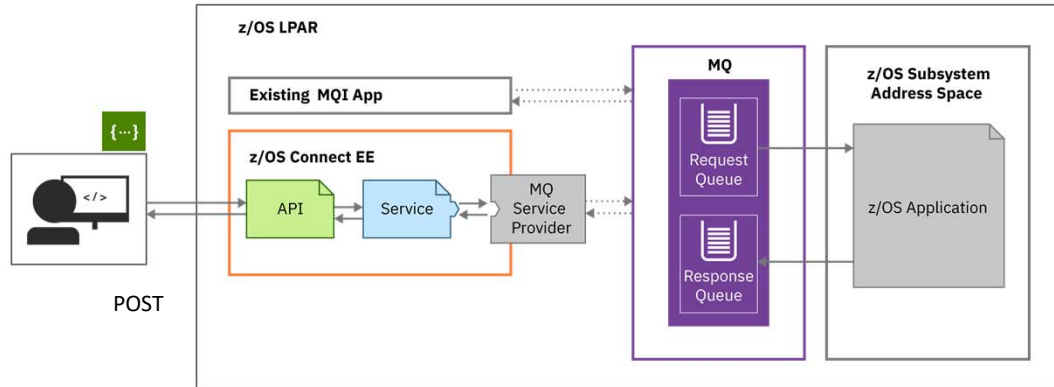
```

© 2018, 2019 IBM Corporation

Connections to MQ



Topology (Two-way service example)

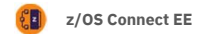


You can also configure one-way services.

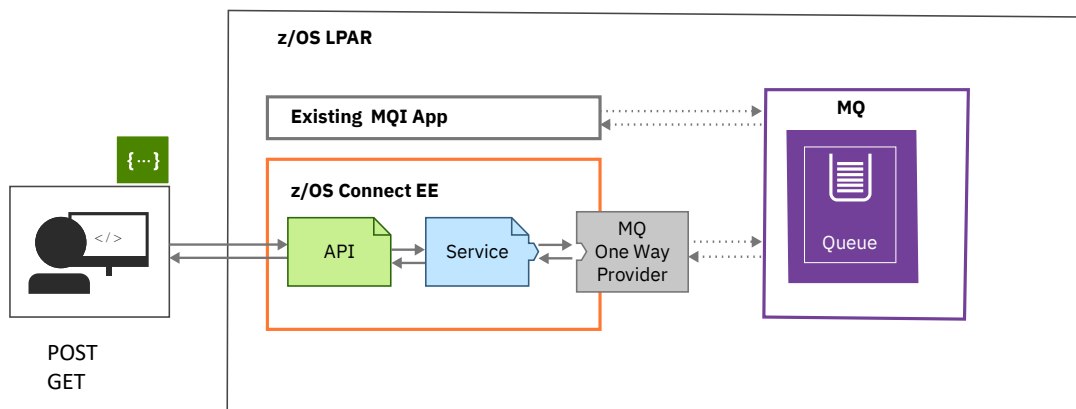
ibm.biz/zosconnect-mq-service-provider

© 2018, 2019 IBM Corporation

Connections to MQ



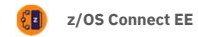
Topology (One-way service example)



ibm.biz/zosconnect-mq-service-provider

© 2018, 2019 IBM Corporation

The server.xml File (MQ)



mq.xml

Read only Close

Design Source

