Db2 for z/OS: REST and Hybrid Cloud

Virtual Workshop

Keziah Knopp
Db2 for z/OS Specialist – WSC
keziah.knopp@ibm.com

Eric Higgins
ZStack Technical Specialist
erichiggins@us.ibm.com

Q: What role most closely aligns with your job?

A) System admin

B) DBA

C) System programmer

D) Application developer

E) Other

Agenda

Mobile Trends & the API Economy

RESTful APIs Overview

Db2 for z/OS REST Services
- Creating, discovering, and invoking Db2 REST services

Versioning Db2 REST Services

z/OS Connect EE Overview
- Service & deployment process, data mapping, and performance

Db2 REST & z/OS Connect EE

Lab Exercises 1: Native REST Services

Lab Exercises 2: z/OS Connect

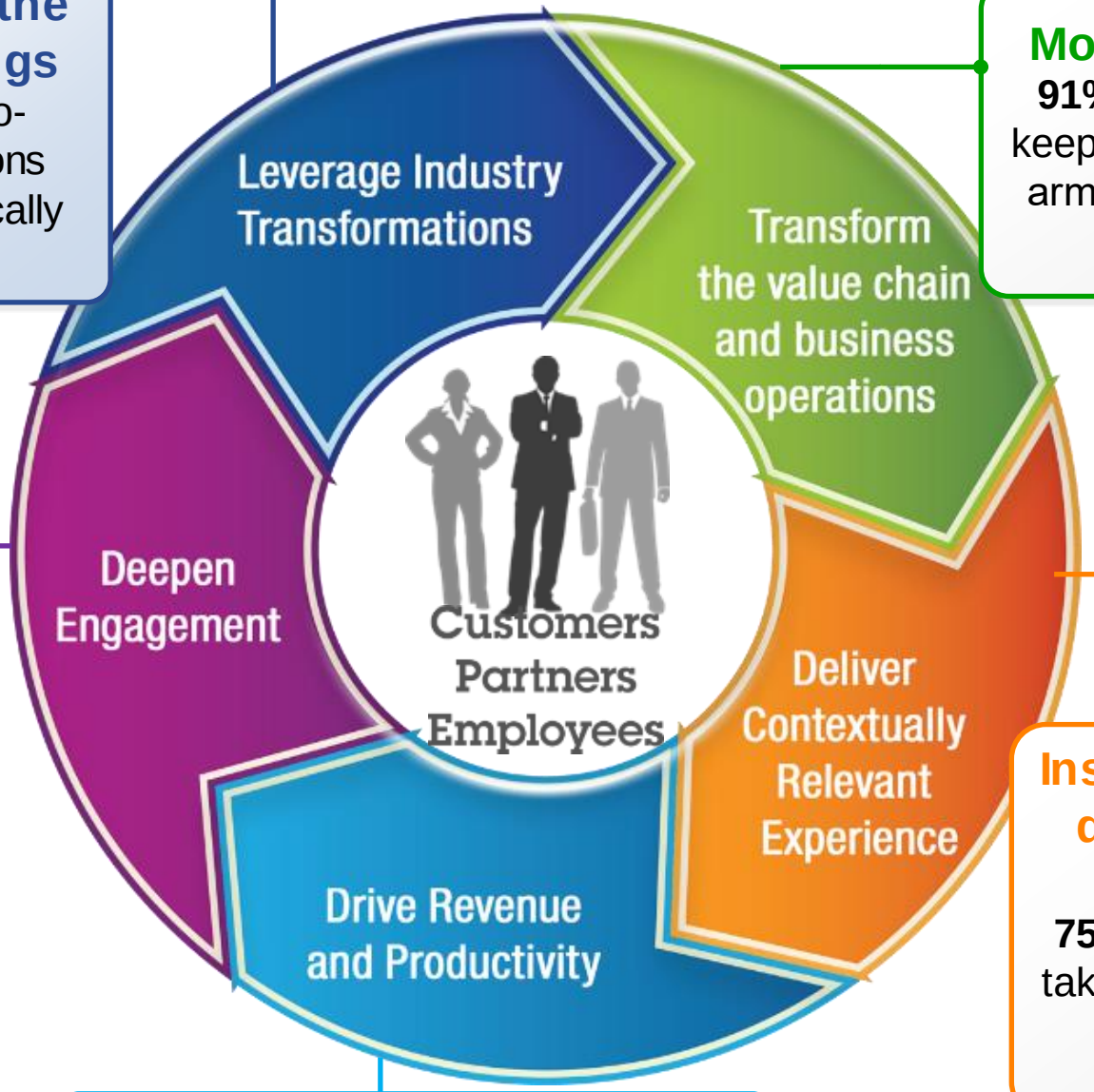
Mobile Trends & the API Economy

It’s not just a fad

5 mobile trends with significant implications for the enterprise

Mobile enables the Internet of Things
Global machine-to-machine connections increasing dramatically

Mobile is primary
91% of mobile users keep their device within arm's reach 100% of the time



Mobile must create a continuous brand experience
90% of users use multiple screens as channels come together to create integrated experiences

Insights from mobile data provide new opportunities
75% of mobile shoppers take action after receiving a location-based messages

Mobile is about transacting
Evidence: "Cyber Monday"

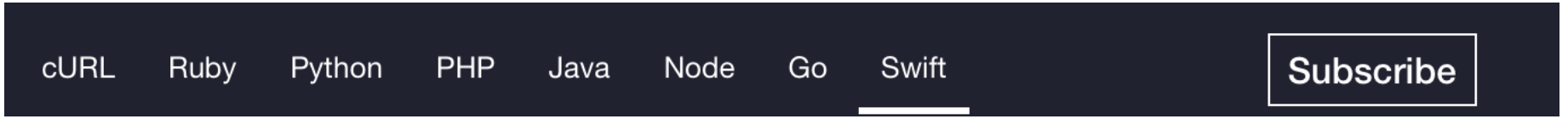
Expectations of Cloud

- I can connect to whatever I want
- I can try things out without penalty
- I won’t be tied to any one single solution
- Everything will be familiar and standard

Misconceptions of Z

- Doesn’t support modern technologies
- Changes take too long
- Security policies seem foreign
- Z doesn’t integrate

The Reality



default

GET /hotels

Response Class (Status 200)

normal response

Model | Example Value

Example Response

Definition

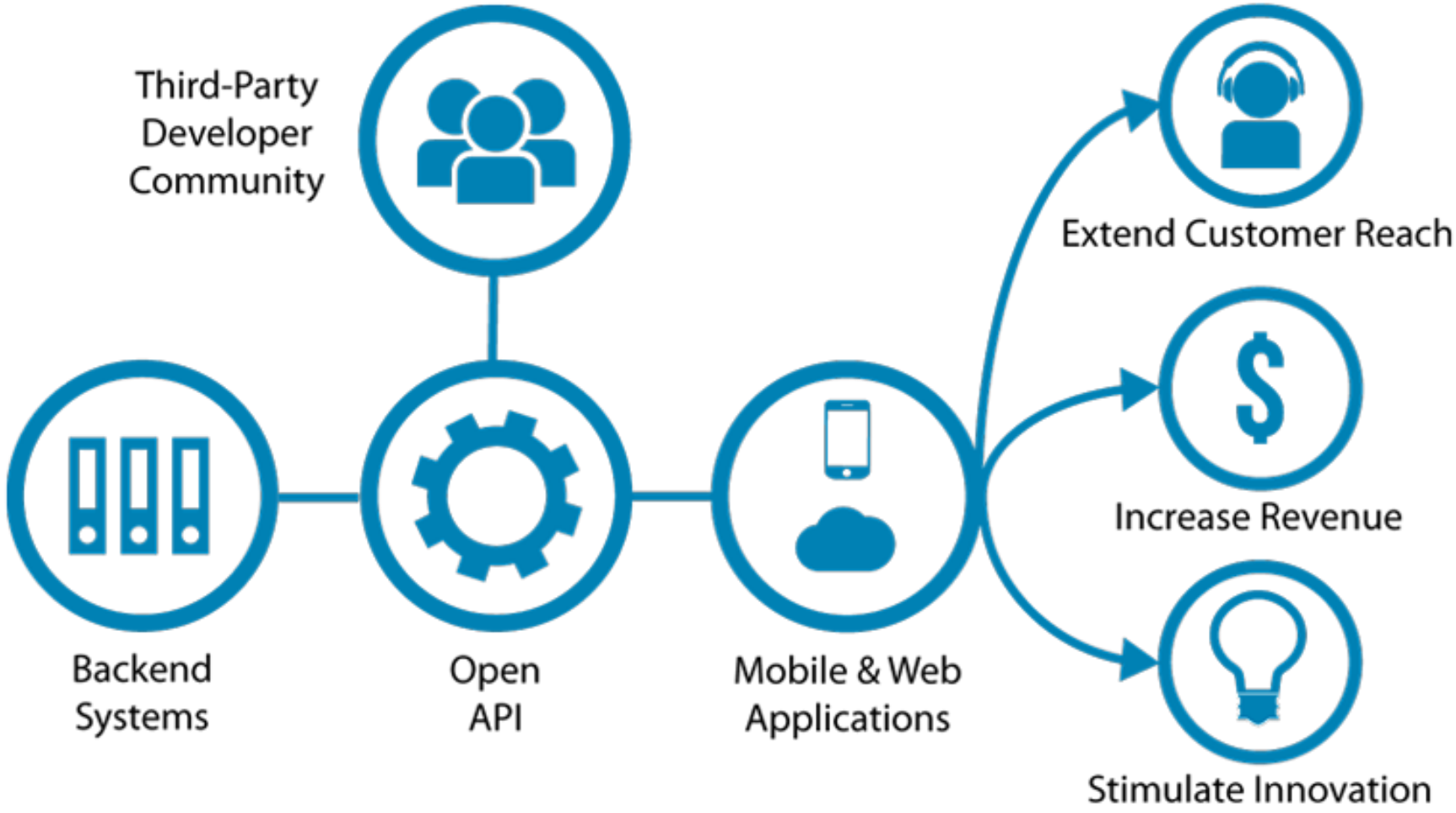
```
GET https://api.us.apiconnect.ibmcloud.com/jbistiusibmcom-dev/sb/travel_jbisti/hotels
```

Response

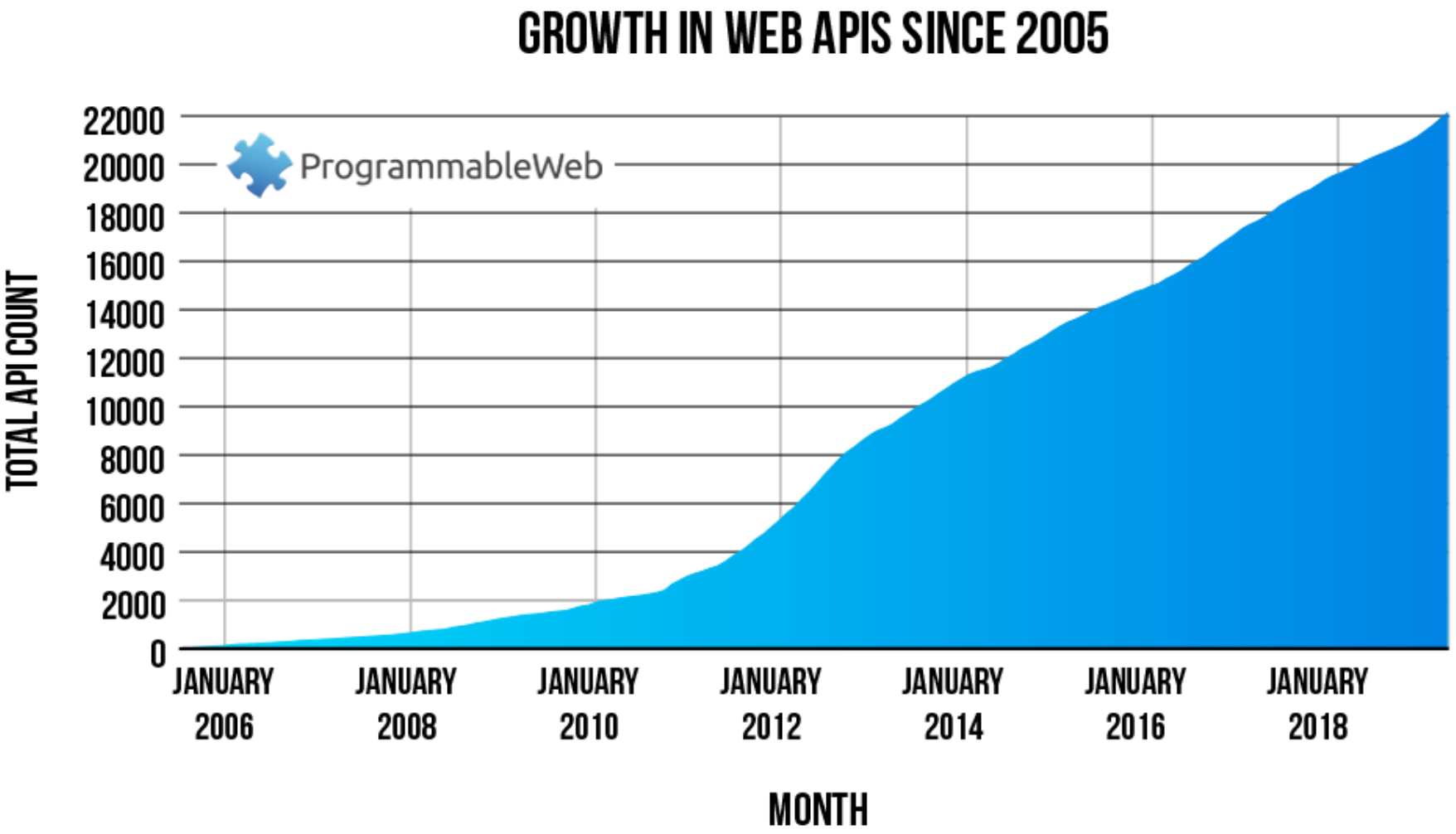

```
{
  "HOTEL_LIST_OUT": {
    "HOTELS": [
      {
        "HOTEL_NAME": "Maud Morales",
        "GEO": {
          "HOTEL_LAT": "ruhvumcucol",
          "HOTEL_LNG": "otie"
        },
        "HOTEL_LOCATION": "rawinelapracruduwtefebjujum",
        "HOTEL_RATING": "duol"
      }
    ]
  }
}
```

What is the API economy

The use of “business APIs” to positively affect the company



REST APIs are not just a fad...



Total new APIs added since 2015	8,076
Average new APIs added yearly	2,019
Average new APIs added monthly	168

- There are about 23,000 public APIs available.
- In the first 6 months of 2019, there was an increase of 30% of API growth, compared to the last four years (220 per month).

Q: What is your experience level with RESTful APIs?

A) Very familiar, like the back of my hand

B) I’ve worked with them before

C) I know REST services. Does that count?

D) APIs, sure. RESTful, maybe not.

E) This is the first I’m hearing about them

RESTful APIs

Technical overview

What is a **RESTful API**?

REST stands for Representational State Transfer architecture. (It is sometimes spelled "**ReST**".)

- stateless,
- client-server,
- cacheable communications protocol
  - ✓ HTTP protocol is used

A RESTful API is an application programming interface (API) that uses HTTP requests to GET, PUT, POST, and DELETE data.

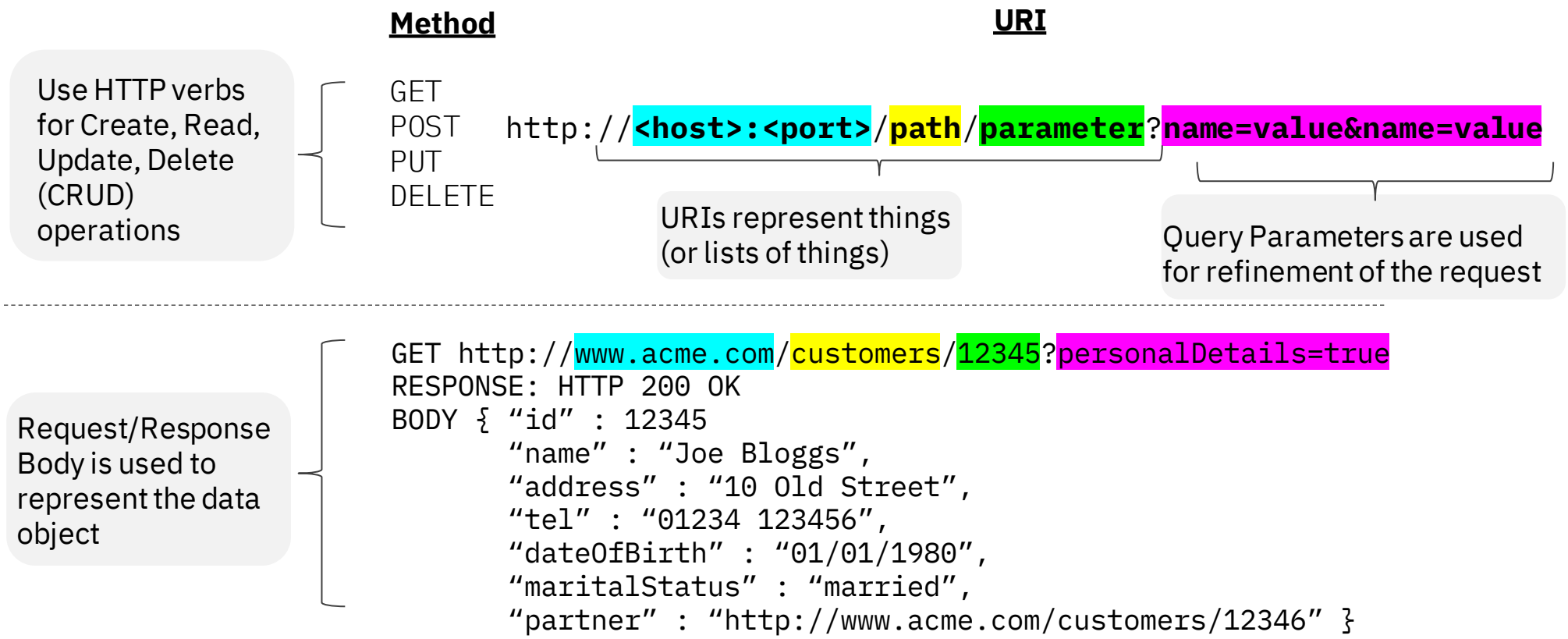
NOTE

Db2 native REST only supports the POST method for applications.

GET can be used for some system related functions only, not applications. The z/OS Connect EE API Editor allows you to reassign POST to a different method.

This is why Db2 native REST is REST, while zCEE is RESTful

Key principles of REST



REST and JSON

Throughout this workshop our focus will be on REST and JSON as the interface and data payload format:

Representational **S**tate **T**ransfer **(REST)**

```
http://www.myhost.com:port/account/update
```

The application understands what to do based on the URI

URI=Uniform Resource Identifier

Using HTTP verbs: GET, PUT, POST, etc.

Java**S**cript **O**bject **N**otation **(JSON)**

```
{
  "account":   "12345",
  "lastName":  "Smith",
  "action":    "Deposit",
  "amount":    "$1000.00",
}
```

Data is represented as a series of name/value pairs.

This is serialized and passed in with the URI or returned with a response.

