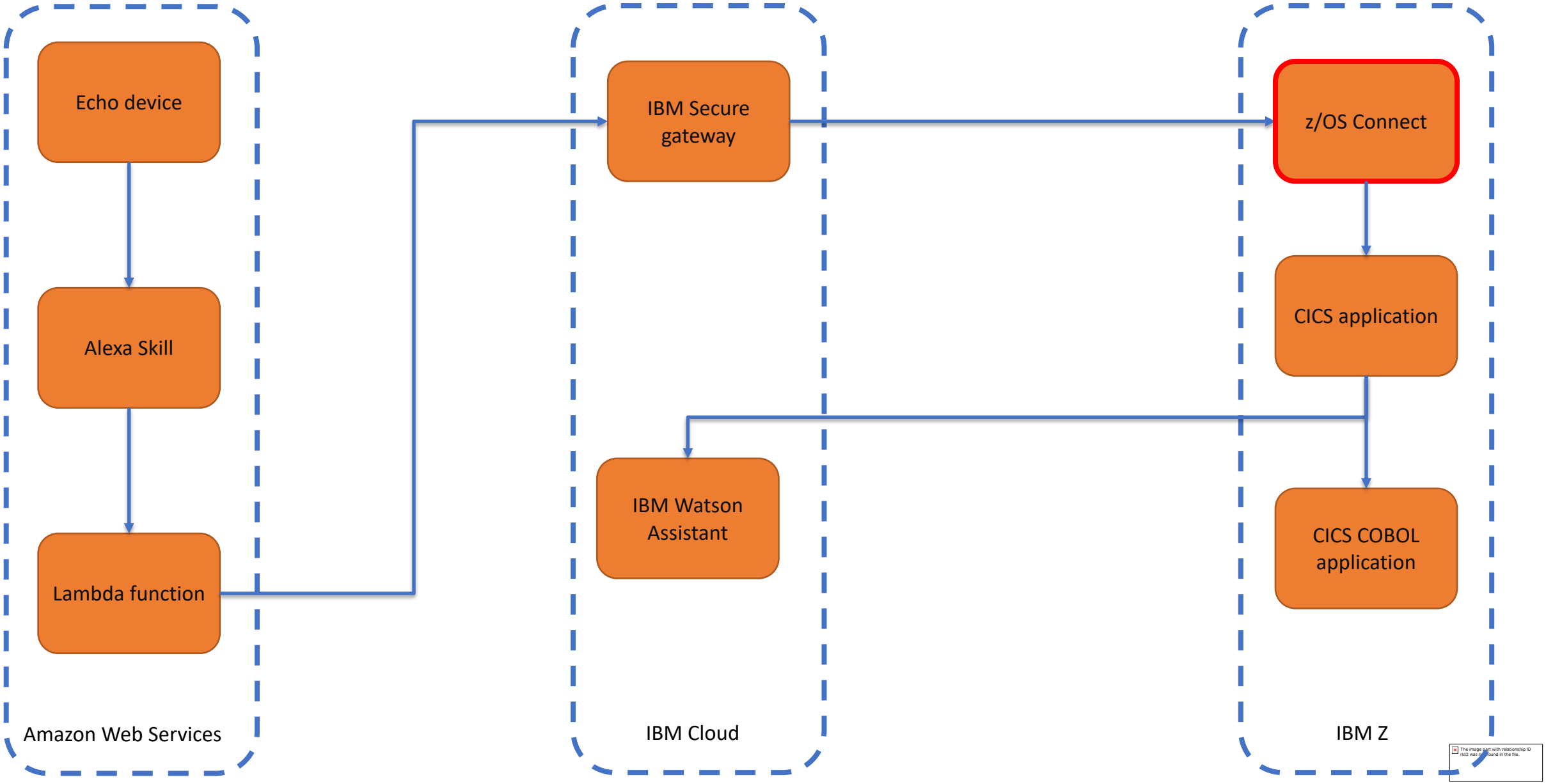
IBM z/OS Connect Enterprise Edition

Truly RESTful APIs to and from your mainframe.