```

1 <server description="MQ Service Provider">
2
3   <featureManager>
4     <feature>zosconnect:mqService-1.0</feature>
5   </featureManager>
6
7   <variable name="vmqJmsClient.rar.location"
8     value="/usr/lpp/mqm/V9R1M1/java/lib/jca/vmq.jmsra.rar"/>
9   <vmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
10
11   <connectionManager id="ConMgr1" maxPoolSize="5"/>
12
13   <jmsConnectionFactory id="qmgrCF" jndiName="jms/qmgrCF"
14     connectionManagerRef="ConMgr1">
15     <properties>
16       <property name="transportType" value="BINDINGS" />
17       <property name="queueManager" value="QMZ1" />
18     </properties>
19   </jmsConnectionFactory>
20
21   <jmsQueue id="default" jndiName="jms/default">
22     <properties>
23       <property name="baseQueueName" value="ZCONN2.DEFAULT.MQZCEE.QUEUE" />
24       <property name="CCSID" value="37" />
25     </properties>
26   </jmsQueue>
27
28   <jmsQueue id="request" jndiName="jms/request">
29     <properties>
30       <property name="baseQueueName" value="ZCONN2.TRIGGER.REQUEST" />
31       <property name="targetClient" value="MQ" />
32       <property name="CCSID" value="37" />
33     </properties>
34   </jmsQueue>
35
36   <jmsQueue id="response" jndiName="jms/response">
37     <properties>
38       <property name="baseQueueName" value="ZCONN2.TRIGGER.RESPONSE" />
39       <property name="targetClient" value="MQ" />
40       <property name="CCSID" value="37" />
41     </properties>
42   </jmsQueue>
43 </server>

```

Features related to JMS Support

JMS Connection Factories,

JMS Destinations (queues)

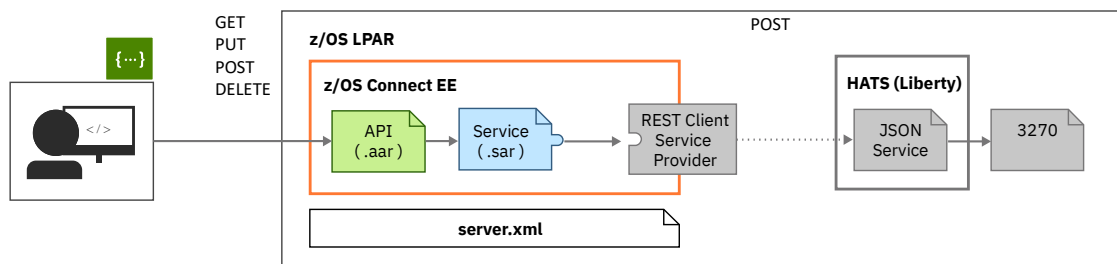
MQ V9.1.1 Added support for remote queue managers.

© 2018, 2019 IBM Corporation

Connection to HATS



Topology



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

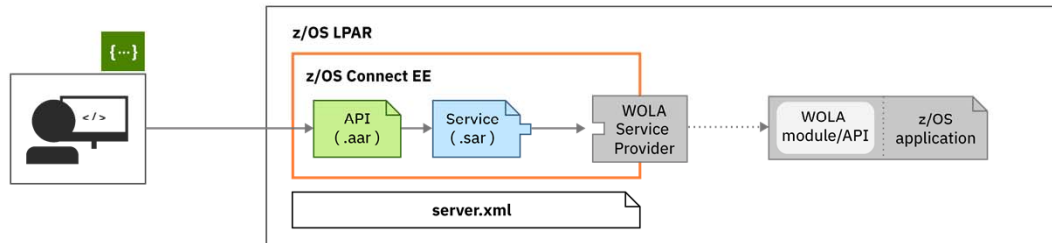
ibm.biz/zosconnect-db2-rest-services

© 2018, 2019 IBM Corporation

Connections to a MVS batch application



Topology

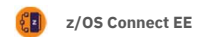


Connection to WOLA is configured in `server.xml`.

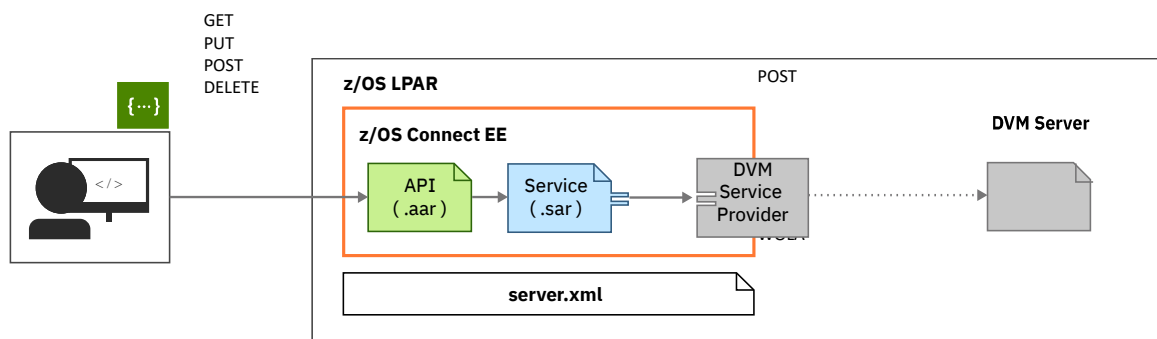
The z/OS application must be WOLA-enabled.

© 2018, 2019 IBM Corporation

Connections to DVM



Topology



The DVM service provider uses WOLA

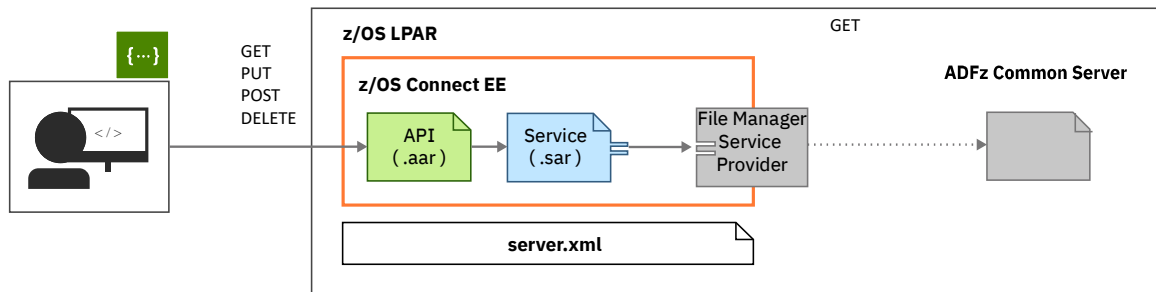
ibm.biz/zosconnect-db2-rest-services

© 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

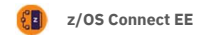


/miscellaneousTopics

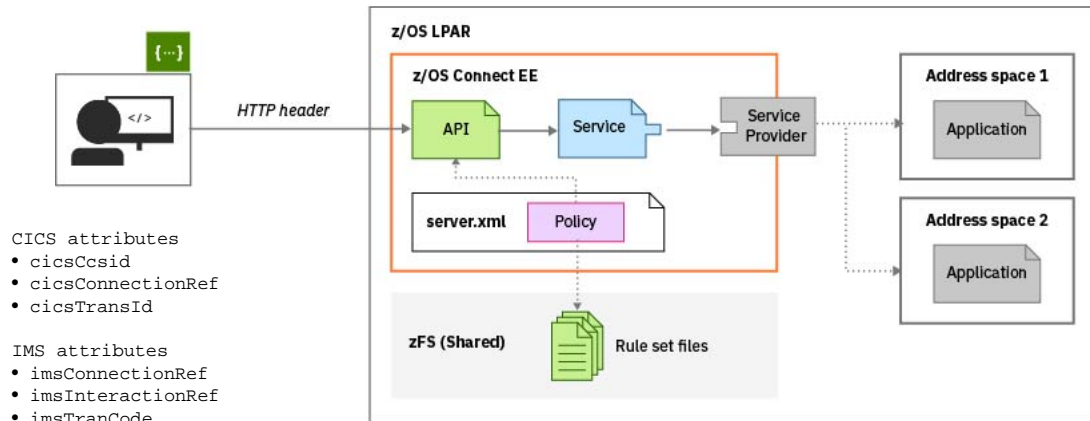
performance, high availability, Liberty

© 2018, 2019 IBM Corporation

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)

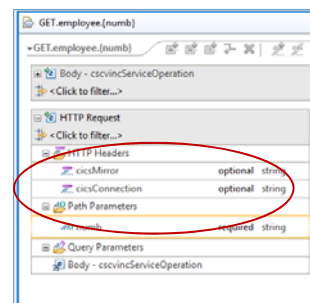


© 2018, 2019 IBM Corporation

A sample API Policies for CICS