HTTP Request Method Examples

Using the URL: **https://myhost.com/customer/235**

[GET] = Record for customer #235.

(in SQL terms - SELECT)

[PUT] + Info = Updated record for customer #235.

(in SQL terms - UPDATE)

[POST] + Info = New record for customer #235.

(in SQL terms - INSERT)

[DELETE] = Customer #235 Deleted.

(in SQL terms - DELETE)

NOTE

Db2 native REST can only use POST for applications, however this can be paired with SELECT, INSERT, UPDATE, DELETE, TRUNCATE, and WITH. For example, you can use the POST method with the SQL issuing a DELETE.

Why is REST popular?

Increasingly Common

Relatively Lightweight

Relatively Easy Development

Ubiquitous Foundation

Stateless

Db2 for z/OS REST Services

Technical overview

Db2 for z/OS REST **objectives**

Using REST and JSON to invoke one SQL statement or Stored Procedure

Enabling new business value for your enterprise data

Modernizing using the power of SQL

Unleashing Db2 data for the API Economy

Db2 REST services

Db2 REST service invocation

- New direct Db2 DDF REST access

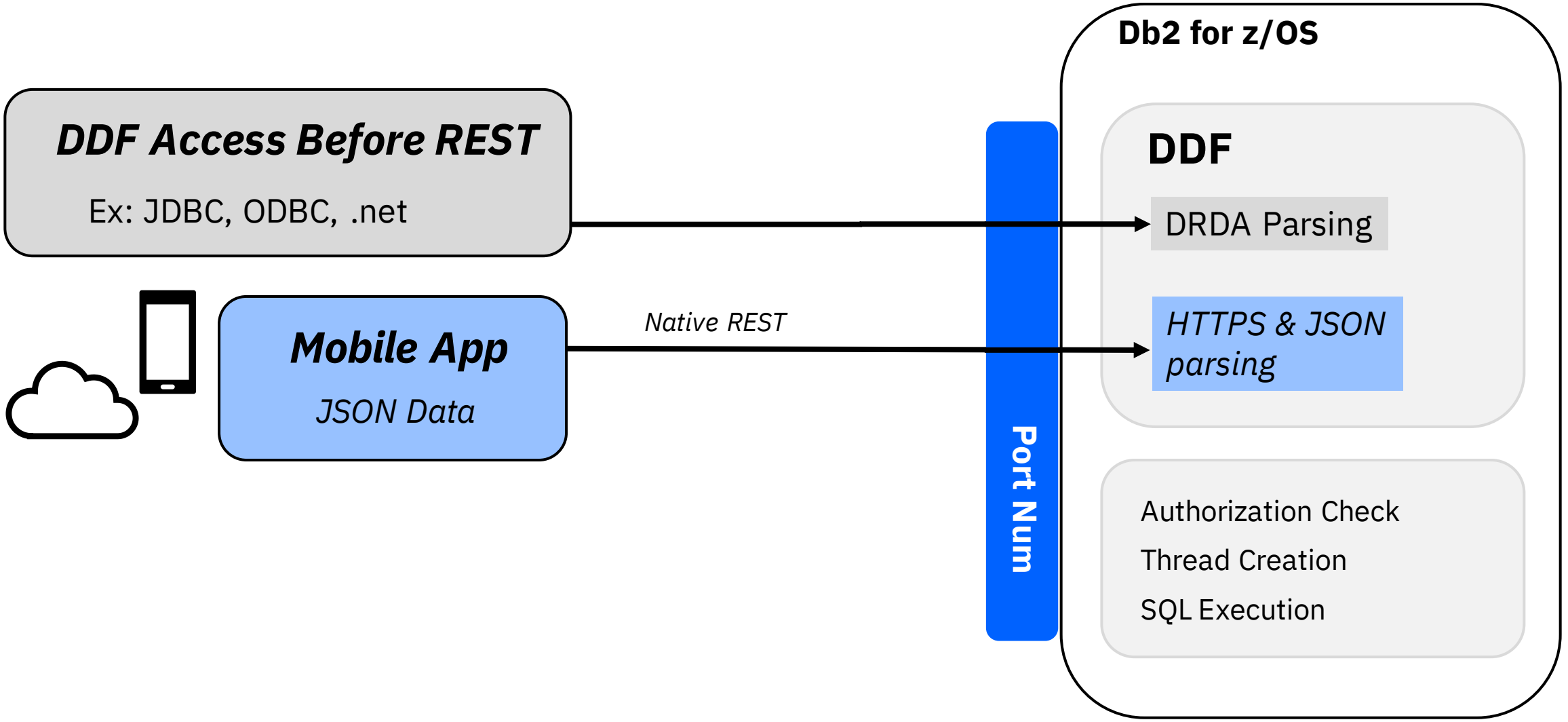
Service details

- One SQL statement or Stored Procedure call is permitted per service
  - Service is a statically bound package in Db2
- CALL, DELETE, INSERT, SELECT, TRUNCATE, UPDATE and WITH
  - MERGE can be in a service comprised of a Stored Procedure, not in a service comprised of a single SQL statement

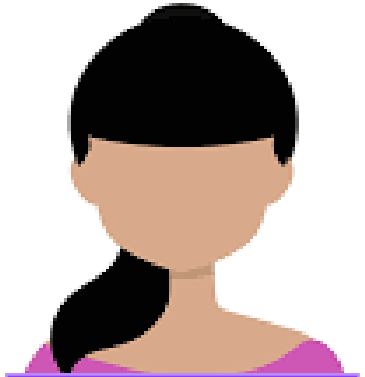
All of the various Db2 base SQL data types

- Including BLOB, CLOB, DBCLOB and XML

Architecture Diagram



When a developer goes to retrieve data



Invokes a Db2 REST service
- Service consists of Db2 stored procedure or SQL statement

Output returns in JSON format

Doesn’t need to know SQL, nor that the data came from Db2

Creates a service that consists of a stored procedure or SQL statement

Creates or reuses the stored procedure or SQL statement used in the REST call

Doesn’t need to know JSON

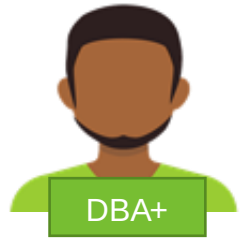
Managing Db2 REST services

REST client in browser
- Typically a plug-in

REST app
- Browser look and feel

BIND subcommand
- Standard Db2 interaction

Db2 REST Service Progression



Db2 REST Service Progression

Discover Service → Create the Service → Display the Service → Execute the Service → Delete the Service

Before you begin

You must have one of the following privileges or authorities to discover all Db2 REST services:

- Execute privilege on the package for the service
- Ownership of the service
- SYSADM or SYSCTRL authority
- System DBADM

Procedure

To discover all services, issue an HTTP or HTTPS GET or POST request through a REST client with the following URI:

- POST https://<host>:<port>/services/Db2ServiceDiscover
  - *Note: Set the HTTP header Accept and Content-Type fields to application/json for the POST request.*
- GET https://<host>:<port>/services

To discover all services using a browser use the following URL:

- https://<host>:<port>/services

Successful completion:

REST Status Code = 201.

This is an HTTP code - not Db2!

Db2 REST Service Progression

Discover Service | **Create the Service** | Display the Service | Execute the Service | Delete the Service

Before you begin

When you create a service, Db2 identifies you or the authorization ID that you use as the default owner of the service.

Therefore, you must have the required privileges to create a service and bind the associated package into a collection.

For example, you must be authorized to execute the SQL statement that is embedded in the service.

Procedure

To create a service, issue an HTTP or HTTPS POST request through a REST client with the following URI:

- POST https://<host>:<port>/services/Db2ServiceManager
  - *Note: Set the HTTP header Accept and Content-Type fields to application/json for the POST request.*

Specify the following HTTP body for the request – note the JSON name pair format:

```
{ "requestType": "createService",
 "sqlStmt": "<sqlStatement>",
 "collectionID": "<serviceCollectionID>",
 "serviceName": "<serviceName>",
 "description": "<serviceDescription>",
 "bindOption": "<bindOption>"}
```

Successful completion:

REST Status Code = 201.

This is an HTTP code - not Db2!

Creating a Db2 REST Service using JCL

Discover Service | **Create the Service** | Display the Service | Execute the Service | Delete the Service

Before you begin

The BIND SERVICE (DSN) subcommand builds an application package that represents a Db2 REST service.

The package owner must have the required authorization, such as SYSADM authority, to execute the SQL statement embedded in a package and to build the package.

Procedure

To create a service, submit a batch JCL job through a TSO interface with the following format:

Example:

```
//RESTSP   JOB MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID,REGION=0M
//BIND EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD DSN=DSN1210.DB2.SDSNEXIT,DISP=SHR
//         DD DSN=DSN1210.DB2.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//DSNSTMT  DD *
CALL EMPL_DEPTS_NAT(:whichQuery,:department1,:department2)
//SYSTSIN  DD *
DSN SYSTEM(DSN2)
BIND SERVICE("SYSIBMService") -
NAME("selectByDeptSP") -
SQLENCODING(1047) -
DESCRIPTION('Select employees by departments or department range')
```

Successful completion:

0000

This is not an HTTP code!

Batch JCL:
Stored Procedure

RESTSP.jcl | RESTV1.jcl | RESTV2.jcl | RESTV3.jcl | RESTFREE.jcl

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7-
//RESTSP   JOB MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID,REGION=0M
//BIND EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD DSN=DSN1210.DB2.SDSNEXIT,DISP=SHR
//         DD DSN=DSN1210.DB2.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//DSNSTMT  DD *
CALL EMPL_DEPTS_NAT(:whichQuery,:department1,:department2)
//SYSTSIN  DD *
DSN SYSTEM(DSN2)
BIND SERVICE("SYSIBMService") -
NAME("selectByDeptSP") -
SQLENCODING(1047) -
DESCRIPTION('Select employees by departments or department range')
/*
```

Postman:
Stored Procedure

POST Create a Db2 REST Service: St...

Create a Db2 REST Service: Stored Procedure

POST http://wg31.washington.ibm.com:2446/services/DB2ServiceManager

Params | Authorization | Headers (12) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
{
"requestType": "createService",
"sqlStmt": "call USER1.EMPL_DEPTS_NAT(:whichQuery,:department1,:department2)",
"collectionID": "SYSIBMSERVICE",
"serviceName": "selectByDeptSP",
"bindOption": "ISOLATION(UR)",
"description": "Select employees by departments or department range."
}
```

Db2 REST Service Progression

For more information about these steps, please visit the following pages in IBM Docs:

https://www.ibm.com/docs/en/db2-for-zos/12?topic=db2-rest-services

Troubleshooting REST service requests

CATEGORY	DESCRIPTION
1xx: Informational	Communicates transfer protocol-level information.
2xx: Success	Indicates that the client's request was accepted successfully.
3xx: Redirection	Indicates that the client must take some additional action in order to complete their request.
4xx: Client Error	This category of error status codes points the finger at clients.
5xx: Server Error	The server takes responsibility for these error status codes.

Common HTTP status codes for REST service error conditions

For more information on HTTP status codes, please visit: https://restfulapi.net/http-status-codes/

HTTP status code	Description
HTTP 500 (Internal Server Error)	Indicates that the server could not fulfill a request. In most cases, the HTTP status code is accompanied by a DB2 SQL code that provides more details about the error condition.
HTTP 400 (Bad Request)	Indicates a problem with an input parameter, such as a missing required input parameter, that is detected by the DB2 DDF native REST code prior to executing the DB2 SQL statement. This code is also used for many DB2ServiceManager failures (for example, Create/Drop service) and DB2DiscoverService failures (discover service/discover service details), which are typically caused by incorrect or missing inputs.
HTTP 401 (Unauthorized)	Indicates that the user could not be successfully authenticated.
HTTP 403 (Forbidden)	Indicates that the user might not have the required permissions to access a resource.

Versioning Db2 REST Services

APAR PI98649

Versions of REST Services

Allow for development and deployment of new versions of REST Services while existing versions are still being used

Built on existing package versioning support

Use same authorizations

Specify *“version ID”* or accept default “V1”

Select default version

URI Format

Original	/services[/<collection id>]/<service name> Example: /services/SYSIBMServices/displayEmployee
Versioning	/services/<collection id>/<service name>[/<version>] Example: /services/SYSIBMServices/selectByEmpNum/V1

/services/IBMServices/**displayEmployee**/

SELECT FNAME, LNAME FROM EMPLOYEE

Versioning ENABLED

/services/IBMServices/**selectByEmpNum**/V1

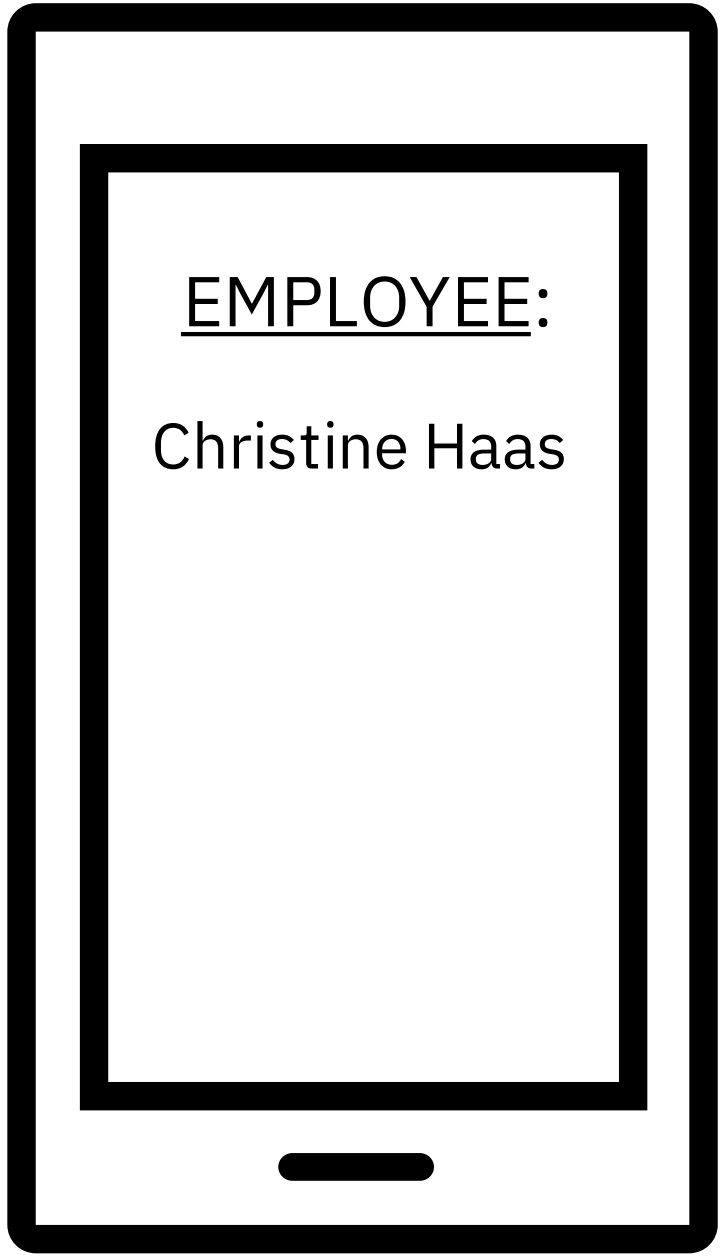
SELECT FIRSTNME, LASTNAME, PHONENO, WORKDEPT FROM DSN81210.EMP WHERE EMPNO = :EMPNUM

/services/IBMServices/**selectByEmpNum**/V2

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM and E.WORKDEPT = D.DEPTNO and D.MGRNO = M.EMPNO

/services/IBMServices/**selectByEmpNum**/V3

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER, M.PHONENO AS MGRPHONE FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM AND E.WORKDEPT = D.DEPTNO and



/services/IBMServices/**displayEmployee**/

SELECT FNAME, LNAME FROM EMPLOYEE

Versioning ENABLED

/services/IBMServices/**selectByEmpNum**/V1

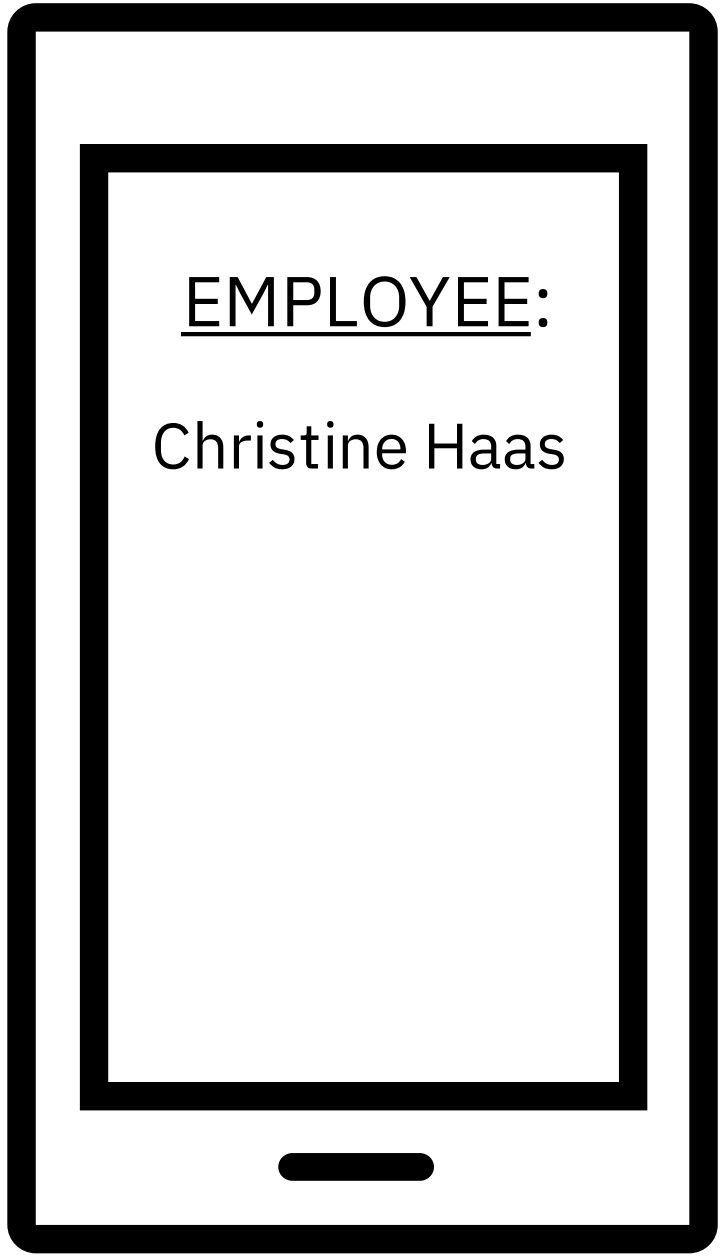
SELECT FIRSTNME, LASTNAME, PHONENO, WORKDEPT FROM DSN81210.EMP WHERE EMPNO = :EMPNUM

/services/IBMServices/**selectByEmpNum**/V2

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM and E.WORKDEPT = D.DEPTNO and D.MGRNO = M.EMPNO

/services/IBMServices/**selectByEmpNum**/V3

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER, M.PHONENO AS MGRPHONE FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM AND E.WORKDEPT = D.DEPTNO and