Technical Overview

”Ask CICS bank how much is in my account, it’s Sophie”


Echo device
Alexa Skill
Lambda function
Amazon Web Services
IBM Secure gateway
IBM Watson Assistant
IBM Cloud
z/OS Connect
CICS application
CICS COBOL application
IBM Z

/the_api_economy

Why your business should care?

The History of APIs

1960 - 1980

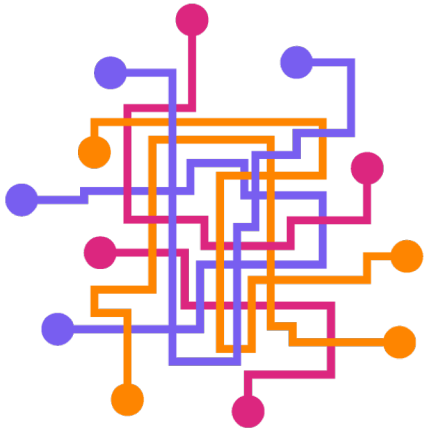
Application specific interfaces.

Source: Deloitte University Press, *API economy: From systems to business services*

The History of APIs

1960 - 1980

Application specific interfaces.



1980 - 1990

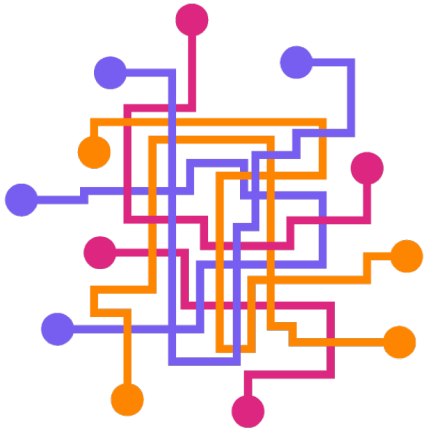
Generic interfaces called by many applications.

Source: Deloitte University Press, *API economy: From systems to business services*

The History of APIs

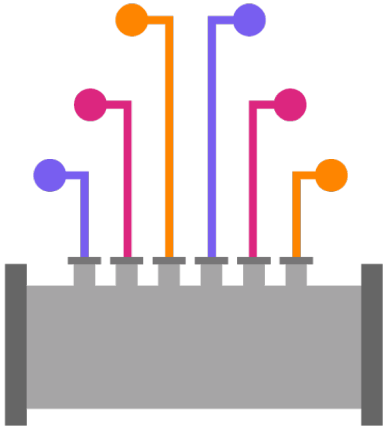
1960 - 1980

Application specific interfaces.



1980 - 1990

Generic interfaces called by many applications.



1990 - 2000

Focus on making it easier to provide and manage interfaces.

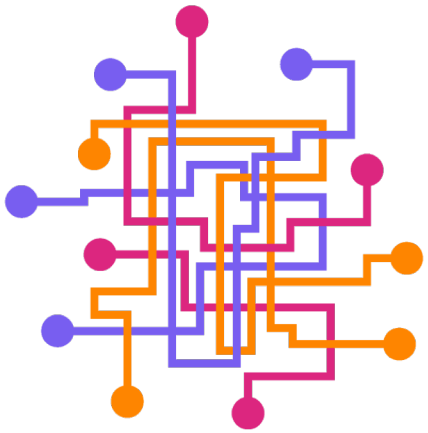
Source: Deloitte University Press, *API economy: From systems to business services*

The History of APIs

z/OS Connect EE

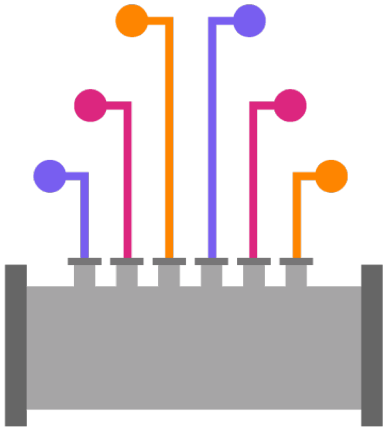
1960 - 1980

Application specific interfaces.