```
<ruleset name="CICS rules">
  <rule name="csmi-rule">
    <conditions>
      <header name="cicsMirror" value="CSMI,MIJO"/> 1
    </conditions>
    <actions>
      <set property="cicsTransId" value="{cicsMirror}"/>
    </actions>
  </rule>
  <rule name="connection-rule">
    <conditions>
      <header name="cicsConnection"
        value="cscvinc,cics92,cics93"/>
    </conditions>
    <actions>
      <set property="cicsConnectionRef"
        value="{cicsConnection}"/>
    </actions>
  </rule>
</ruleset>
```



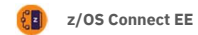
Curl

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

¹Transaction MIJO needs to be a clone of CSMI (e.g. invoke program DFHMIRS)

© 2018, 2019 IBM Corporation

Displaying zCEE messages on the console and/or spool



server.xml

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

MVS Console

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

STDERR

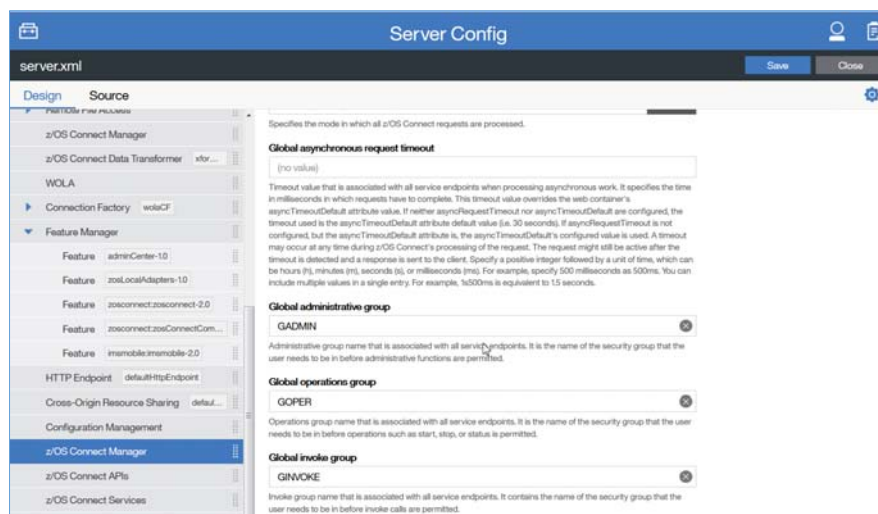
```
YERROR " BAQR0686E: Program CSCVINC is not available in the CICS region with connection cscvinc and service cscvincService.
YERROR " BAQR0686E: Program CSCVINC is not available in the CICS region with connection cscvinc and service cscvincService.
YERROR " BAQR0657E: Transaction abend MIJO occurred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