/services/IBMServices/**displayEmployee**/

SELECT FNAME, LNAME FROM EMPLOYEE

Versioning ENABLED

/services/IBMServices/**selectByEmpNum**/V1

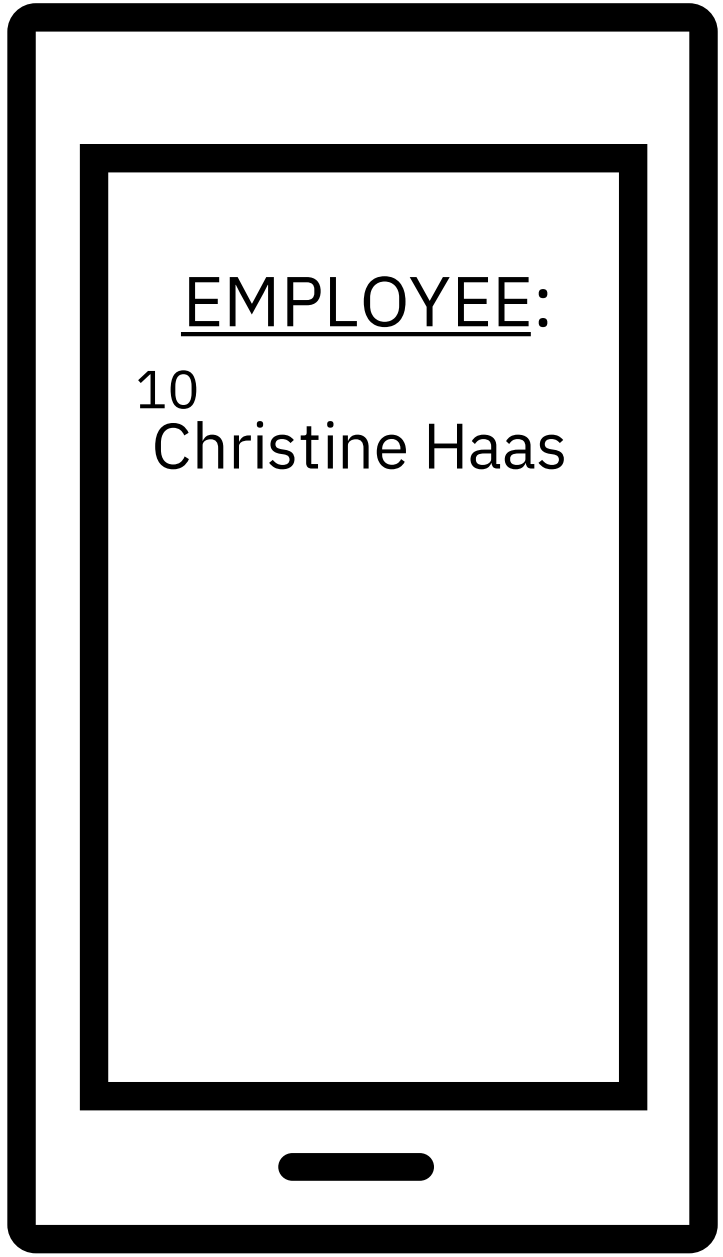
SELECT FIRSTNME, LASTNAME, PHONENO, WORKDEPT FROM DSN81210.EMP WHERE EMPNO = :EMPNUM

/services/IBMServices/**selectByEmpNum**/V2

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM and E.WORKDEPT = D.DEPTNO and D.MGRNO = M.EMPNO

/services/IBMServices/**selectByEmpNum**/V3

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER, M.PHONENO AS MGRPHONE FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM AND E.WORKDEPT = D.DEPTNO and



/services/IBMServices/**displayEmployee**/

SELECT FNAME, LNAME FROM EMPLOYEE

Versioning ENABLED

/services/IBMServices/**selectByEmpNum**/V1

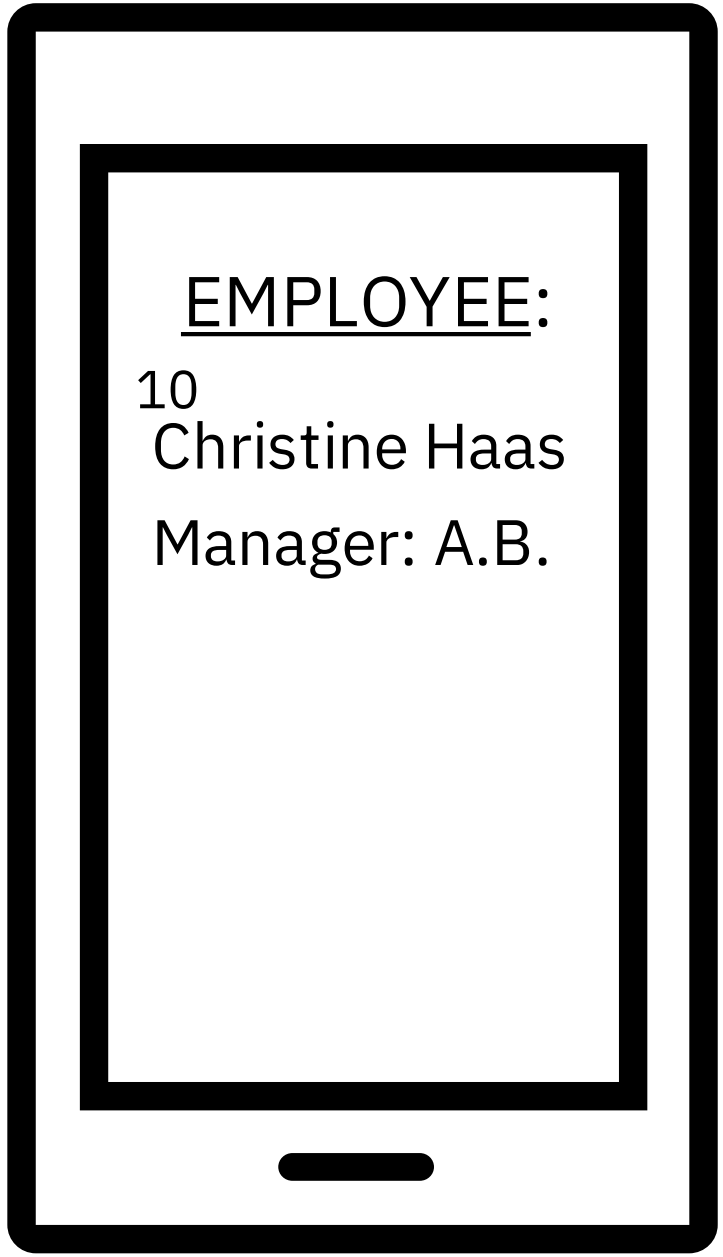
SELECT FIRSTNME, LASTNAME, PHONENO, WORKDEPT FROM DSN81210.EMP WHERE EMPNO = :EMPNUM

/services/IBMServices/**selectByEmpNum**/V2

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM and E.WORKDEPT = D.DEPTNO and D.MGRNO = M.EMPNO

/services/IBMServices/**selectByEmpNum**/V3

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER, M.PHONENO AS MGRPHONE FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM AND E.WORKDEPT = D.DEPTNO and



/services/IBMServices/**displayEmployee**/

SELECT FNAME, LNAME FROM EMPLOYEE

Versioning ENABLED

/services/IBMServices/**selectByEmpNum**/V1

SELECT FIRSTNME, LASTNAME, PHONENO, WORKDEPT FROM DSN81210.EMP WHERE EMPNO = :EMPNUM

/services/IBMServices/**selectByEmpNum**/V2

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM and E.WORKDEPT = D.DEPTNO and D.MGRNO = M.EMPNO

/services/IBMServices/**selectByEmpNum**/V3

SELECT E.FIRSTNME, E.LASTNAME, E.PHONENO, E.WORKDEPT, M.LASTNAME AS MANAGER, M.PHONENO AS MGRPHONE FROM DSN81210.EMP E, DSN81210.EMP M, DSN81210.DEPT D WHERE E.EMPNO = :EMPNUM AND E.WORKDEPT = D.DEPTNO and

Db2 REST Service Versioning Enablement

Apply Db2 APAR PI98649

Enable versioning by running sample job DSNTIJR2

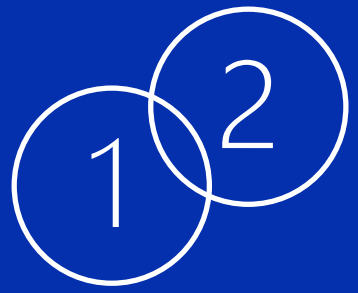
NOTE

If APAR PI98649 is **removed**, the entire Db2 REST service functionality will be **UNAVAILABLE**.

Versioning Features

No impact to pre-existing, version-less REST services

Empty string value “” version ID for version-less services



Services created after enablement are always versioned

Simplify modification of services; improve time to market

z/OS Connect

Modernize and transform

Q: What is your level of experience with z/OS Connect?

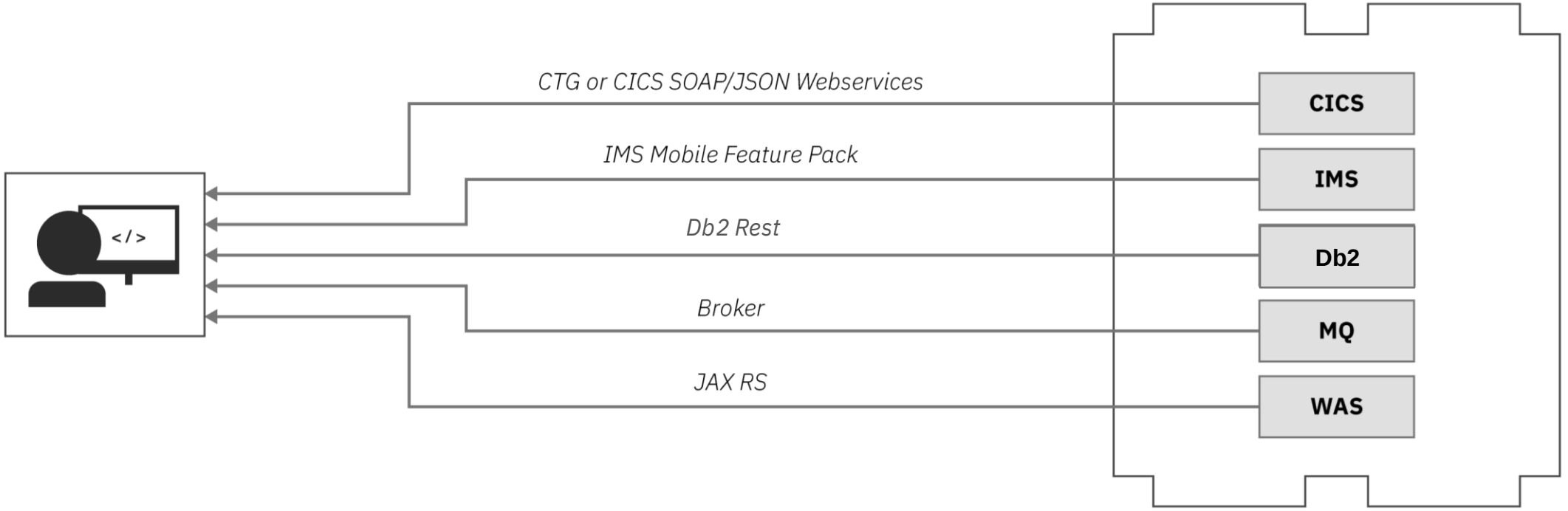
A) I use it regularly!

B) I’ve used it before

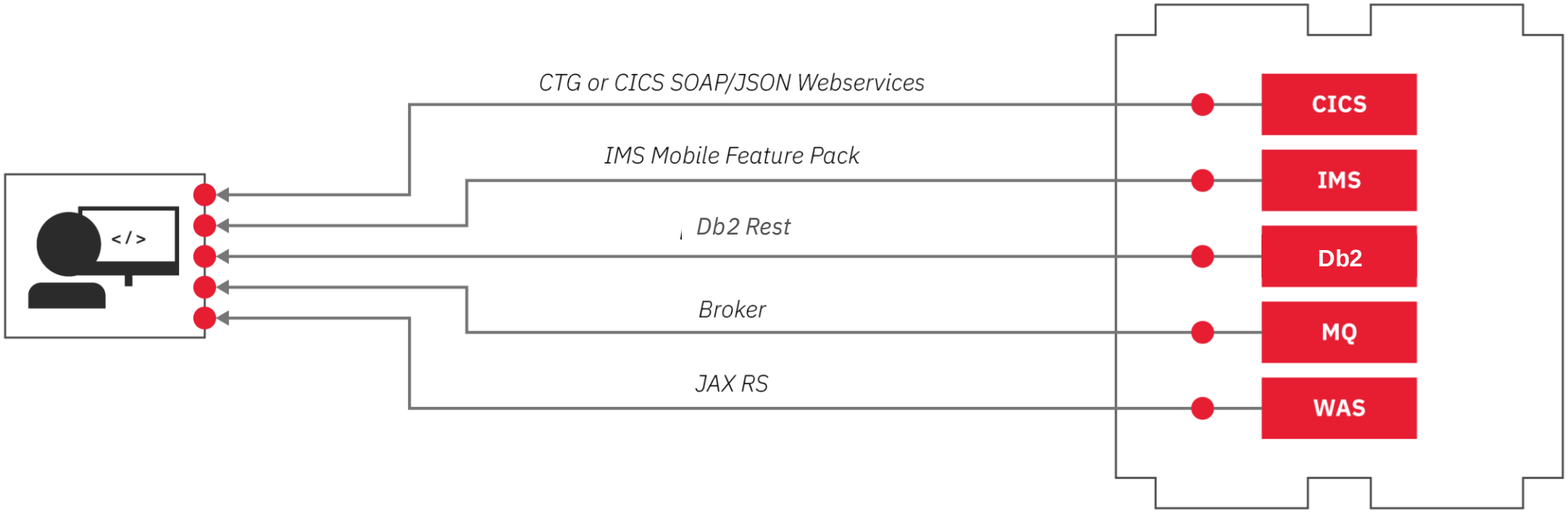
C) I’ve heard of it but haven’t used it

D) I don’t know what it is

Can’t we do JSON and REST already?



Sort of, but...



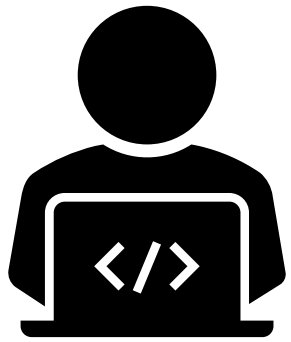
Completely different configuration and management

Multiple endpoints for developers to call/maintain

These are typically not RESTful!

Single Point of Entry

z/OS Connect exposes z/OS resources to the "cloud" via RESTful APIs



- CICS
- IMS
- Db2
- MQ
- File Manager
- 3270
- DVM
- WAS
- MVS

Single Configuration Administration

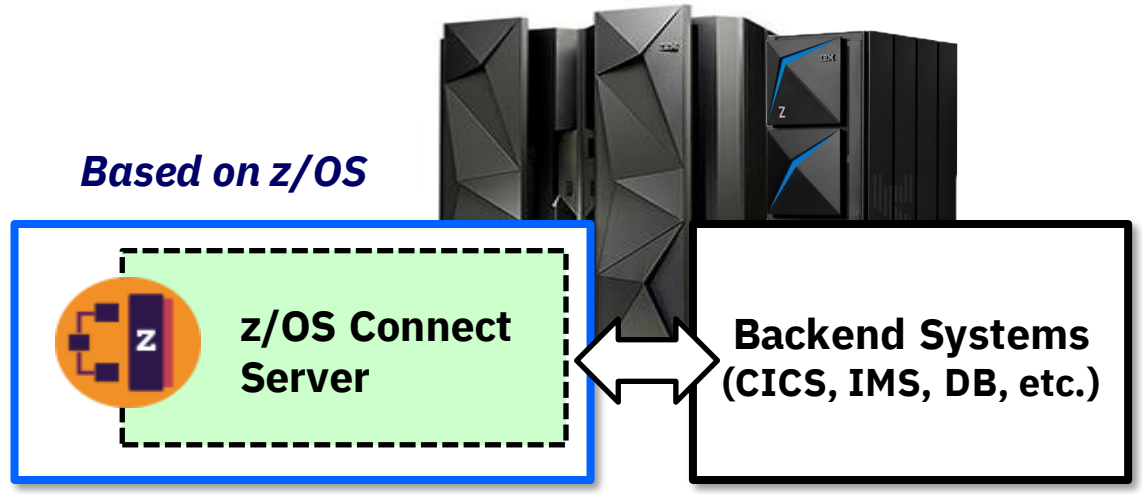
Single Security Administration

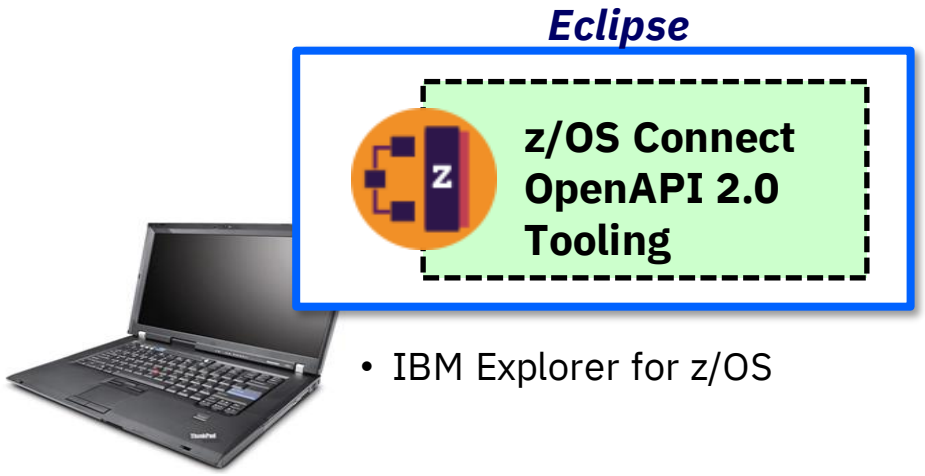
With sophisticated mapping of truly RESTful APIs to existing mainframe and services data without writing any code

High Level Overview of z/OS Connect

1 Runtime Server

- **Hosts APIs you define to run**
- **Connects with backend system**
- **Allows for multiple instances**