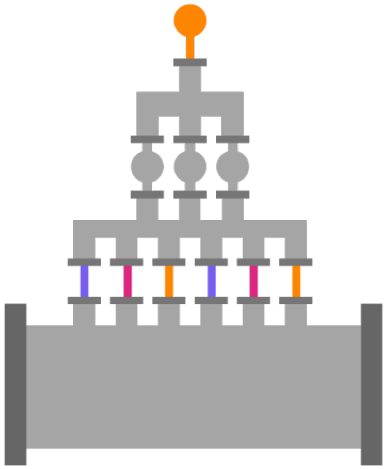
1980 - 1990

Generic interfaces called by many applications.



1990 - 2000

Focus on making it easier to provide and manage interfaces.



2000 - today

Focus on making it easier to discover, consume and combine interfaces.

Source: Deloitte University Press, *API economy: From systems to business services*

Types of API

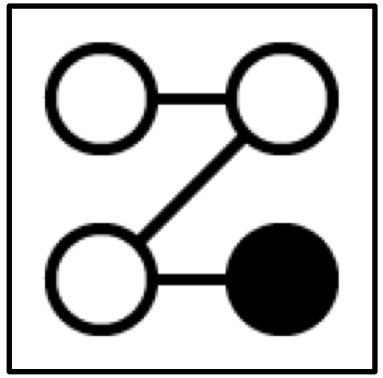
Public **Partner** **Private**

API business models

Free

Paid

Share



Indirect

Developers are your new customers

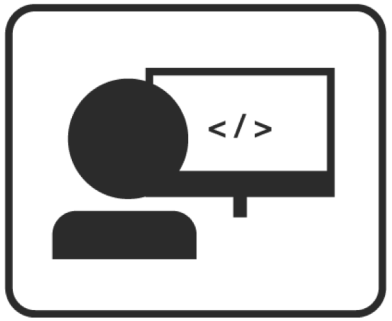
APIs are your new products

Valuable, Timesaving functions

JSON

Familiar Interfaces (intuitive)

RESTful



Developers

Composable

Quick to learn

Well documented

Discoverable

/what_is_REST?

What makes an API “RESTful”?

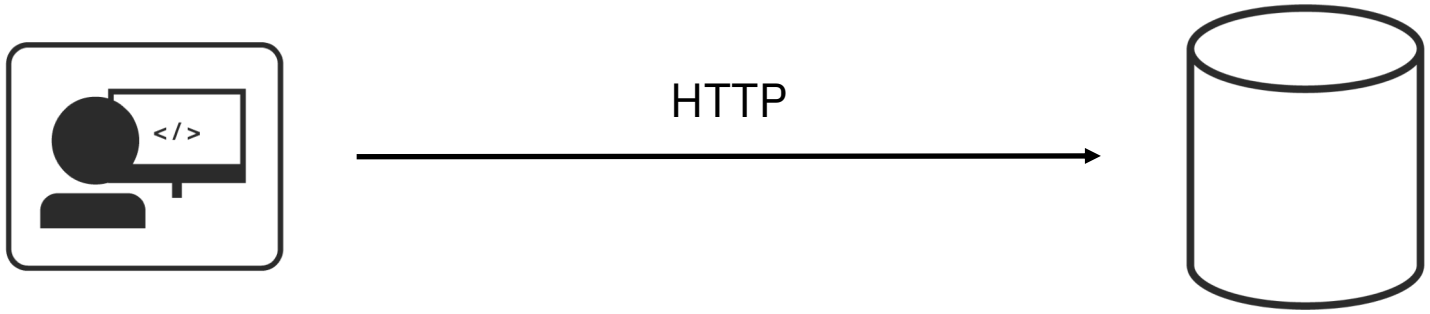
REST is an Architectural Style

REST stands for **Re**presentational **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**).



Roast API Recipe

(How not to do REST...)

1. Take a SOAP/XML web service name, add a “/” before it.
2. Choose randomly an HTTP method between GET, PUT, POST, DELETE.
3. Transform input/output data from XML to JSON.
4. If the method is GET or DELETE, put all parameters in query variables.
5. And be sure to always return HTTP status 200.