© 2018, 2019 IBM Corporation

Liberty's "adminCenter" Feature

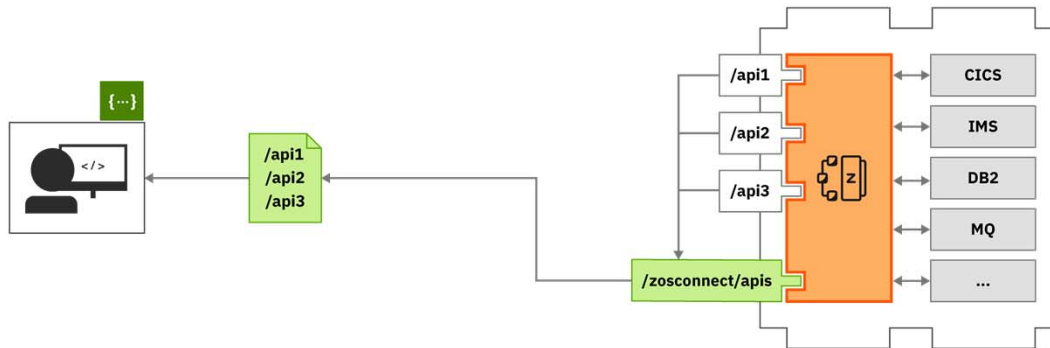


Web browser interface to the server's configuration files



© 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}`

*Useful for deploying DB2 and HATS
service archive files

© 2018, 2019 IBM Corporation

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

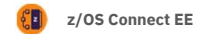
```

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

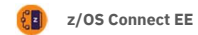
```

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

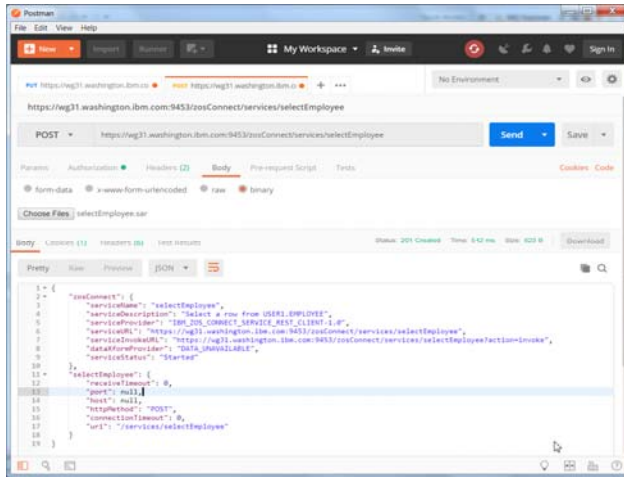
```

© 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",
    "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"
    }
  }
}
```