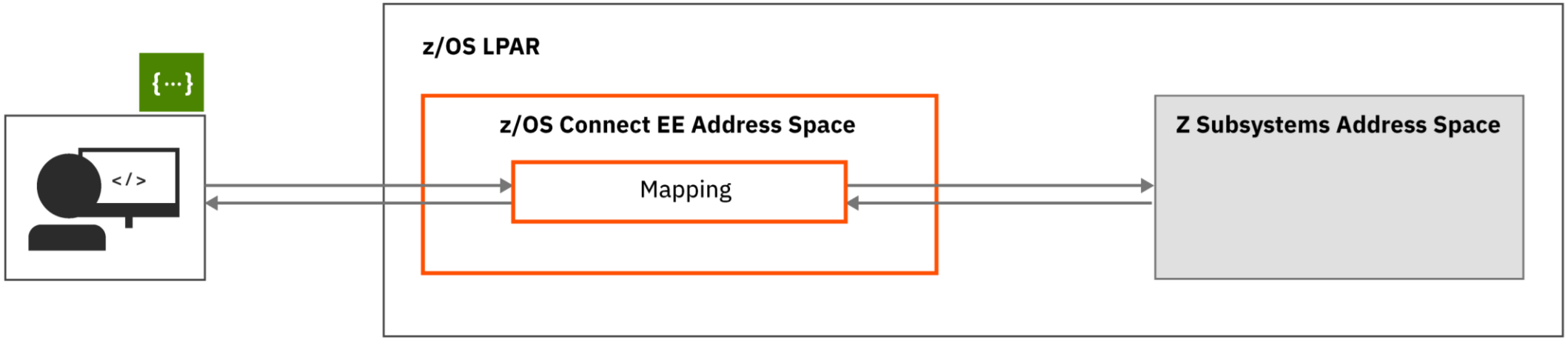


- IBM Explorer for z/OS

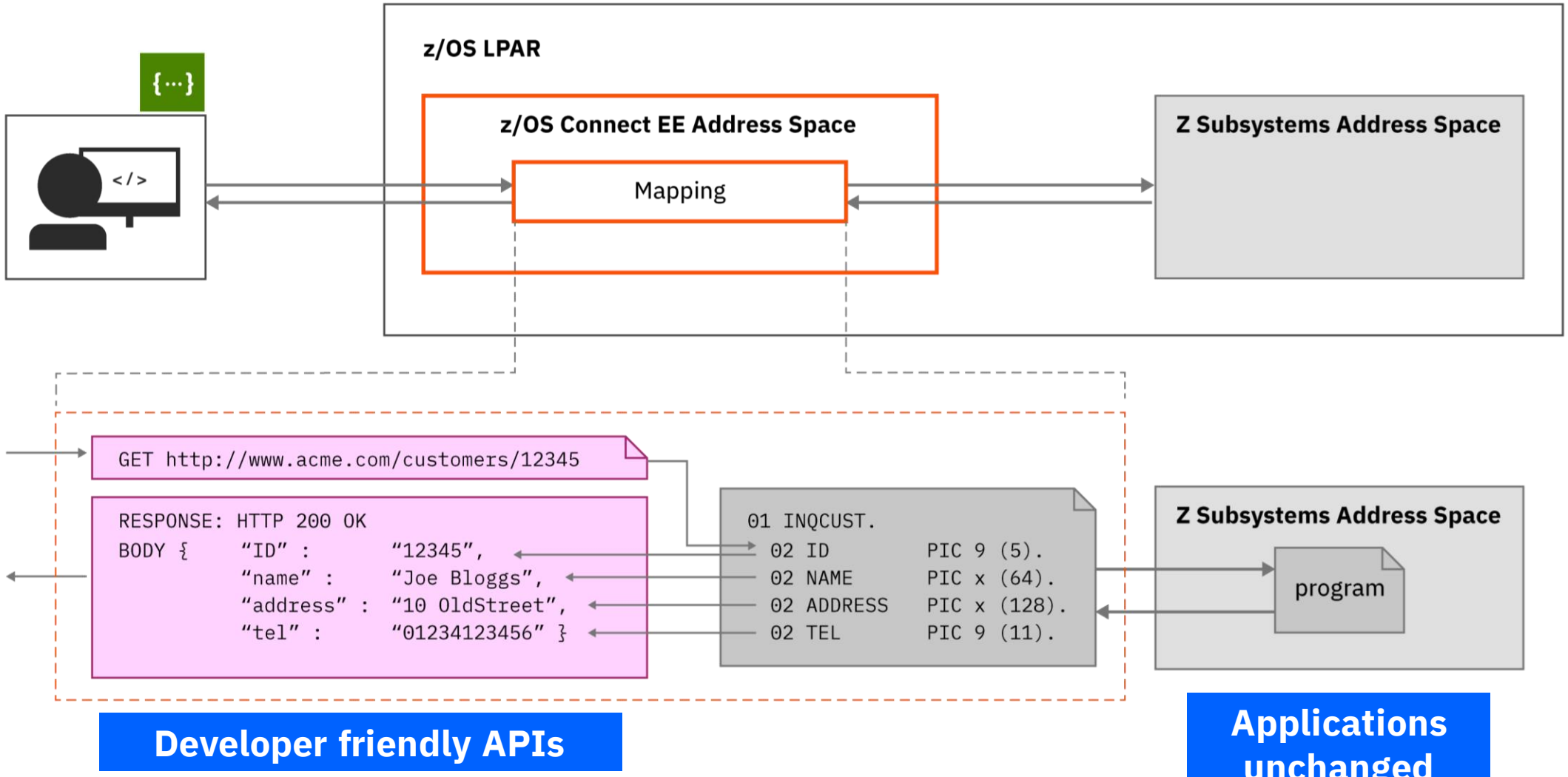
2 Tooling Platform – Eclipse based

- **Define APIs**
- **Define data mapping**
- **Deploy APIs to runtime server**
- **Create an Open**
- **Export API archive for other tools to deploy**

Data Mapping: Overview



Data Mapping: A Closer Look



API, Service Creation, & Deployment Process

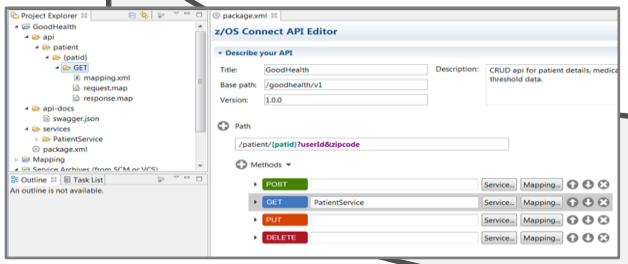
Tooling and utilities to extract JSON schema from existing applications

```
*
  Catalogue COMMAREA structure
    03 CA-REQUEST-ID                PIC X(6).
    03 CA-RETURN-CODE               PIC 9(2) DISPLAY.
    03 CA-RESPONSE-MESSAGE          PIC X(79).
* Fields used in Inquire Single
    03 CA-INQUIRE-SINGLE.
      05 CA-ITEM-REF-REQ            PIC 9(4) DISPLAY.
      05 FILLER                     PIC 9(4) DISPLAY.
      05 FILLER                     PIC 9(3) DISPLAY.
      05 CA-SINGLE-ITEM.
        07 CA-SNGL-ITEM-REF         PIC 9(4) DISPLAY.
        07 CA-SNGL-DESCRIPTION      PIC X(40).
        07 CA-SNGL-DEPARTMENT       PIC 9(3) DISPLAY.
        07 CA-SNGL-COST             PIC X(6).
        07 IN-SNGL-STOCK            PIC 9(4) DISPLAY.
        07 ON-SNGL-ORDER            PIC 9(3) DISPLAY.
```

Backend Program

Source of copy book and target of invocation

API Editor



API and Services Creation

API Definition

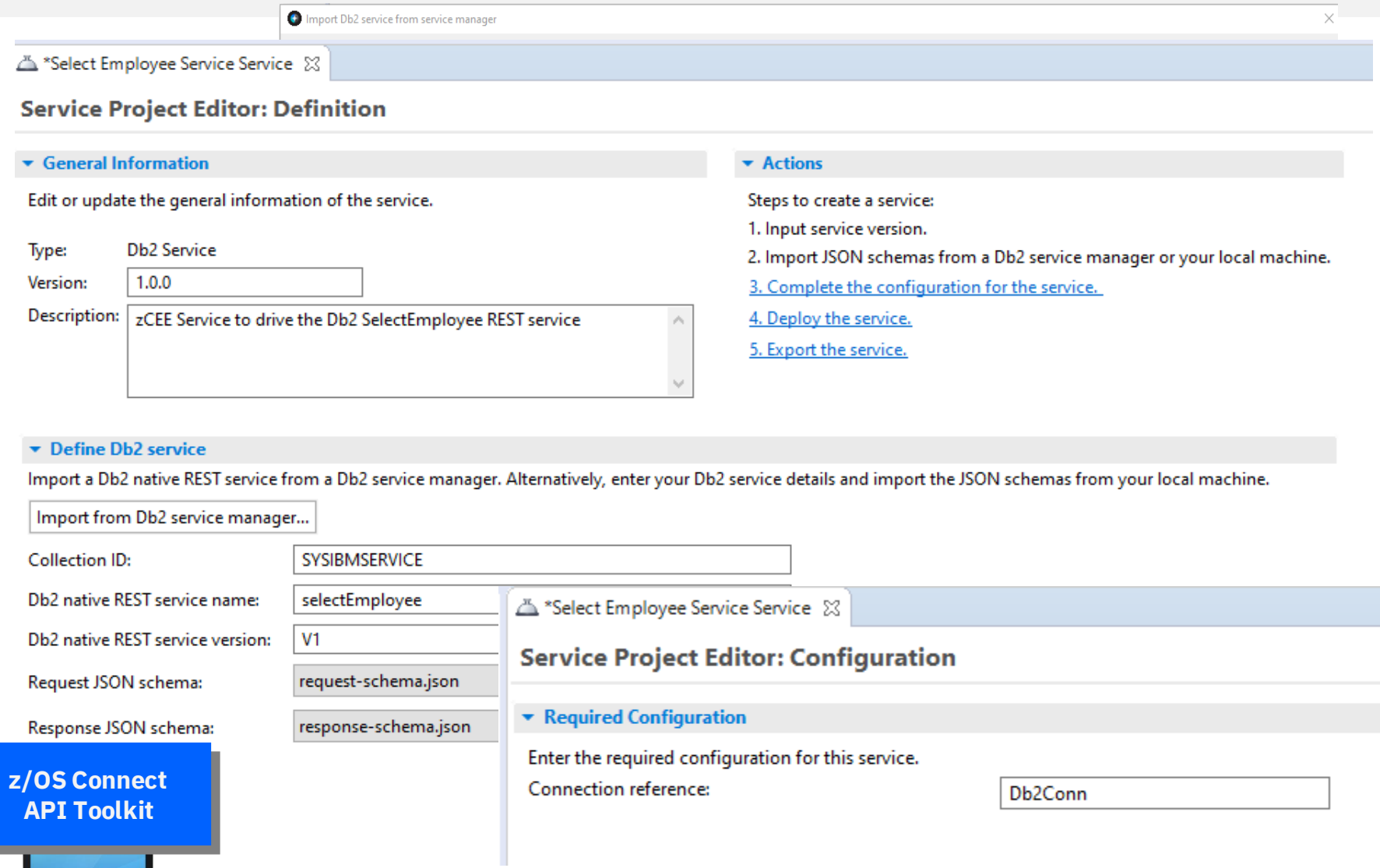
Services Definition

Dynamic API and Services Deployment

z/OS Connect Runtime Server

Service Projects & Service Creation

Import the Db2 Rest Service, define the service interface, & configure the service



1 Import the Db2 Rest Service

2 Request/Response schemas are automatically populated

3 Specify connection reference for the Db2 system

z/OS Connect API Toolkit

API Projects & API Creation

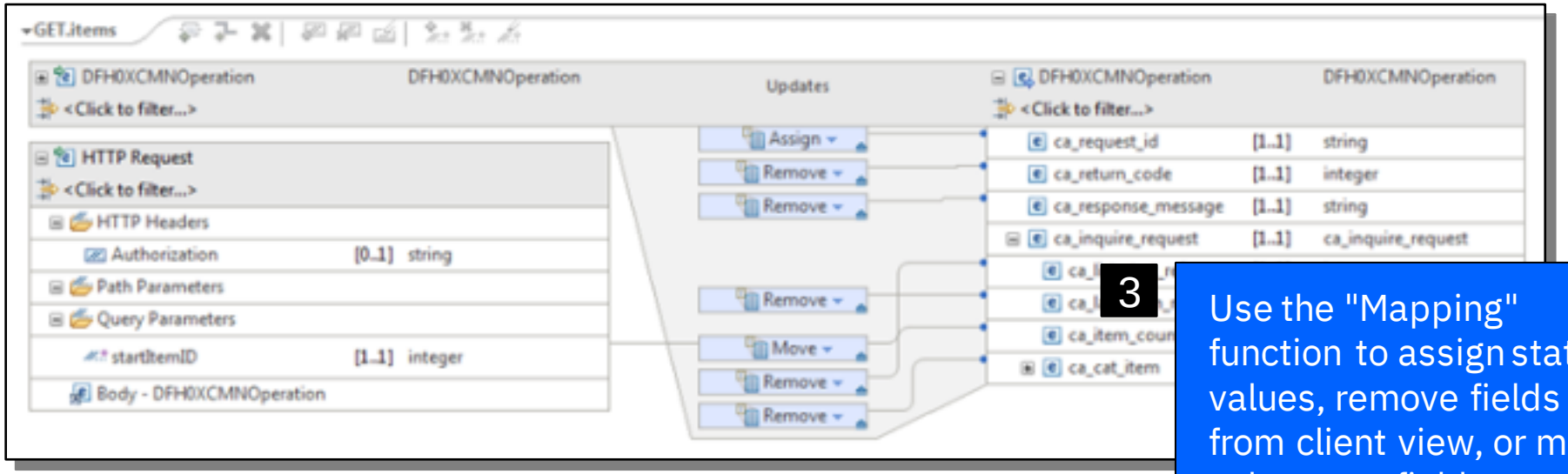
Define the URI path, HTTP verbs and JSON mappings for the API