Source: apihandyman.io

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

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” }
```

Some red flags…

(How to know if you are doing it wrong)

1. Unique URIs for different operations on the same object

```
http://www.acme.com/customers/GetCustomerDetails/12345
http://www.acme.com/customers/UpdateCustomerAddress/12345?address=
```

2. Different representations of the same objects

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



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

3. Operational data in the request body

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



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

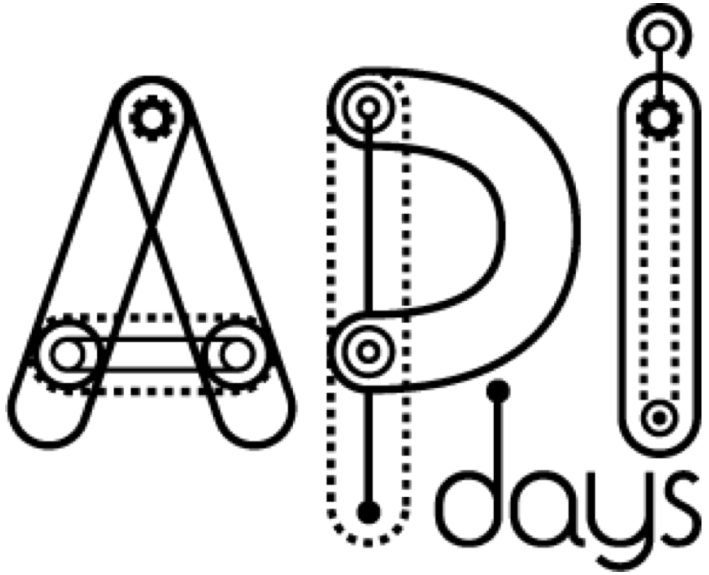
Why is REST popular?

Ubiquitous Foundation	It's based on HTTP, which operates on TCP/IP, which is a ubiquitous networking topology.
Relatively Lightweight	Compared to other technologies (for example, SOAP/WSDL), the REST/JSON pattern is relatively light protocol and data model, which maps well to resource-limited devices.
Relatively Easy Development	Since the REST interface is so simple, developing the client involves very few things: an understanding of the URI requirements (path, parameters) and any JSON data schema.