© 2018, 2019 IBM Corporation

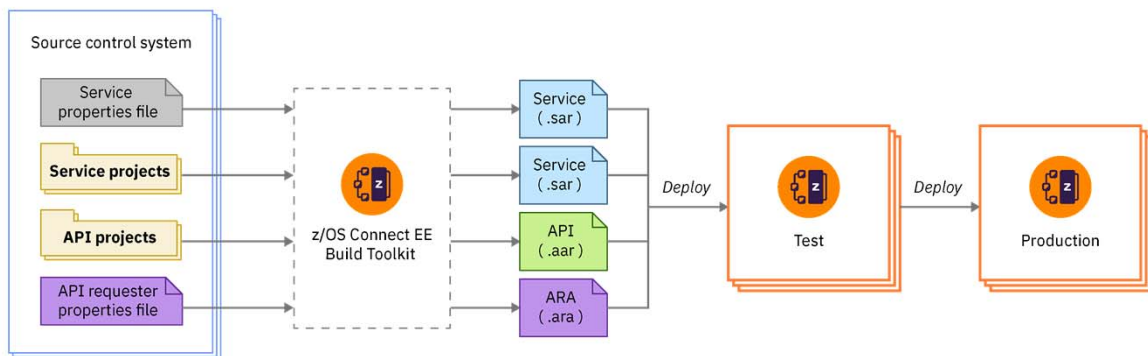
96

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



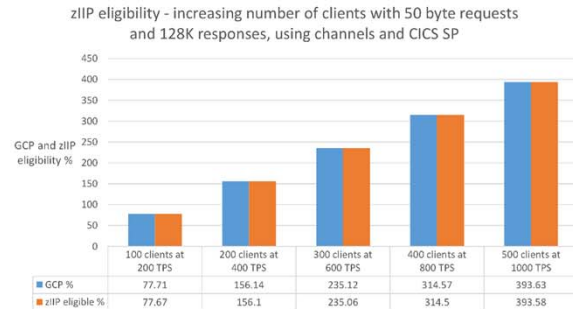
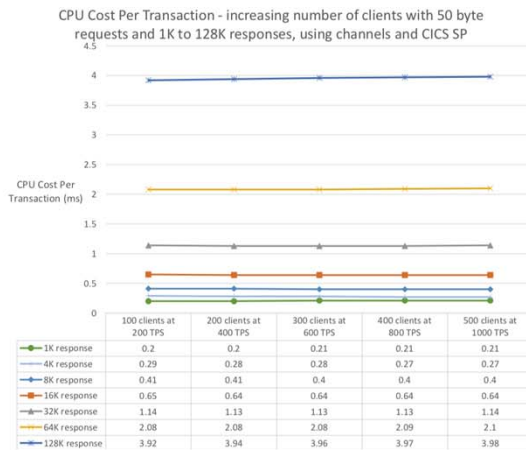
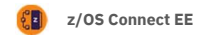
© 2019 IBM Corporation

ibm.biz/zosconnect-devops

97

Performance: API Provider

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**.

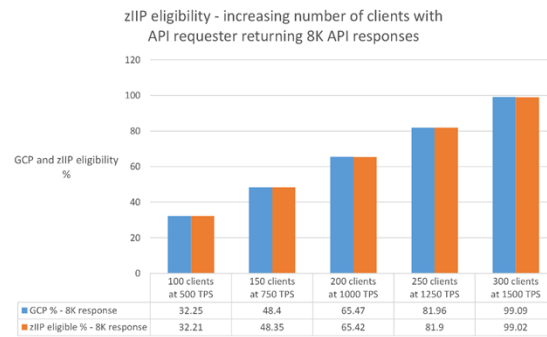
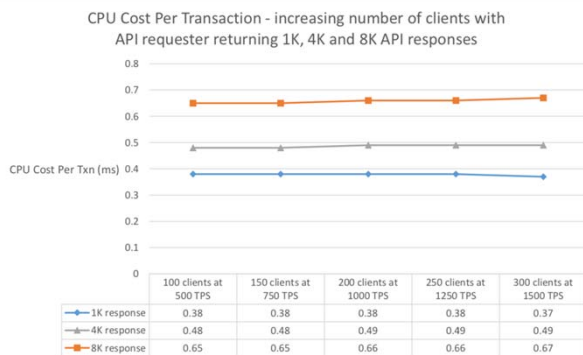
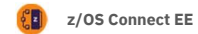
© 2019 IBM Corporation

ibm.biz/zosconnect-performance-report

100

Performance: API Requester

High Speed, High Throughput, Low Cost



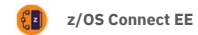
z/OS Connect EE is a Java-based product:
Over **99%** of its MIPs are **eligible** for **zIIP offload**.