1 Compose the API URI path

2 Select the HTTP verb, and map the call to the underlying service

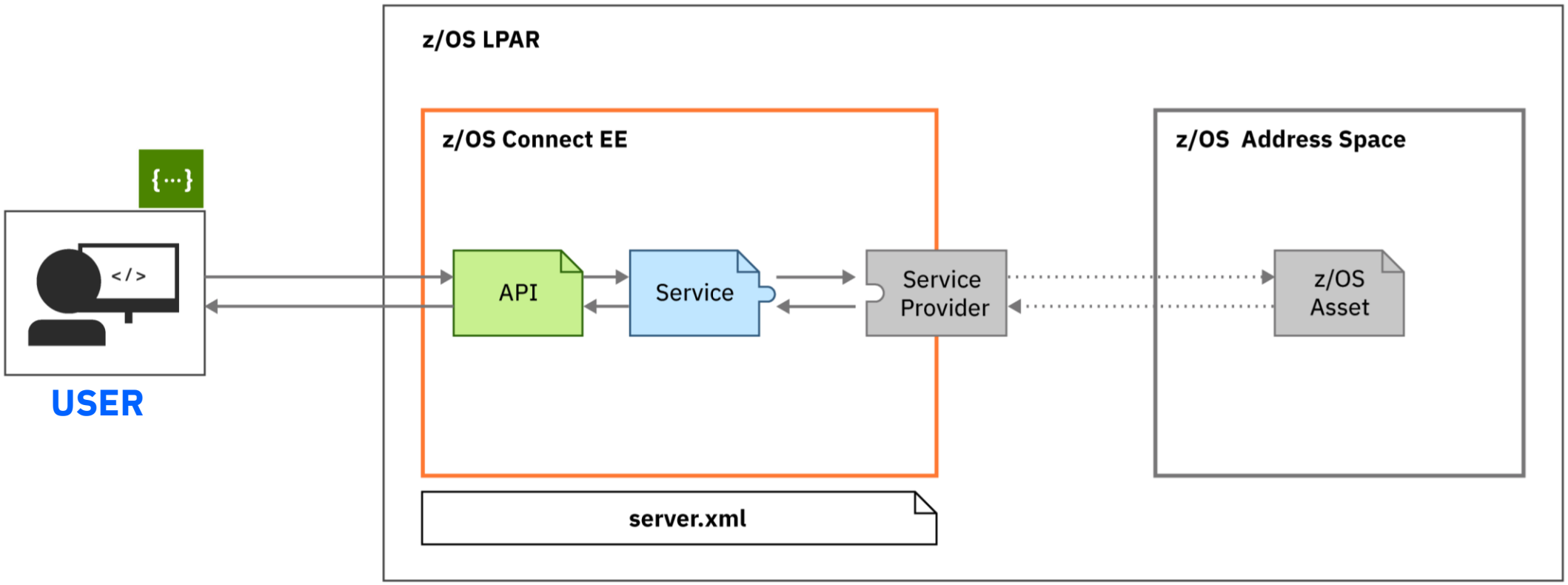
z/OS Connect API Toolkit

IBM Explorer for z/OS

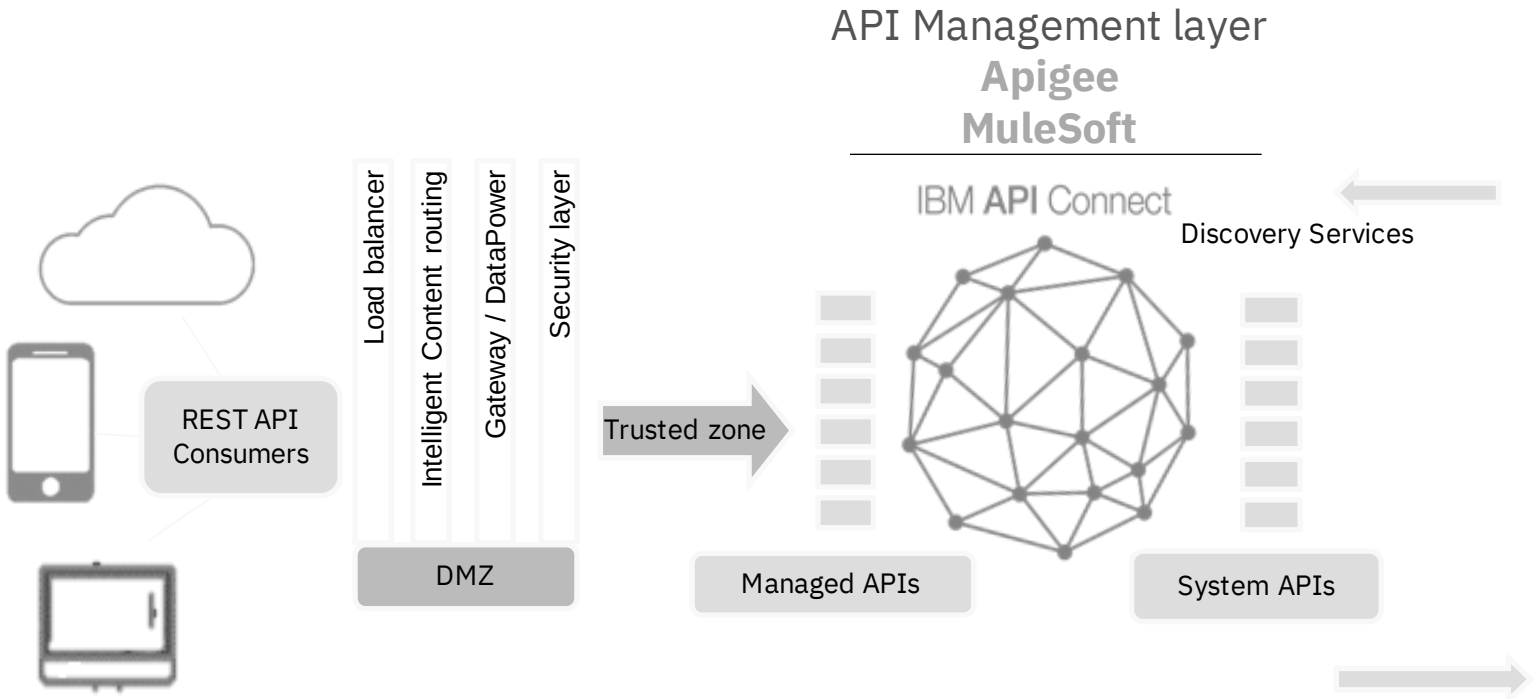


3 Use the "Mapping" function to assign static values, remove fields from client view, or move values to a field

z/OS Connect Runtime Concepts



z/OS Connect in the Big Picture



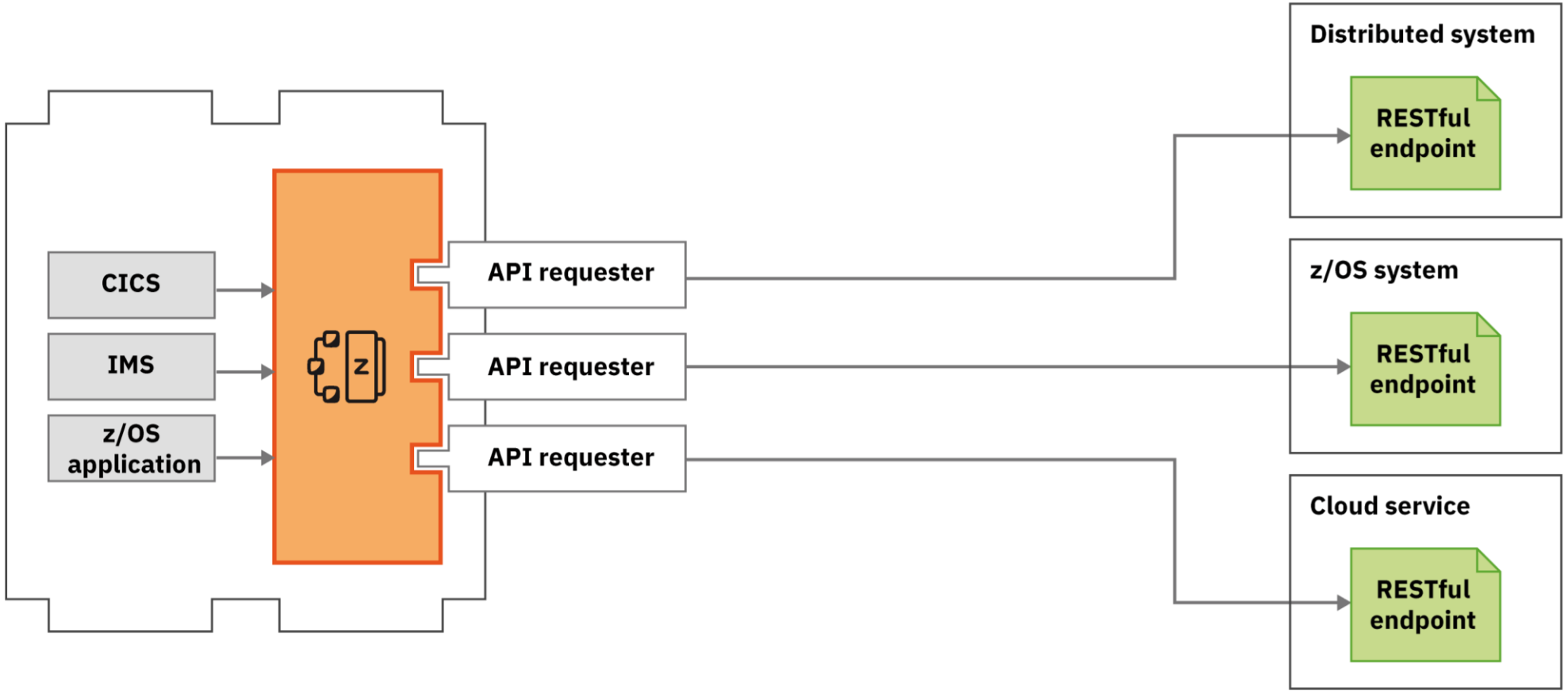
Catalog All Enterprise APIs
Chargeback / Billing
Engage Development Community

Manage Access to APIs
Manage Traffic
Build Composite APIs

Non-Z APIs

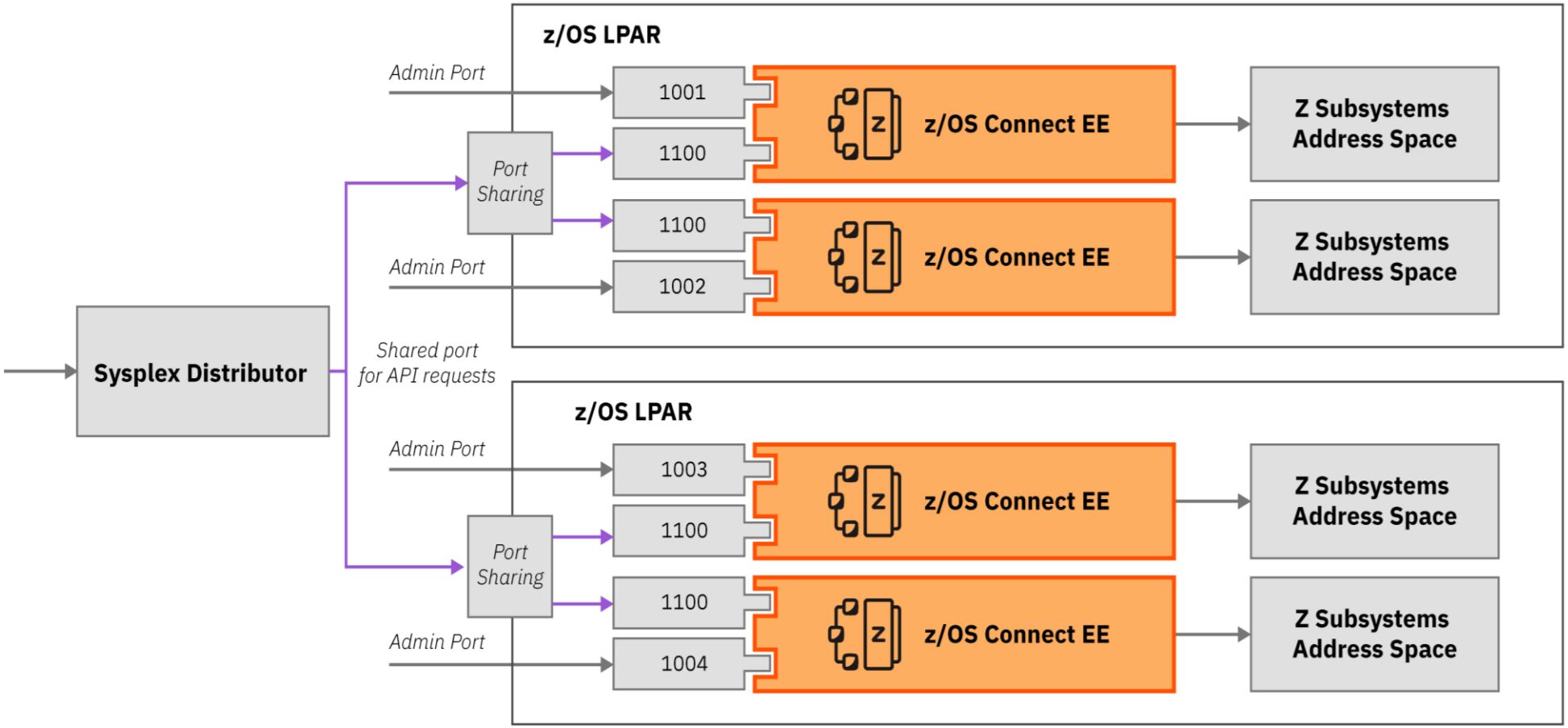
z/OS Connect EE API Requester

Mainframe applications can easily invoke RESTful APIs



ibm.biz/zosconnect-api-requester-overview

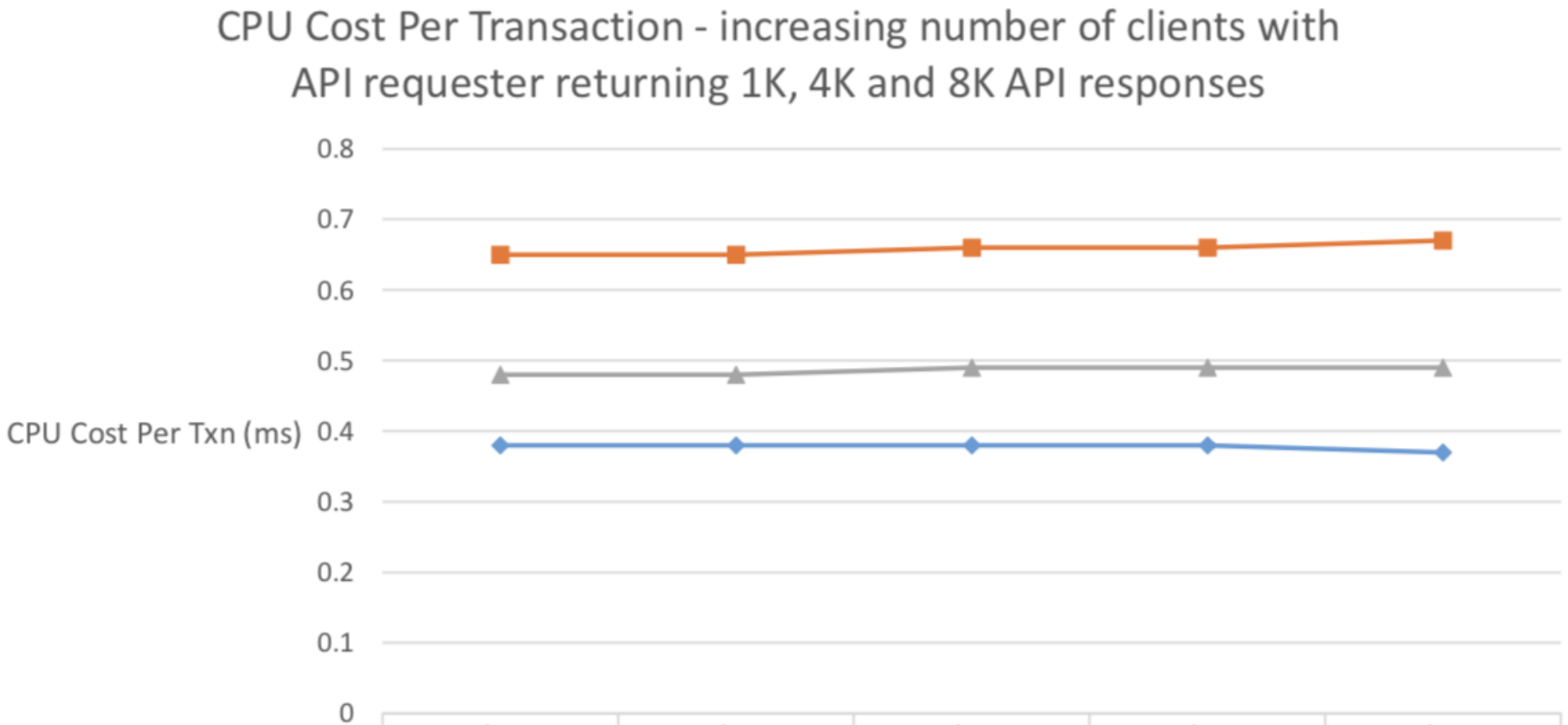
z/OS Connect in High Availability Topology



ibm.biz/zosconnect-ha-concepts

ibm.biz/zosconnect-scenarios

Performance: z/OS Connect scales with increased volumes



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

ibm.biz/zosconnect-performance-report

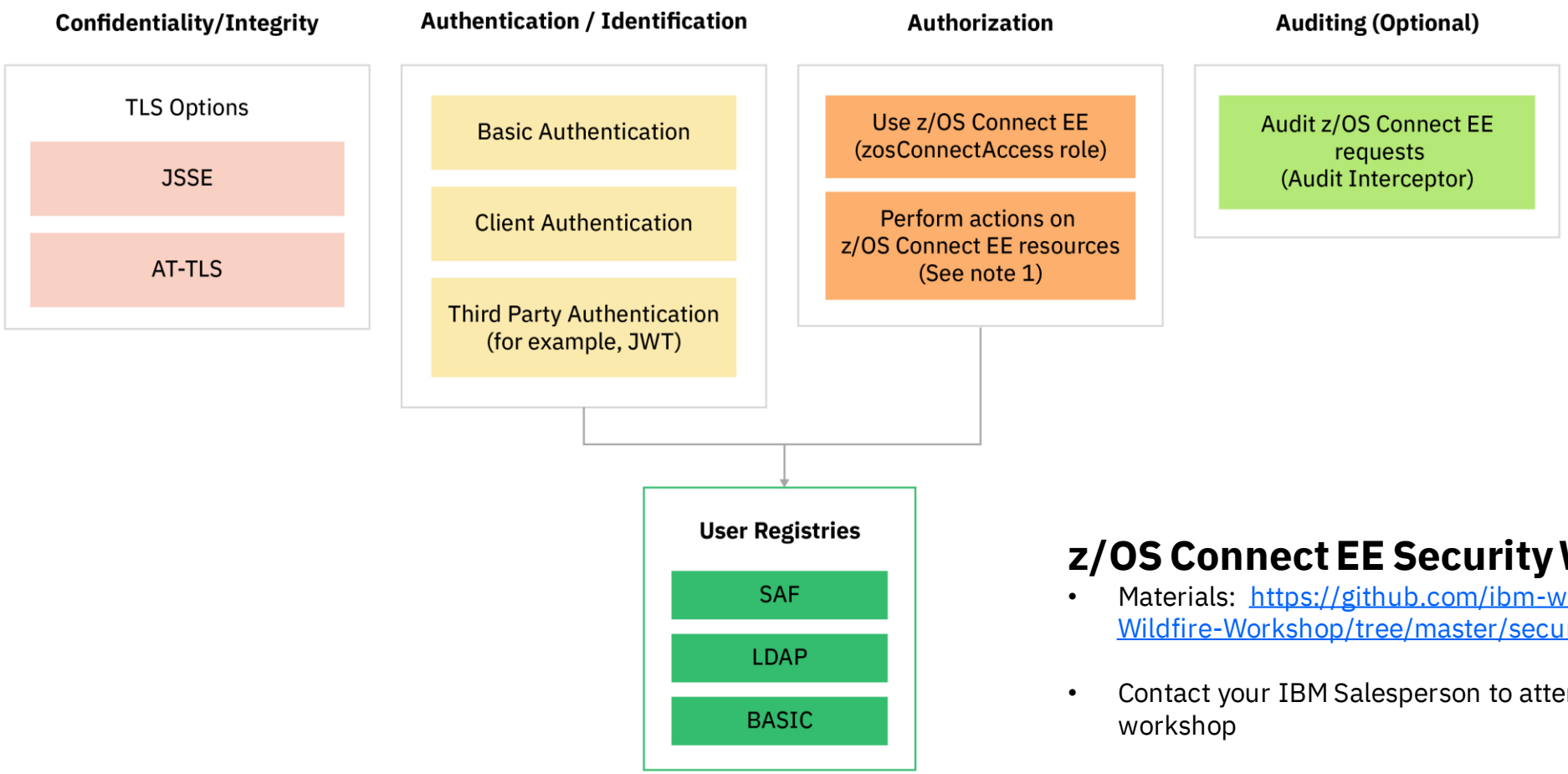
Performance: zIIP Eligibility

99% zIIP eligible

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

ibm.biz/zosconnect-performance-report

Security: High level options available in z/OS Connect



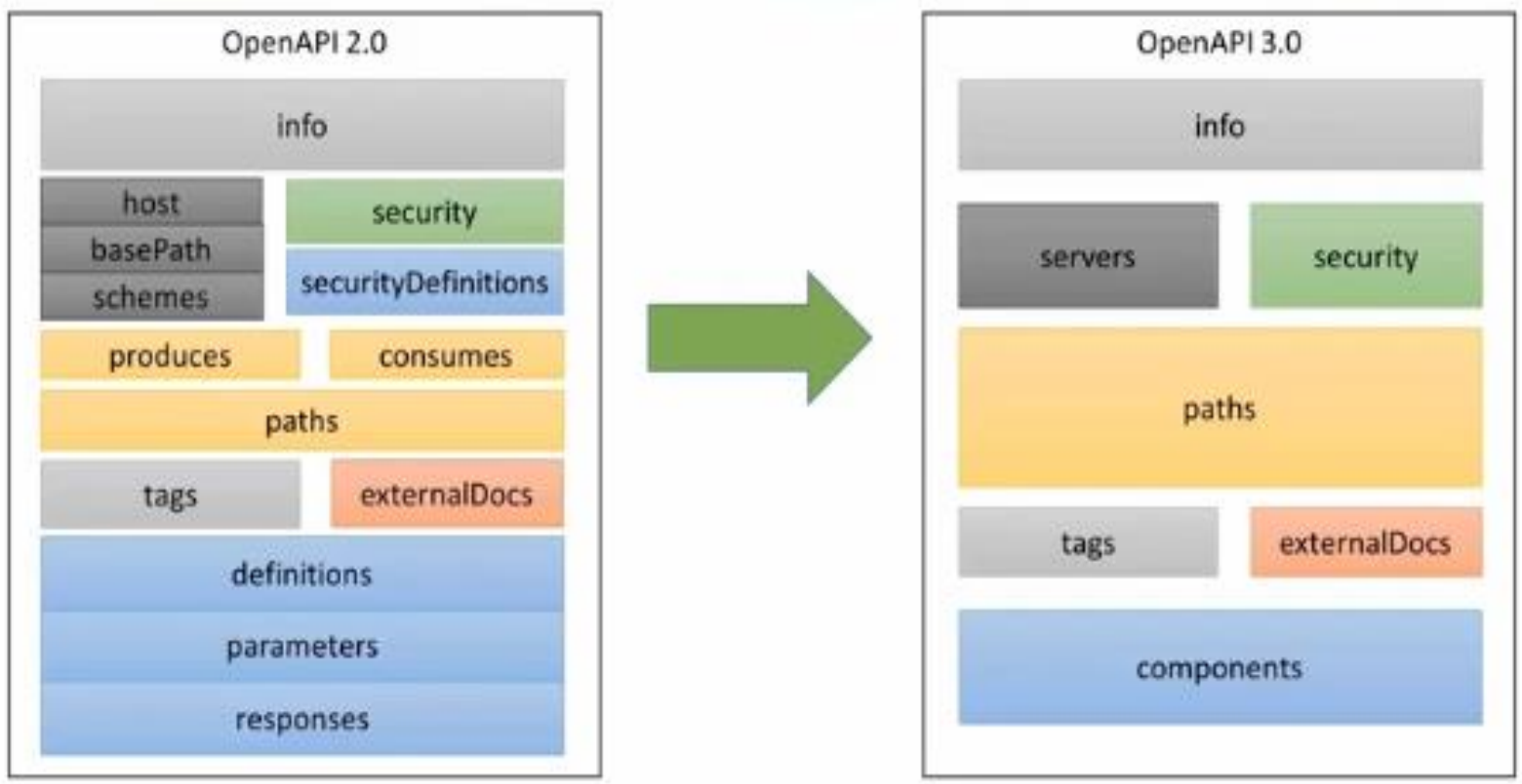
z/OS Connect EE Security Workshop

- Materials: https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop/tree/master/security
- Contact your IBM Salesperson to attend a future workshop

http://ibm.biz/zosconnect-security

z/OS Connect introduced support for OpenAPI 3.0 in March 2022

- z/OS Connect originally built for OpenAPI 2.0 (Swagger 2.0).
- The OpenAPI Specification 3.0 (OAS 3.0) has matured and has been widely adopted.
- OAS 3.0 is governed by the Linux foundation.
- The diagram to the right highlights the changes between the OpenAPI 2.0 and OpenAPI 3.0 specification structure.