Increasingly Common	REST/JSON is becoming more and more a de facto "standard" for exposing APIs and Microservices. As more adopt the integration pattern, the more others become interested.
Stateless	REST is by definition a stateless protocol, which implies greater simplicity in topology design. There's no need to maintain, replicate or route based on state.

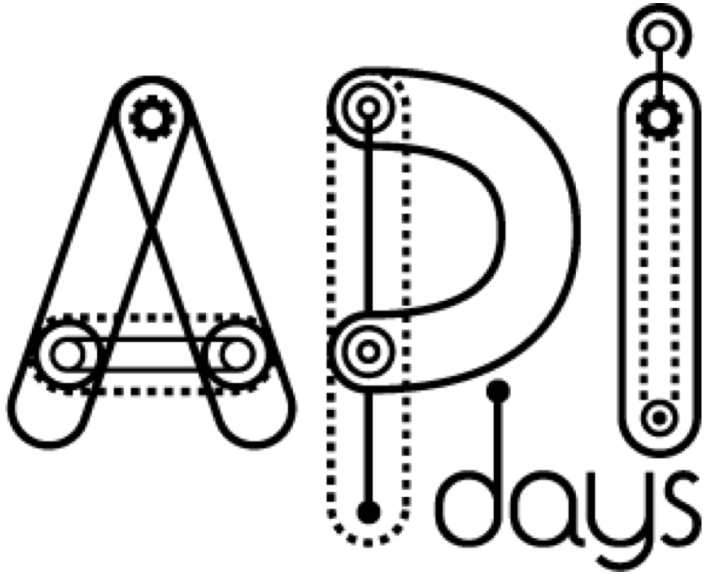
/hosting_apis

How IBM Z can give you the edge

What makes a good API host?



What makes a good API host?



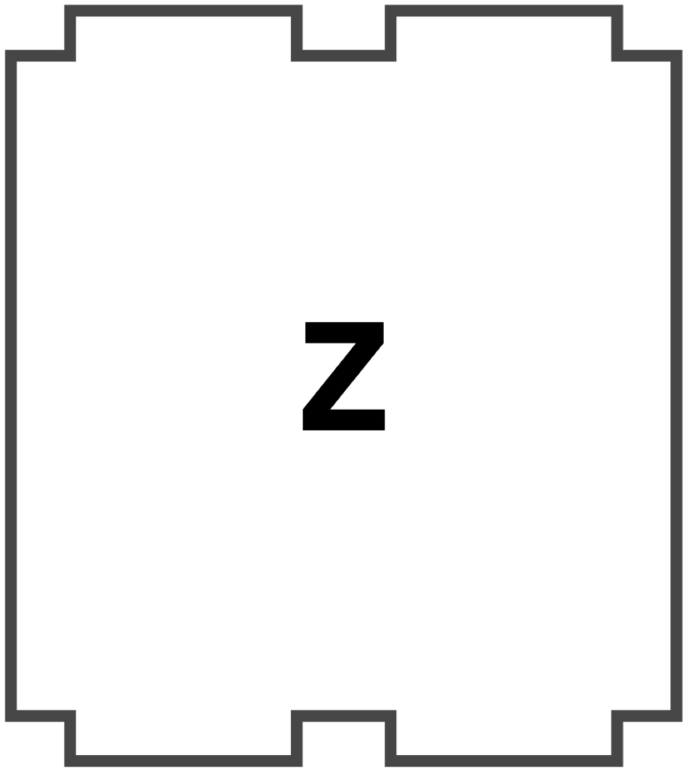
“Fast response time”.

“Extremely reliable”.

“Handles high workloads”.

“Highly available"

What makes a good API host?



"Fast response time".

"Extremely reliable".