© 2019 IBM Corporation

ibm.biz/zosconnect-performance-report

101

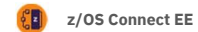
IBM z Omegamon for JVM



The screenshots show the IBM z Omegamon for JVM interface. The top two windows display 'z/OS Connect Request Summary' for various services like ADMIN, cecvnc, and filequeue. The bottom window shows 'z/OS Connect Request Detail' for a specific request, including fields like Event time, API name, Request URI, Method, Port, HTTP code, Service Name, Total Req Time, z/OS Conn Time, SQR Resp Time, SQR ID, SQR Ref, SQR Resource, Remote Address, Request Length, Response Length, Correlator, Operation, Provider, and User ID.

© 2018, 2019 IBM Corporation

IBM z Omegamon for JVM



The screenshots show the IBM z Omegamon for JVM interface displaying detailed request information for different services. The top two windows show details for 'filequeue' and 'selectEmployee' requests. The bottom two windows show details for 'selectEmployee' and 'phonebook' requests. The details include fields like Event time, API name, Request URI, Method, Port, HTTP code, Service Name, Total Req Time, z/OS Conn Time, SQR Resp Time, SQR ID, SQR Ref, SQR Resource, Remote Address, Request Length, Response Length, Correlator, Operation, Provider, and User ID.

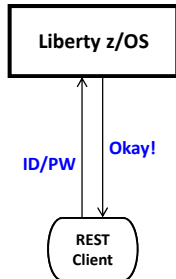
© 2018, 2019 IBM Corporation

Authentication



Several different ways this can be accomplished:

Basic Authentication



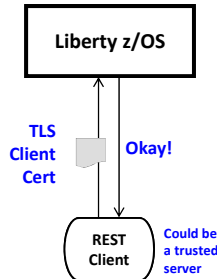
Server prompts for ID/PW

Client supplies ID/PW

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

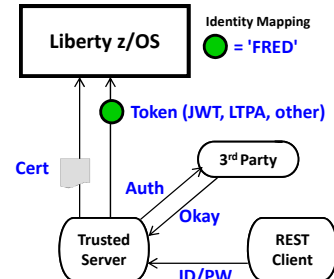
Client supplies certificate

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:

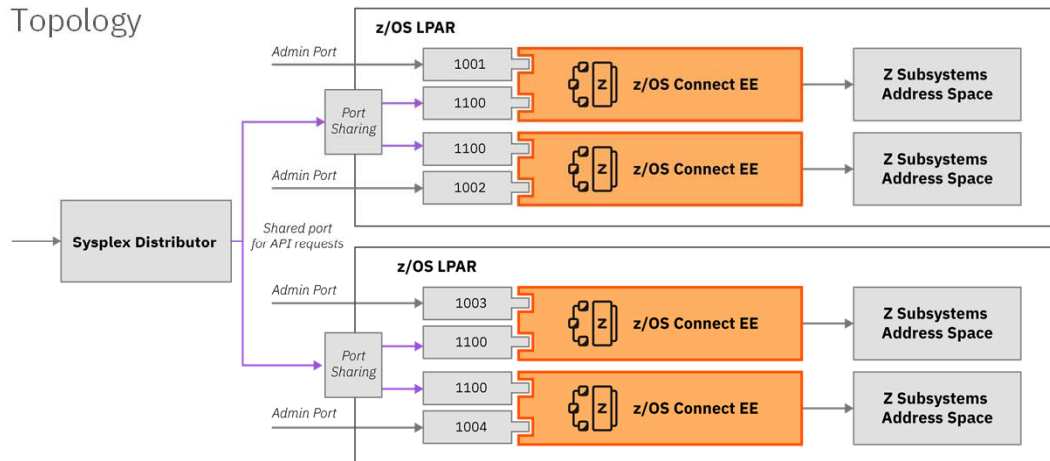
- LDAP
- SAF

© 2018, 2019 IBM Corporation

High Availability



Topology



ibm.biz/zosconnect-ha-concepts

ibm.biz/zosconnect-scenarios

© 2018, 2019 IBM Corporation



/questions?thanks=true

Thank you for listening.

© 2018, 2019 IBM Corporation

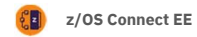


/exercises

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)

Or one or more of the following:

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

- 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
- 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