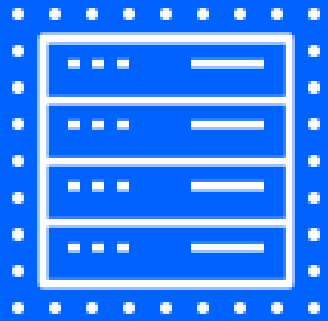
https://swagger.io/blog/news/whats-new-in-openapi-3-0/

z/OS Connect OpenAPI 3.0 Components

- z/OS Connect Designer container
- z/OS Connect Server container

z/OS Connect Designer

- OpenAPI 3 support
- API-first functional mapping
- Powerful new web-based user interface



z/OS Connect Server

- Role based security
- Server and its operators enable development teams to independently deploy and scale API integrations
- Near real time statistics to tooling such as Prometheus.

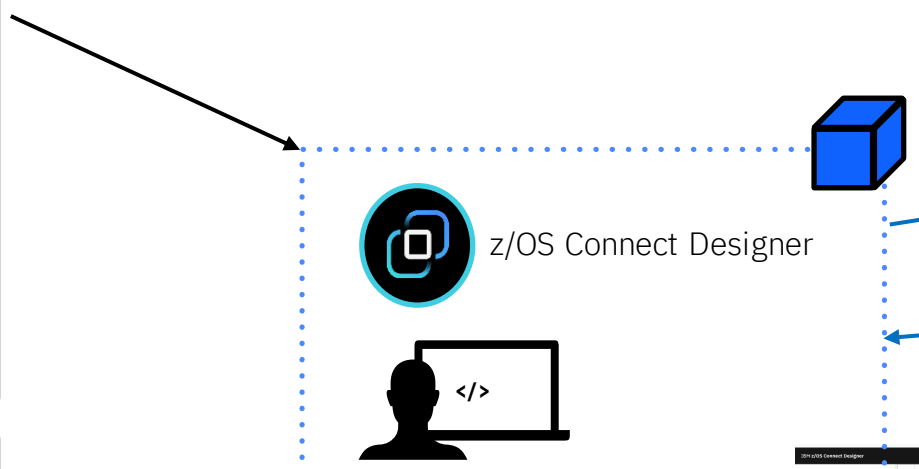
z/OS Connect API Creation for OAS 3.0

OpenAPI 3.0 Document

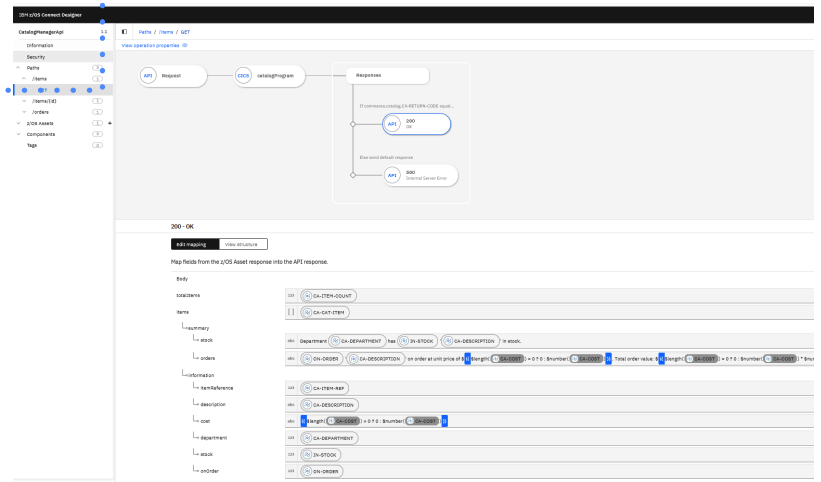
```yaml
openapi: 3.0.0
info:
  title: EmployeesApi
  description: Employee management API for Db2.
  version: "1.1"
  license:
    name: Apache-2.0
    url: https://opensource.org/licenses/Apache-2.0
servers:
- url: /
security:
- BasicAuth: []
- BearerAuth: []
paths:
  /employees/{id}:
    get:
      tags:
      - Edit
      - Discover
      summary: Get an employee
      description: Uses the getEmployee Db2 z/OS asset
```

Interface Copybook (for CICS)

```
*    Catalogue COMMAREA structure
     03 CA-REQUEST-ID            PIC X(6).
     03 CA-RETURN-CODE           PIC 9(2).
     03 CA-RESPONSE-MESSAGE      PIC X(79).
     03 CA-REQUEST-SPECIFIC      PIC X(911).
*    Fields used in Inquire Catalog
     03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
        05 CA-LIST-START-REF      PIC 9(4).
        05 CA-LAST-ITEM-REF       PIC 9(4).
        05 CA-ITEM-COUNT          PIC 9(3).
        05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
        05 CA-CAT-ITEM  REDEFINES CA-INQUIRY-RESPONSE-DATA
                        OCCURS 15 TIMES.
           07 CA-ITEM-REF         PIC 9(4).
           07 CA-DESCRIPTION      PIC X(40).
           07 CA-DEPARTMENT       PIC 9(3).
           07 CA-COST             PIC X(6).
           07 IN-STOCK            PIC 9(4).
           07 ON-ORDER            PIC 9(3).
*    Fields used in Inquire Single
     03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
        05 CA-ITEM-REF-REQ        PIC 9(4).
        05 FILLER                 PIC 9(4).
        05 FILLER                 PIC 9(3).
        05 CA-SINGLE-ITEM.
           07 CA-SNGL-ITEM-REF    PIC 9(4).
           07 CA-SNGL-DESCRIPTION PIC X(40).
           07 CA-SNGL-DEPARTMENT  PIC 9(3).
```



Redhat Code Ready workspace
or Docker workspace



CICS

Db2

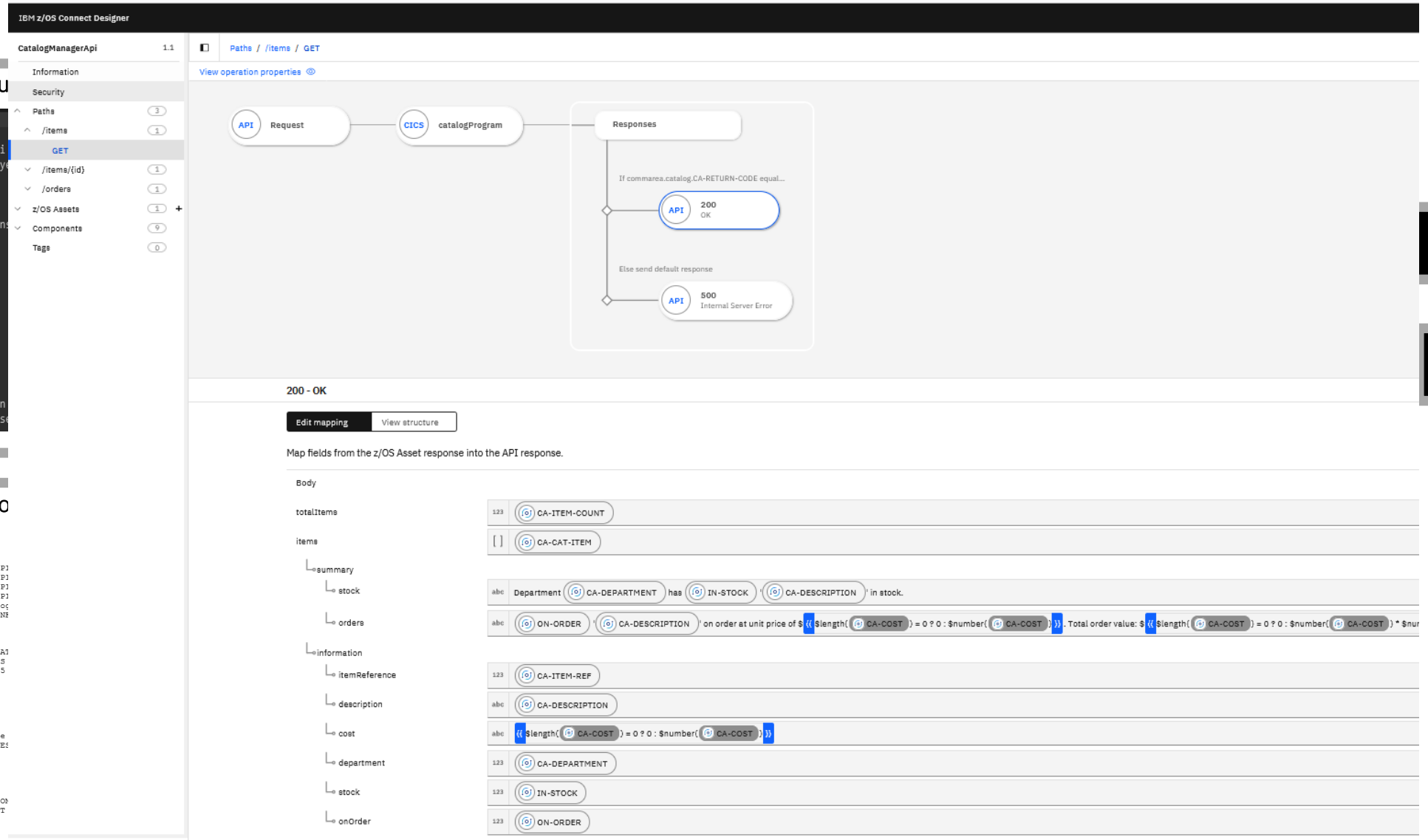
z/OS Connect OAS 3.0 Designer

OpenAPI 3.0 Docu

```
openapi: 3.0.0
info:
  title: EmployeesApi
  description: Employe
  version: "1.1"
  license:
    name: Apache-2.0
    url: https://opens
servers:
- url: /
security:
- BasicAuth: []
- BearerAuth: []
paths:
  /employees/{id}:
    get:
      tags:
      - Edit
      - Discover
      summary: Get an
      description: Us
```

Interface Copyboo

```
*    Catalogue COMMAREA structure
     03 CA-REQUEST-ID            PI
     03 CA-RETURN-CODE           PI
     03 CA-RESPONSE-MESSAGE      PI
     03 CA-REQUEST-SPECIFIC      PI
*    Fields used in Inquire Catalog
     03 CA-INQUIRE-REQUEST REDEFINE
          05 CA-LIST-START-REF
          05 CA-LAST-ITEM-REF
          05 CA-ITEM-COUNT
          05 CA-INQUIRY-RESPONSE-DAT
          05 CA-CAT-ITEM  REDEFINES
                         OCCURS 15
               07 CA-ITEM-REF
               07 CA-DESCRIPTION
               07 CA-DEPARTMENT
               07 CA-COST
               07 IN-STOCK
               07 ON-ORDER
*    Fields used in Inquire Single
     03 CA-INQUIRE-SINGLE REDEFINES
          05 CA-ITEM-REF-REQ
          05 FILLER
          05 FILLER
          05 CA-SINGLE-ITEM.
               07 CA-SNGL-ITEM-REF
               07 CA-SNGL-DESCRIPTION
               07 CA-SNGL-DEPARTMENT
```



CICS

Db2

API Creation – 2.0 vs 3.0

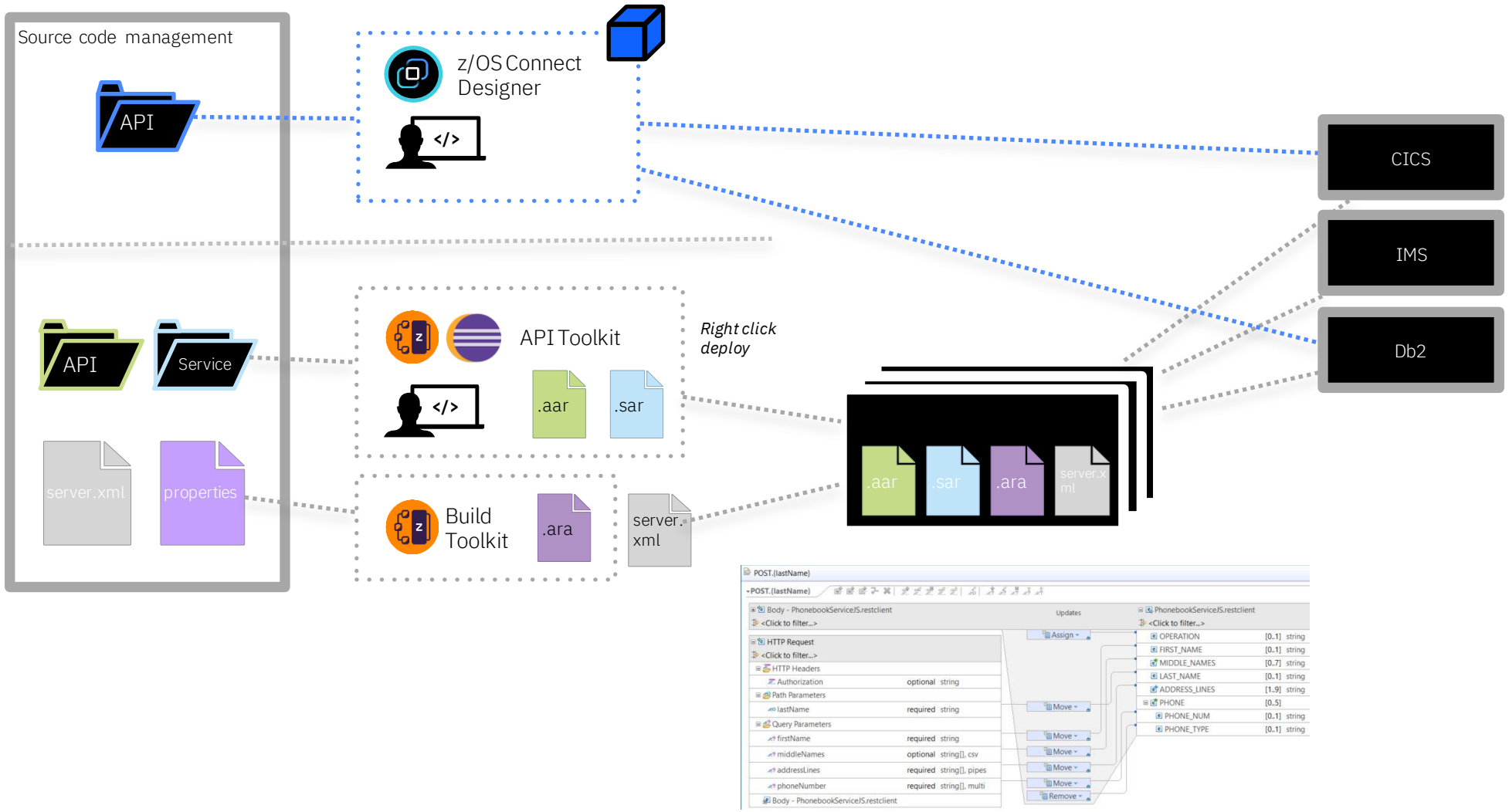
OpenAPI 3.0

- Use the web-based designer
- Designer runs in a container, so no need for a development server to test
- As of today, only support for Db2 and CICS in bound

OpenAPI 2.0

- Use Eclipse based API toolkit

z/OS Connect 2.0 APIs must run in separate servers from the z/OS Connect 3.0 APIs.



Db2 REST & z/OS Connect

Perfectly paired

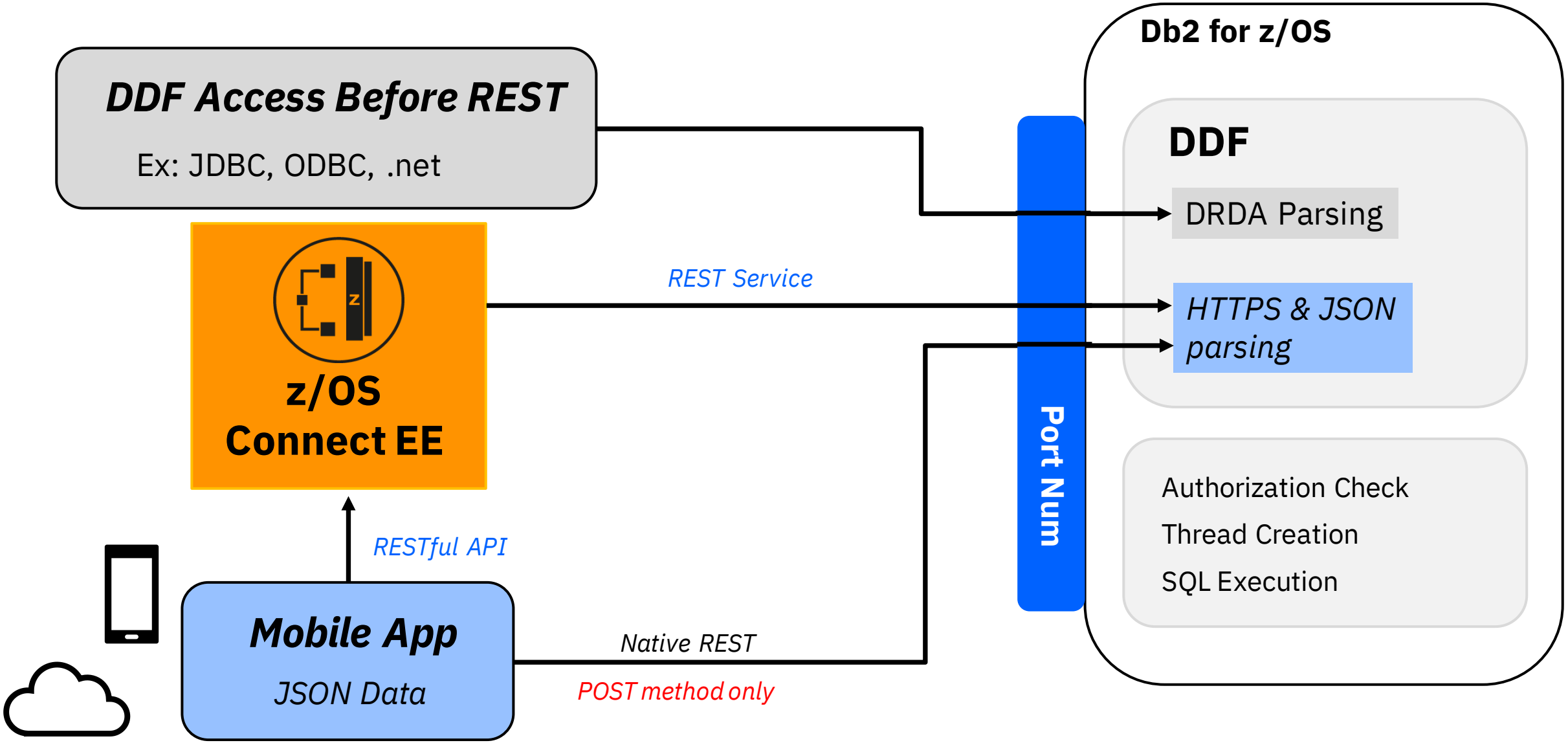
Db2 REST Services with z/OS Connect EE

Db2 user created native REST services are invoked by the *POST method only*

Mobile and cloud programmers following the RESTful API design model use the *HTTP Methods (Verbs): POST, GET, PUT and DELETE*

z/OS Connect Enterprise Edition's tool "API Editor" can map a **Db2 POST method SQL statement to the appropriate RESTful method** for a given behavior