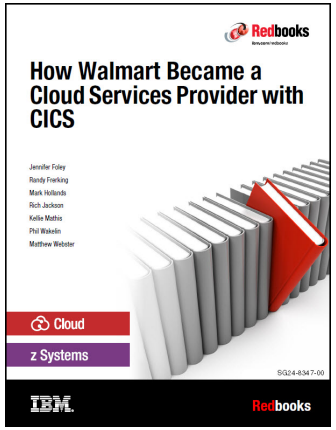
"Handles high workloads".

"Highly available".

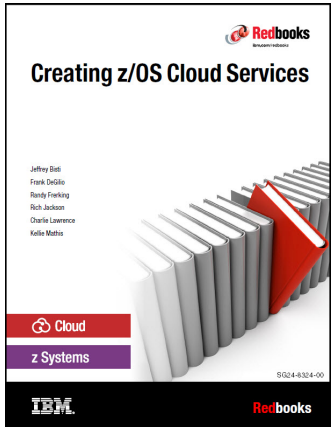
Z has all of these attributes, and it's where your core assets are.

Example: Walmart

The strengths and characteristics of z/OS led Walmart to use this platform, as they transitioned from traditional IT deployment to a cloud model.



How Walmart became a Cloud Services Providers with CICS

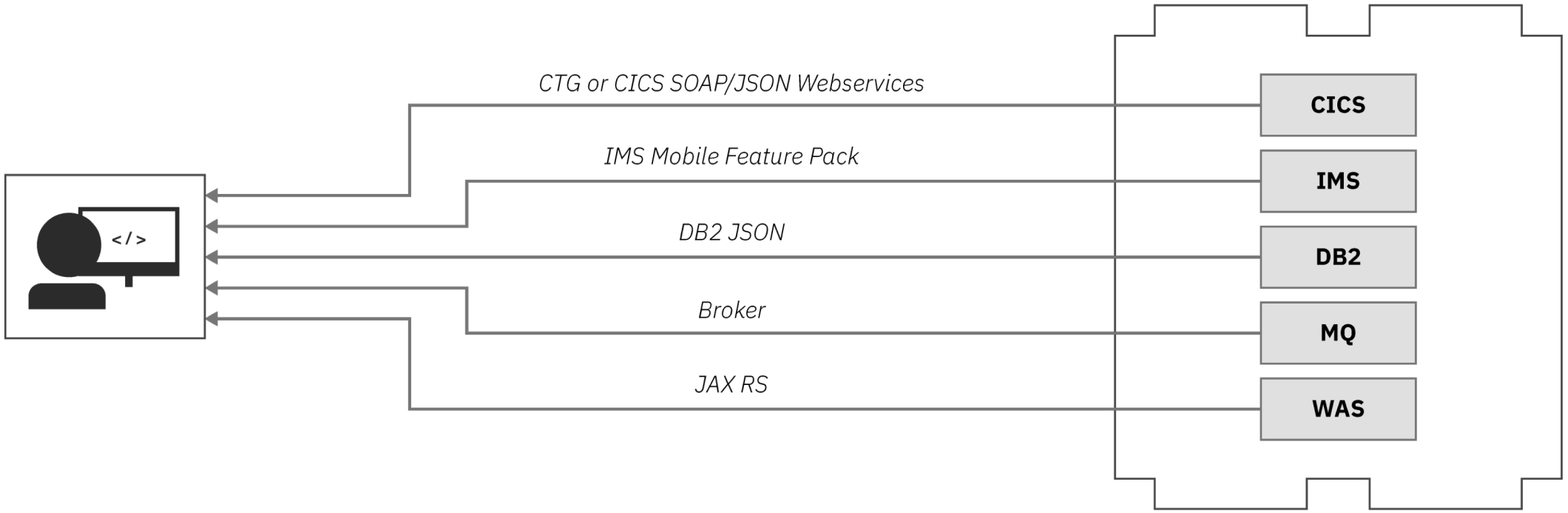


Creating z/OS Cloud Services

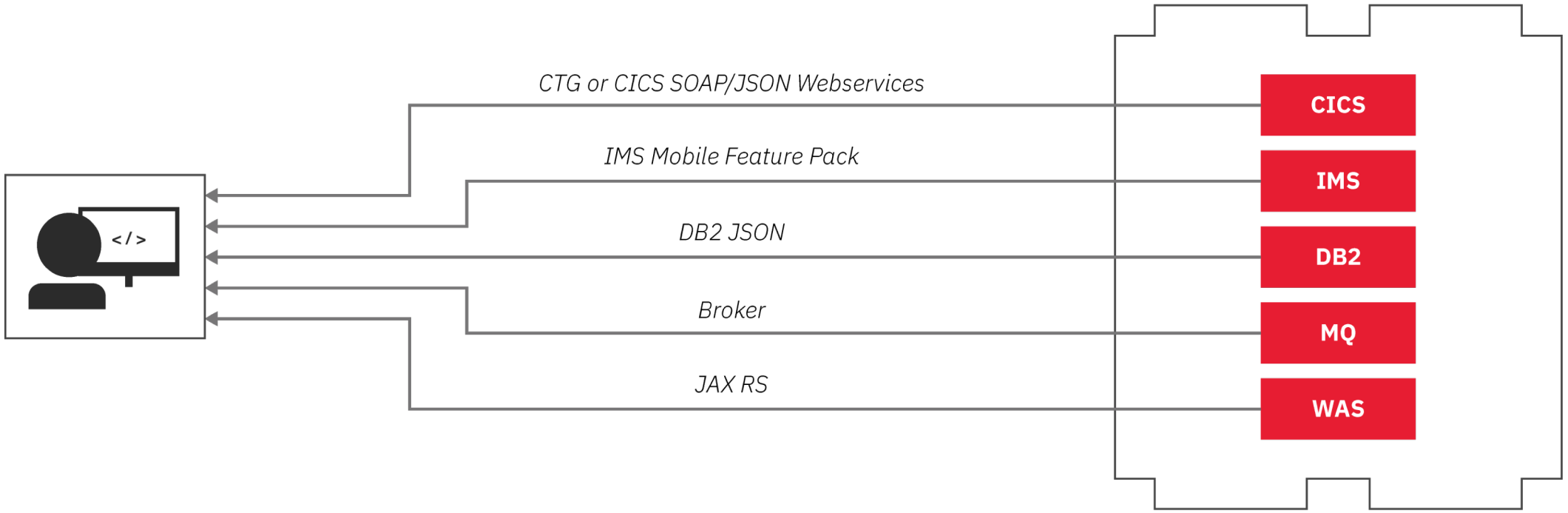
How do we expose z data and services as RESTful APIs?

z/OS Connect EE

Can we do this today?

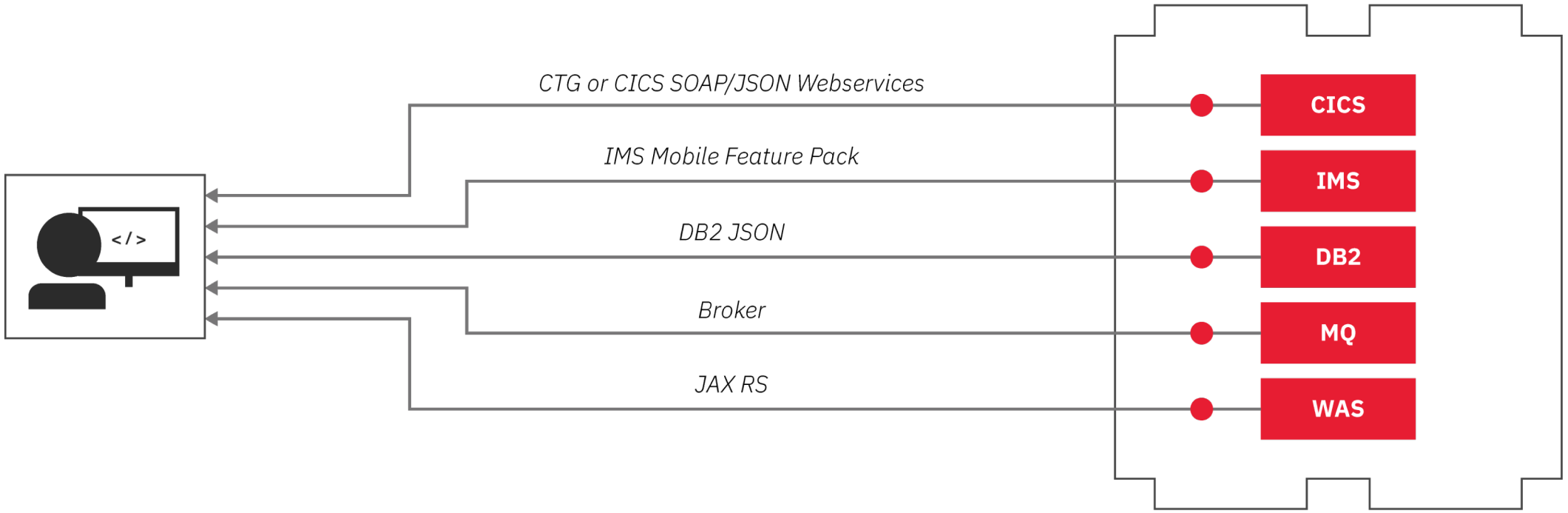


Can we do this today?