Architecture Diagram



Db2 REST service using a stored procedure with select SQL statements (read only)

Db2 REST service

METHOD | POST

URI | http://wg31.washington.ibm.com:1446/services/SYSIBMSERVICE/selectByDeptSP

HEADERS | Content-Type = application/json and Accept: application/json

BODY | {"WHICHQUERY":1,"DEPT1":"A00"}

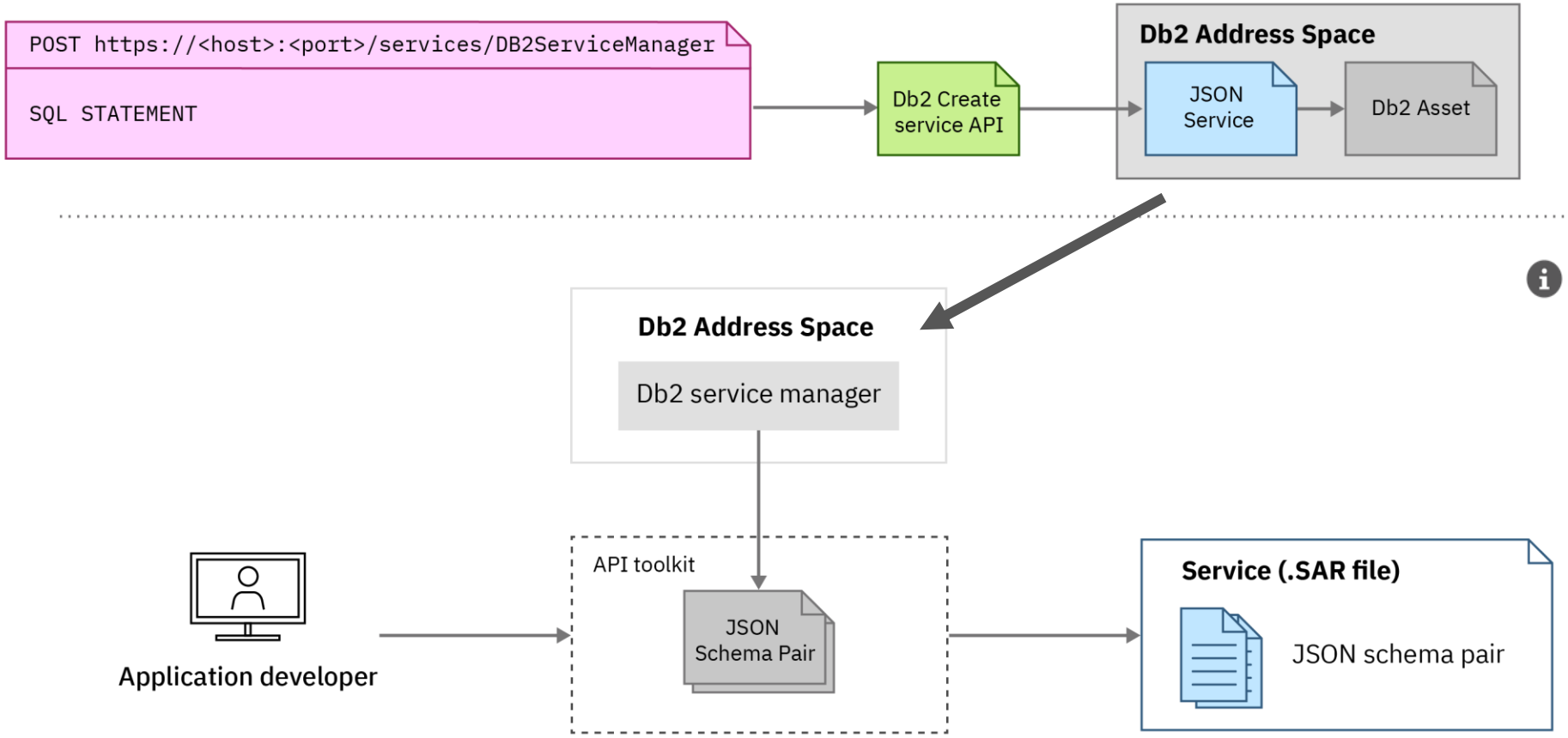
Db2 REST API using z/OS Connect

METHOD | GET

URI | https://wg31.washington.ibm.com:9446/employee/deptNo/A00

- GET method includes input properties within the URI
- API Editor-created constants reduce the number of input parameters
- Both REST statements produce the same output
- z/OS Connect example follows RESTful standard

Connecting to Db2 with z/OS Connect EE:
Create the service definition



Db2 Native REST Service creation
ibm.biz/zosconnect-db2-rest-services

zCEE Db2 native REST service demo
https://www.ibm.com/demos/search/?products=%22IBM%20z%2FOS%20Connect%22&lc=en

Running on REST:
New access with native REST services

The Customer:

A large US manufacturer

Business Challenge:

The company needed a simple portal to navigate Db2 for z/OS

Their Need:

The manufacturer needed an easy access point to navigate around Db2 for z/OS.

Our Solution:

Db2 native REST services were used to create a web-browser UI and tool for interacting with Db2 for z/OS.

Customer Benefit:

All DBAs, new and experienced, could smoothly navigate Db2 for z/OS using a familiar interface.

APIs deliver peace of mind to customers during a global crisis

The Customer:

A large automotive company

Business Challenge:

The automotive company wanted to rapidly automate their manual process for loan extensions.

Their Need:

The company needed to rapidly automate their manual loan extension request process to handle at speed the huge increase of finance extension requests (19,000 per day) driven by the COVID-19 outbreak.

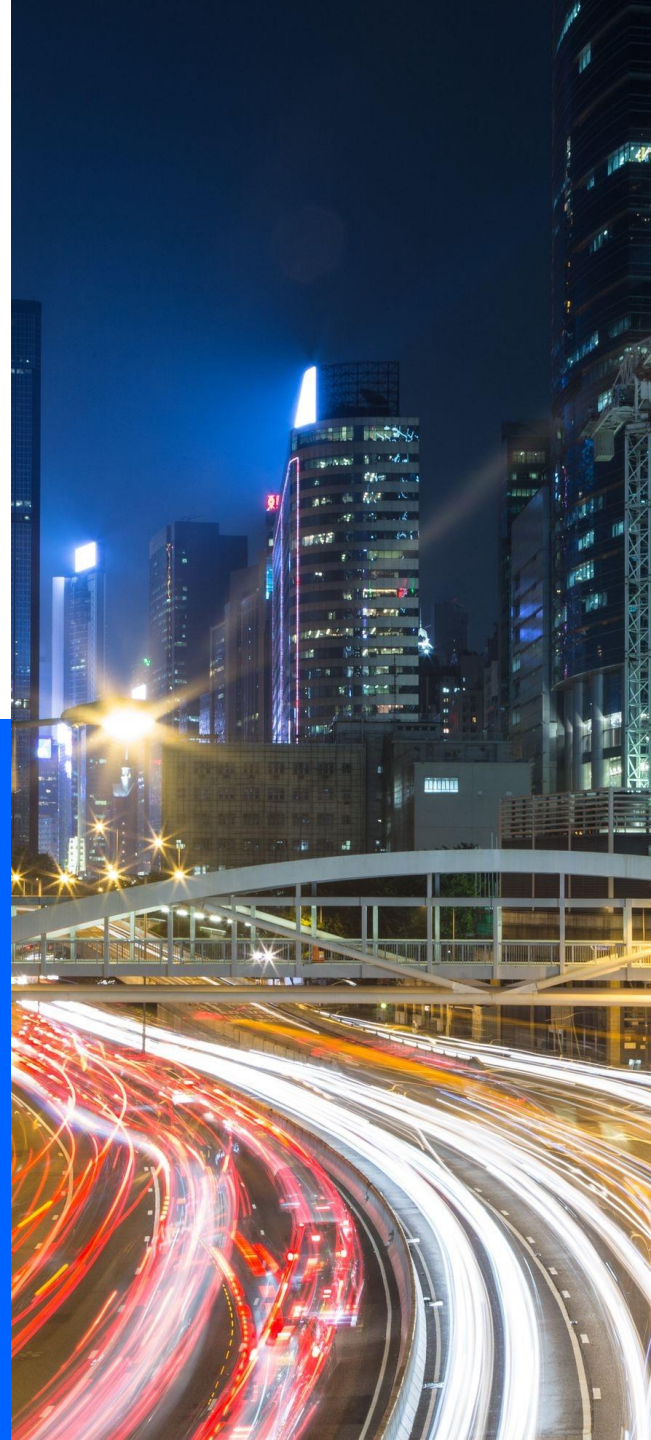
Our Solution:

IBM z/OS Connect EE generated REST APIs that could be easily called from applications on any platform.

Kubernetes®, microservices and z/OS Connect EE API enablement enabled unprecedented acceleration to create new user experiences owned by multiple groups within the enterprise.

Customer Benefit:

Leveraging on z/OS Connect EE's APIs to rapidly innovate processes in days/weeks. Digitization of business processes enables customer needs to be met and the business the capacity to focus recourses on processes requiring manual customization.



Additional Workshops

IBM offers several workshops related to JSON and REST enablement of z assets:

- Db2 for z/OS: REST and Hybrid Cloud Workshop
- IBM z/OS Connect Wildfire Workshop
- IBM z/OS Connect EE Security Wildfire Workshop
- Db2 12 Technology Update Workshop

Notify your IBM representative if you are interested in any of these workshops.

Look at the following link for more events
https://ibm-zcouncil.com/events/

Lab Time

WARNING!

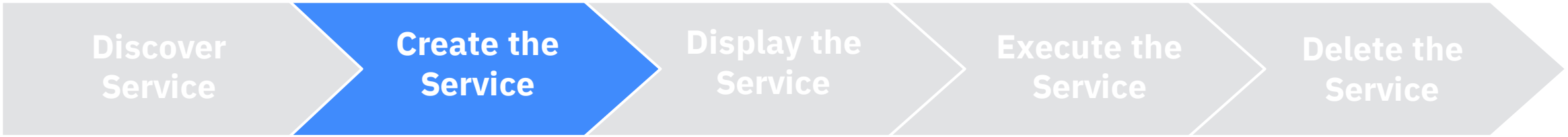
Before beginning, understand that everything you will be working with is mixed case sensitive.

You can very easily lose days trying to resolve a problem because a case on just one character was not set correctly.

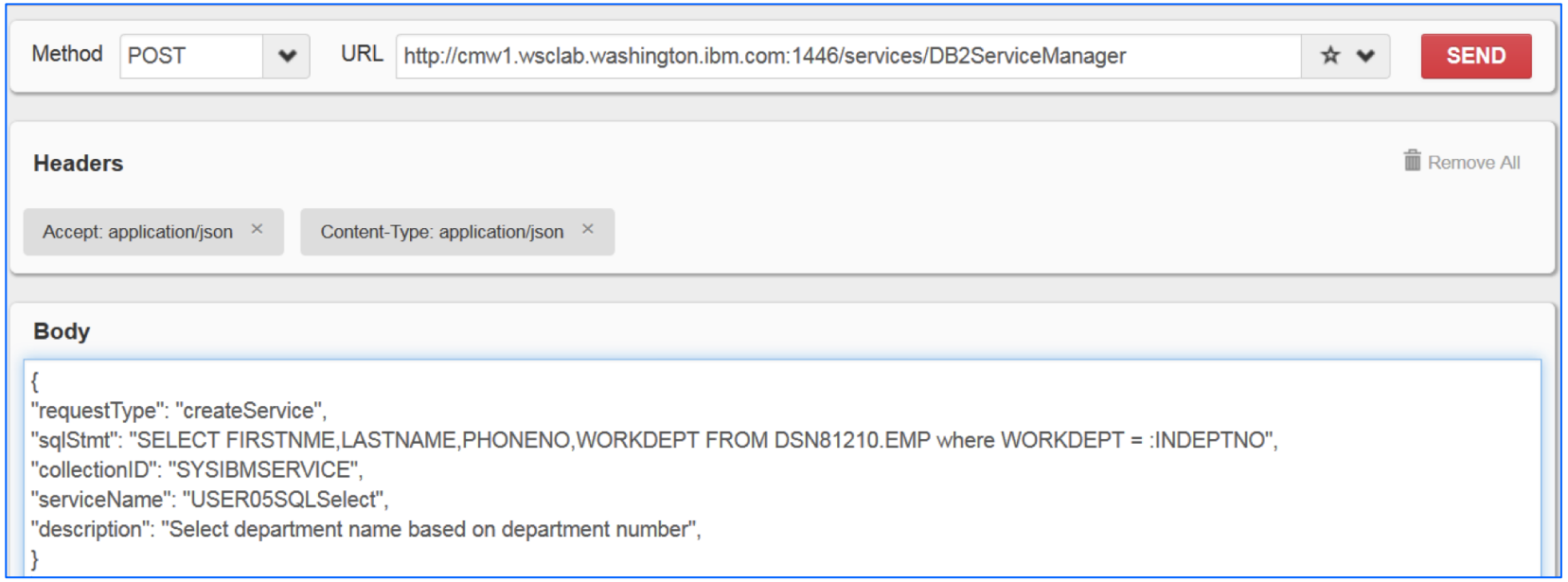
NativeREST <> NativeRESt <> NATIVEREST <> nativerest

Appendix

Db2 REST Service Progression



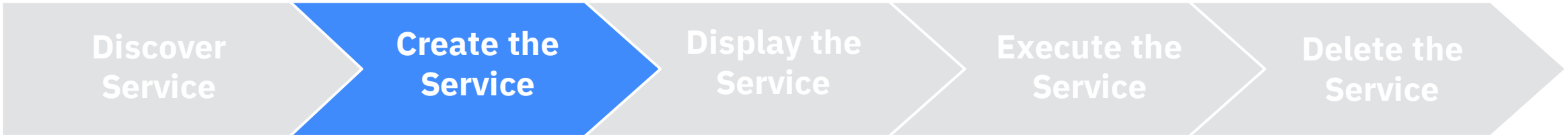
Creating a Db2 REST service – **example SQL statement**



Note: You can also use the IBM Data Studio client to create a Db2 REST service, but Db2 z/OS SSL MUST be operational.

Status Code 201 indicates successful creation

Db2 REST Service Progression



Creating a Db2 REST service – **stored procedure (SP)**

Db2 Stored Procedure CALL statement variables:

CALL USER01.USER01EMPL_DEPTS_NAT(?,?,?)

"?" – Question mark(s) can be used as input variable placeholder; Db2 will replace "?"s with P1, P2, … in JSON request.

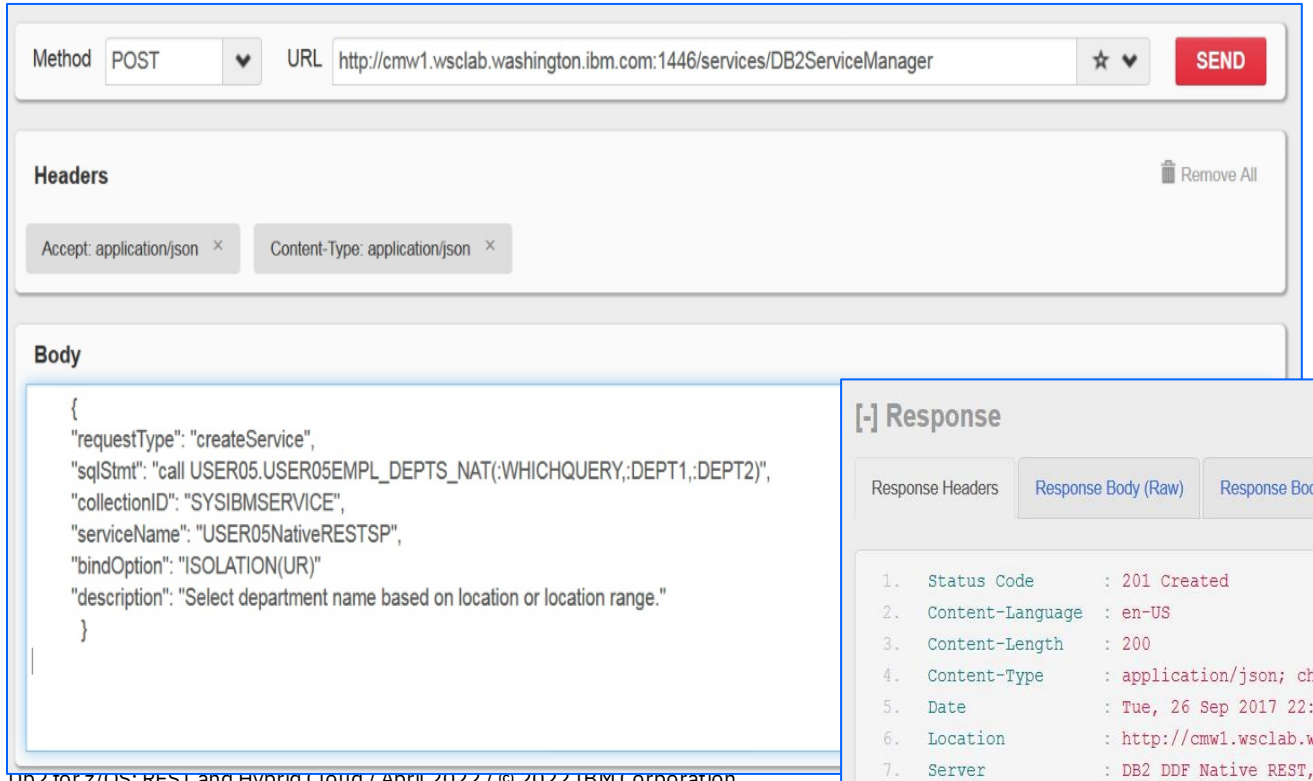
CALL USER01.USER01EMPL_DEPTS_NAT(:WHICHQUERY,:DEPT1,:DEPT2)

Input variable labels "WHICHQUERY", "DEPT1" and "DEPT2" are created manually, and will be used in the JSON request.

Db2 REST Service Progression

Discover Service | **Create the Service** | Display the Service | Execute the Service | Delete the Service

Creating a Db2 REST service – **example stored procedure (SP)**



Note:
Db2 stored procedure call statement host input variables were manually created.

Default host variable name Px, where x = variable number

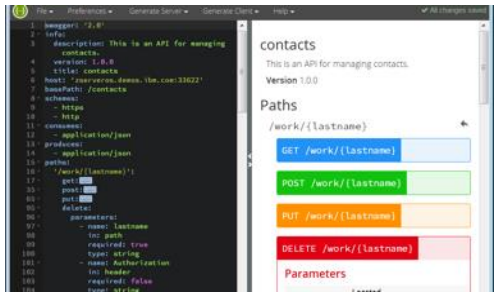
The Open API Initiative

The industry standard framework for describing RESTful APIs, aka **Swagger**

There are a variety of tools available to aid consumption:

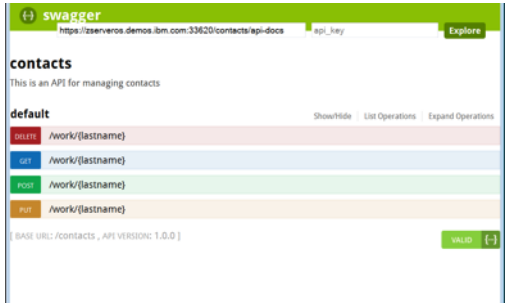
Write Swagger

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



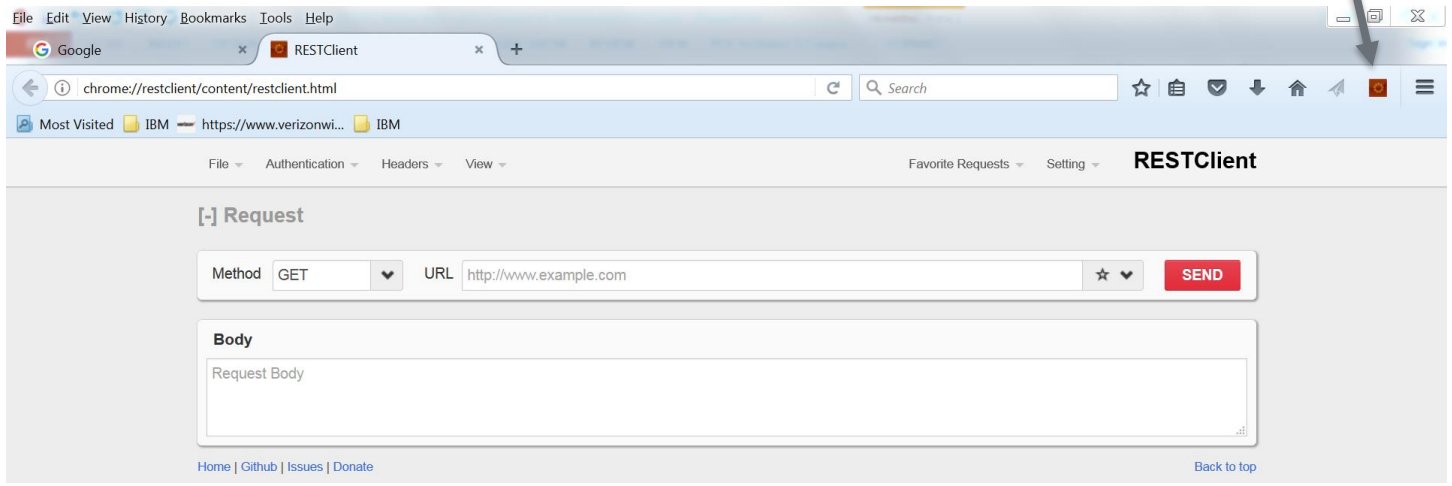
Consume Swagger

[GitHub Link]

Browser REST Client extensions

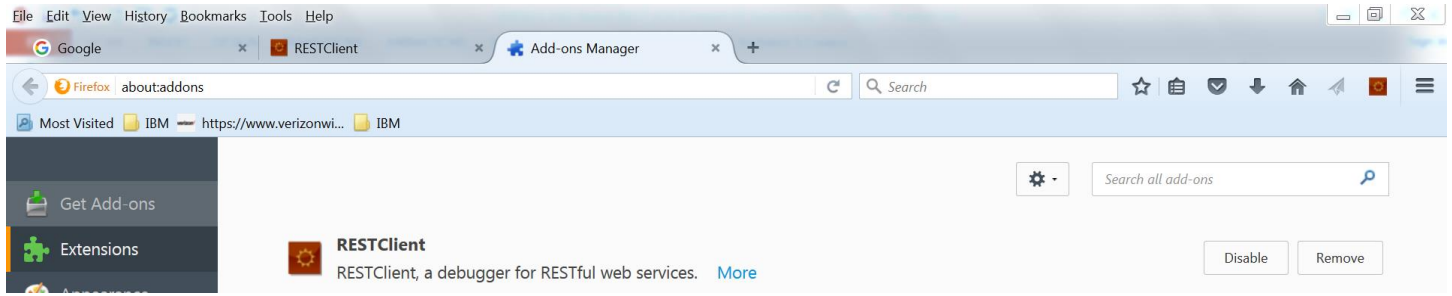
Various browsers provide a REST client extension.

In our example, we use FireFox with RESTClient.



In Firefox, go to add-ons and search for RESTClient.

It will add it as an extension and will be part of your screen



Example: using Db2 IVP data to create callable SP

```
CREATE PROCEDURE USER05EMPL_DEPTS_NAT
     (IN WHICHQUERY INTEGER, IN DEPT1 CHARACTER(3), IN DEPT2 CHARACTER(3))
VERSION V1
           RESULT SETS 1
     LANGUAGE SQL
     ISOLATION LEVEL CS
     DISABLE DEBUG MODE
P1: BEGIN
DECLARE CURSOR1 CURSOR WITH RETURN FOR
     SELECT EMPLOYEE.EMPNO, EMPLOYEE.FIRSTNME, EMPLOYEE.MIDINIT,
     EMPLOYEE.LASTNAME,
     EMPLOYEE.WORKDEPT, EMPLOYEE.PHONENO
            FROM DSN81210.EMP AS EMPLOYEE
            WHERE EMPLOYEE.WORKDEPT=DEPT1
            ORDER BY EMPLOYEE.EMPNO ASC;
DECLARE CURSOR2 CURSOR WITH RETURN FOR
     SELECT EMPLOYEE.EMPNO, EMPLOYEE.FIRSTNME, EMPLOYEE.MIDINIT,
     EMPLOYEE.LASTNAME,
     EMPLOYEE.WORKDEPT, EMPLOYEE.PHONENO
            FROM DSN81210.EMP AS EMPLOYEE
         WHERE EMPLOYEE.WORKDEPT>=DEPT1 AND EMPLOYEE.WORKDEPT<=DEPT2
            ORDER BY EMPLOYEE.WORKDEPT ASC, EMPLOYEE.EMPNO ASC;
CASE WHICHQUERY
    WHEN 1 THEN
        OPEN CURSOR1;
    ELSE
        OPEN CURSOR2;
END CASE;
END P1#
```

If WHICHQUERY=1, SELECT one department only – DEPT1 is the department

If WHICHQUERY=2, SELECT multiple departments – from DEPT1 until DEPT2

For Db2 REST - new catalog table created in installation job DSNTIJRS

The SYSIBM.DSNSERVICE table contains rows that describe Db2 REST services and their corresponding packages.

The following table describes the columns in table SYSIBM.DSNSERVICE:

Column name	Data type	Description
NAME	VARCHAR(128) NOT NULL	Name of the package that contains the service request.
COLLID	VARCHAR(128) NOT NULL	Name of the collection that contains the package.
CONTOKN	CHAR(8) NOT NULL FOR BIT DATA	Consistency token for the package that is generated when the service is created or altered.
ENABLED	VARCHAR(128) NOT NULL	Indicates whether service is enabled: Y - Service is enabled, which is the default setting. N - Service is disabled.
CREATETS	TIMESTAMP NOT NULL	The time when the row is inserted.
ALTEREDTS	TIMESTAMP NOT NULL	The time when the row is last updated.
DESCRIPTION	VARCHAR(250)	A user-specified character string.

Db2 also sets the HOSTLANG column in the SYSIBM.SYSPACKAGE and SYSIBM.SYSPACKCOPY tables to 'R' to mark the package for the REST API

FAQ: What does Stateless mean?

REST itself is **stateless**, meaning that every request/reply is independent from the next.

In Db2 terms, every request/reply is a complete transaction (commit, unless error, then abort), and all of the database locks/resources are freed.

If the request is a SELECT, the entire result set is returned and closed as part of the reply.

For example, you couldn't open a cursor/SELECT in one request, and then try to do a positioned update on that cursor in another request.

FAQ: What type of stored procedures can I use for Db2 REST?

Any type of stored procedure can be used.

Batch TMP alternative for CREATE and FREE

```
//JOBLIB  DD  DISP=SHR,
//            DSN=DB2M.SDSNLOAD
//DSNTIRU EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//DSNSTMT  DD  DISP=SHR,DSN=JOHNICZ.JCL(CRES)
//* NOTE - DSNSTMT CAN ALTERNATELY USE DD * WITH A STATEMENT SUCH AS
//* CALL EMPL_DEPTS_NAT(:WHICHQUERY,:DEPT1,:DEPT2)
//SYSTSIN  DD  *
  DSN SYSTEM(DB2M)
  BIND SERVICE(SYSIBMSERVICE) -
   NAME("SERVICE1") -
   SQLENCODING(1047) -
   DESCRIPTION('RETURN A LIST OF DEPTNAME-
  BASED ON INPUT LOCATION')
/*
```

```
CALL EMPL_DEPTS_NAT(:WHICHQUERY,:DEPT1,:DEPT2)                  00000010
```

```
//JOBLIB   DD   DISP=SHR,
//              DSN=DB2M.SDSNLOAD
//DSNTIRU  EXEC  PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT  DD   SYSOUT=*
//SYSPRINT  DD   SYSOUT=*
//SYSUDUMP  DD   SYSOUT=*
//SYSTSIN   DD   *
   DSN  SYSTEM(DB2M)
   FREE  SERVICE("SYSIBMSERVICE"."SERVICE1")
/*
```

DSNSTMT input cannot include numbers at the end of the statement. Create or Free will fail. Use NUM OFF.

The Other Connects

A little clarity on what does what

DB2 Connect

Provides ODBC/JDBC access to DB2-housed data.

Clients/users would use SQL to formulate requests

No REST access

IMS Connect

THE way to reach IMS Subsystem

OTMA client that provides TCP/IP connectivity to IMS applications/data.

Local access for WAS on z/OS

No REST access

App Connect

Formerly known as IIB or Message Broker

Any-any-connectivity between entities

Orchestration capability

REST access is possible

Typically requires specialist skills

API Connect & z/OS Connect

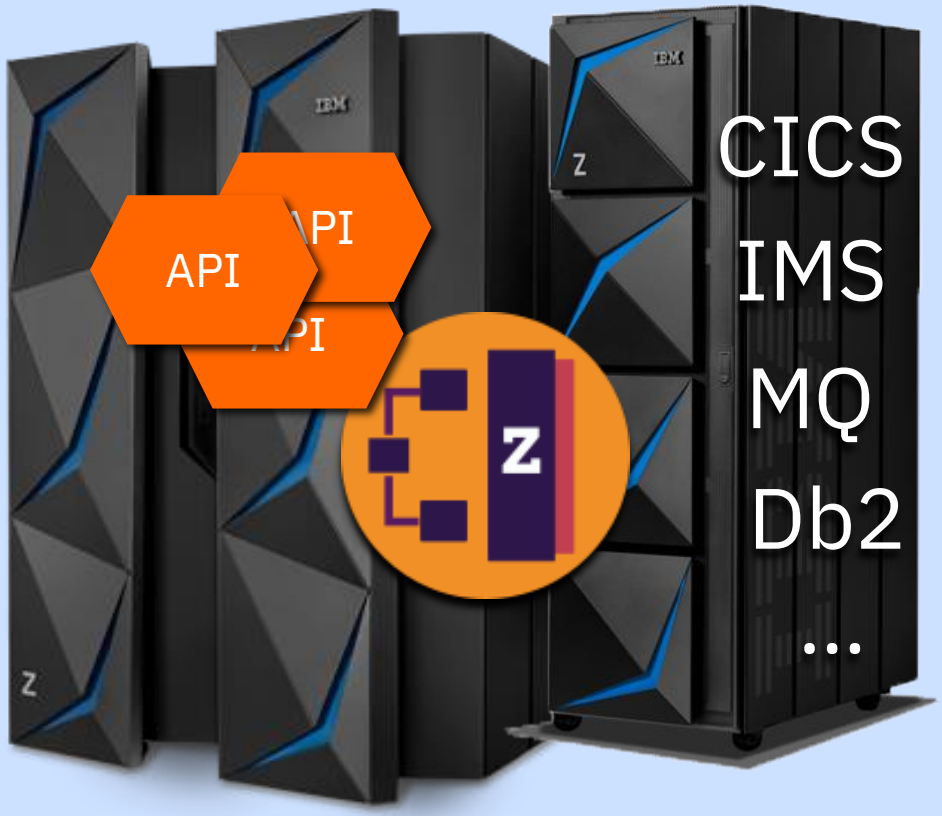
IBM apiconnect

Create APIs and microservices that consume IBM Z APIs

Manage and secure IBM Z APIs created by z/OS Connect

Comprehensive tooling that enables API developers to create RESTful APIs from z/OS-based assets

Delivers APIs as a discoverable resource using the OpenAPI specification (formerly Swagger)

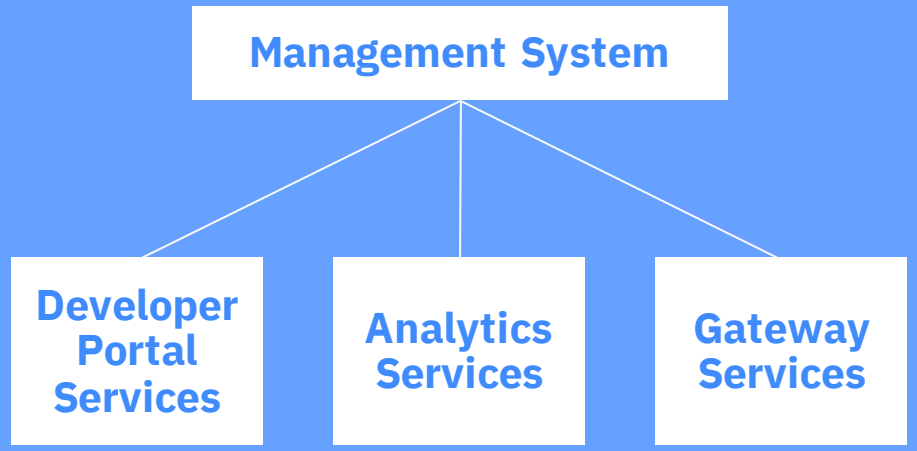


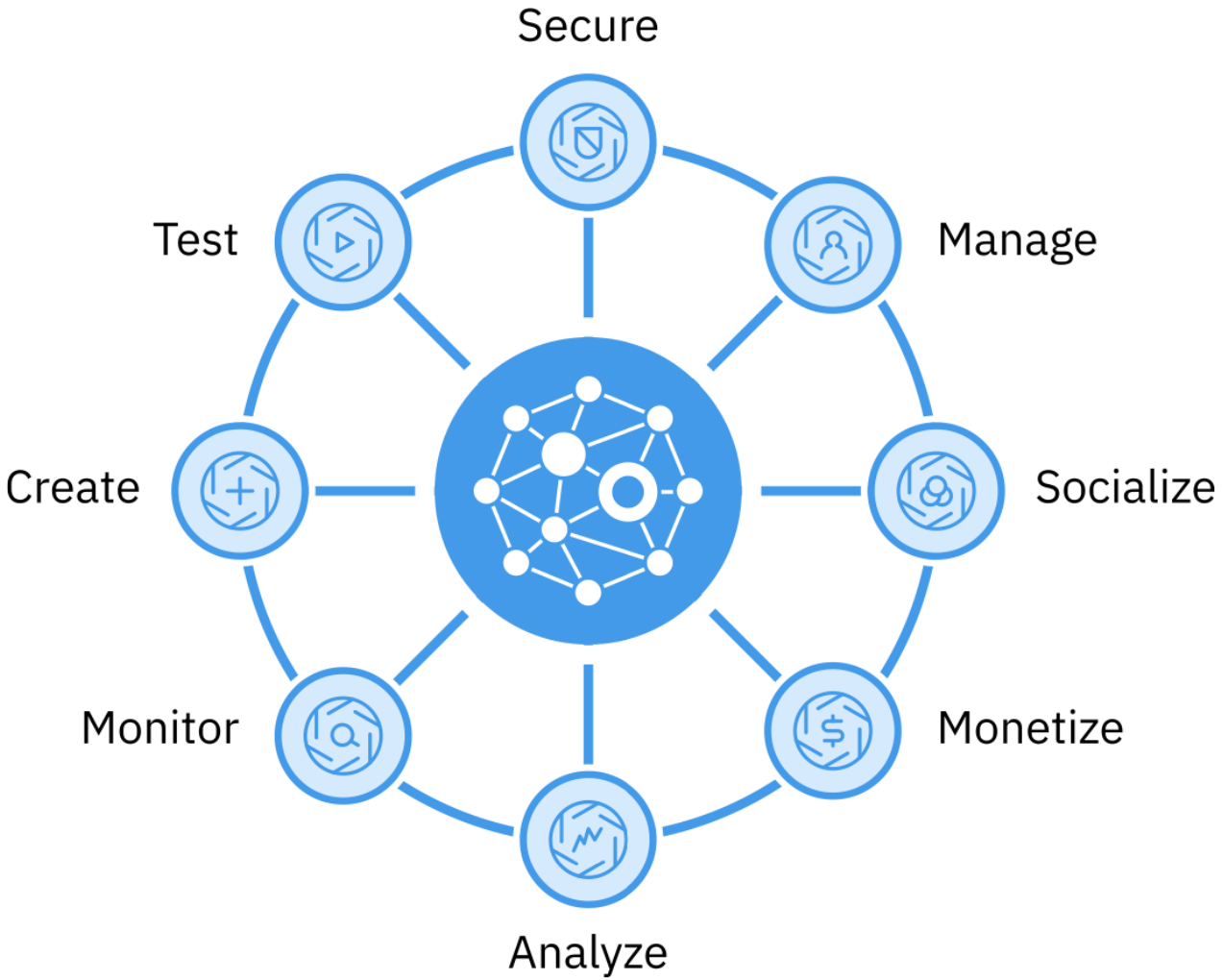
IBM API Connect

The Scalable Multi-Cloud API Platform

A complete, modern and intuitive API lifecycle platform to create, securely expose and manage APIs across clouds to power digital applications

API Connect Components





z/OS Connect EE vs. Db2 Connect

z/OS Connect EE

✓REST APIs are simple

✓Minimum business logic on client

✓No SQL skills needed

✓APIs are more consistent

✓Widespread acceptance

✓Supports Mobile platforms

✓Stateless

Db2 Connect

✓Contains Complex Business logic

✓SQL Skills required

✓Better Isolation

✓Transaction Processing

✓Resource Pooling

✓Sysplex Scalability

✓Transaction Fault-tolerance

Further Use Cases

Scale

Large US Bank

Serves **150 million API requests** per day from z/OS Connect EE to support fintech startups

Speed

European Bank

Reduced API development time from **3 months to less than a day**

Time to value

Australian Bank

Transformed their core banking application with APIs on Z in **half the time and for a fraction of the cost**

ROI

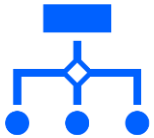
Financial organisation

Savings account creation from 3 days to less than a second through APIs resulting in over **5000 new accounts and $150m** in deposits within 3 months

Expanding Z

Spanish Insurance

Called an external vehicle lookup API from CICS to provide quick quotes based on just registration number. Resulted in 30% more conversions from their quotation website



Simplification

UK Bank

Removed 60% of the time, effort and money required to integrate PSD2 APIs with their core banking system on Z