Completely different configuration and management.

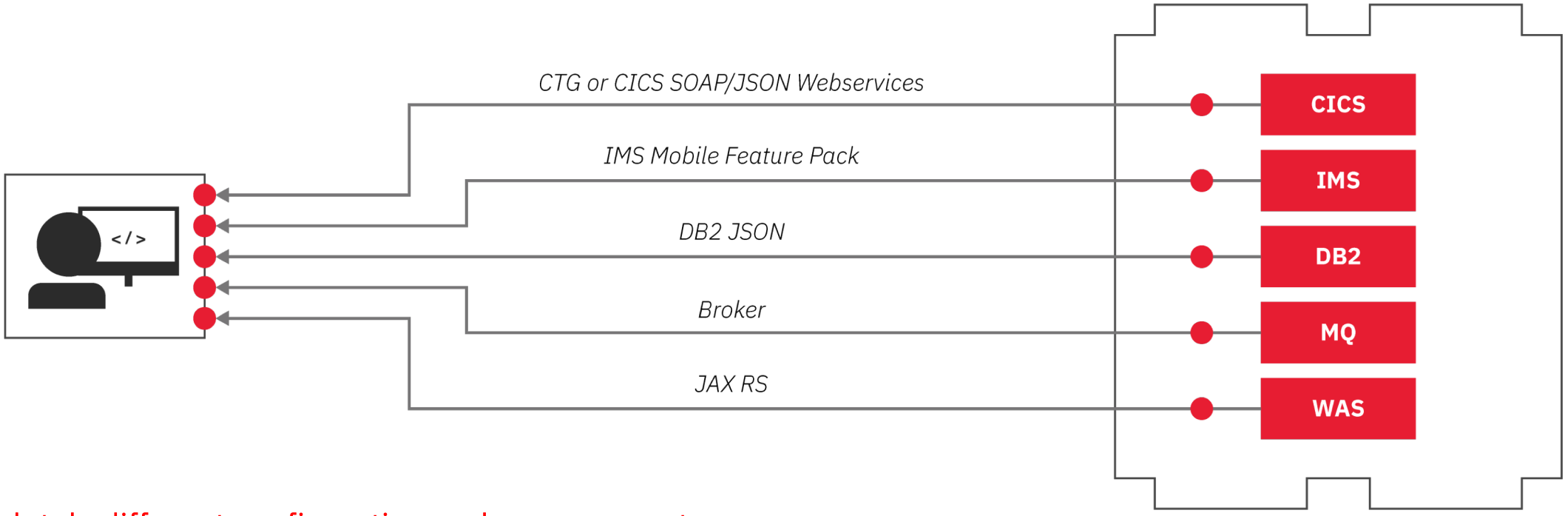
Can we do this today?



Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

Can we do this today?



Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!

Roast API Recipe

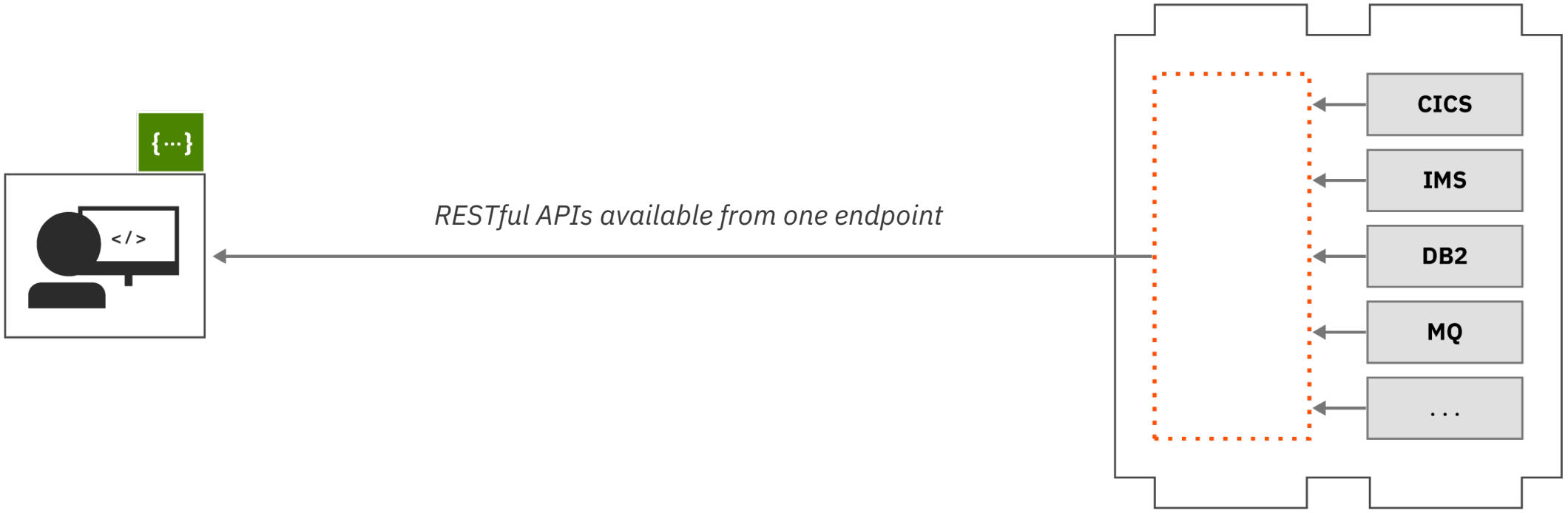
(How not to do REST...)

1. Take a SOAP/XML web service name, add a “/” before it.
2. Choose randomly an HTTP method between GET, PUT, POST, DELETE.
3. Transform input/output data from XML to JSON.
4. If the method is GET or DELETE, put all parameters in query variables.
5. And be sure to always return HTTP status 200.

Source: apihandyman.io

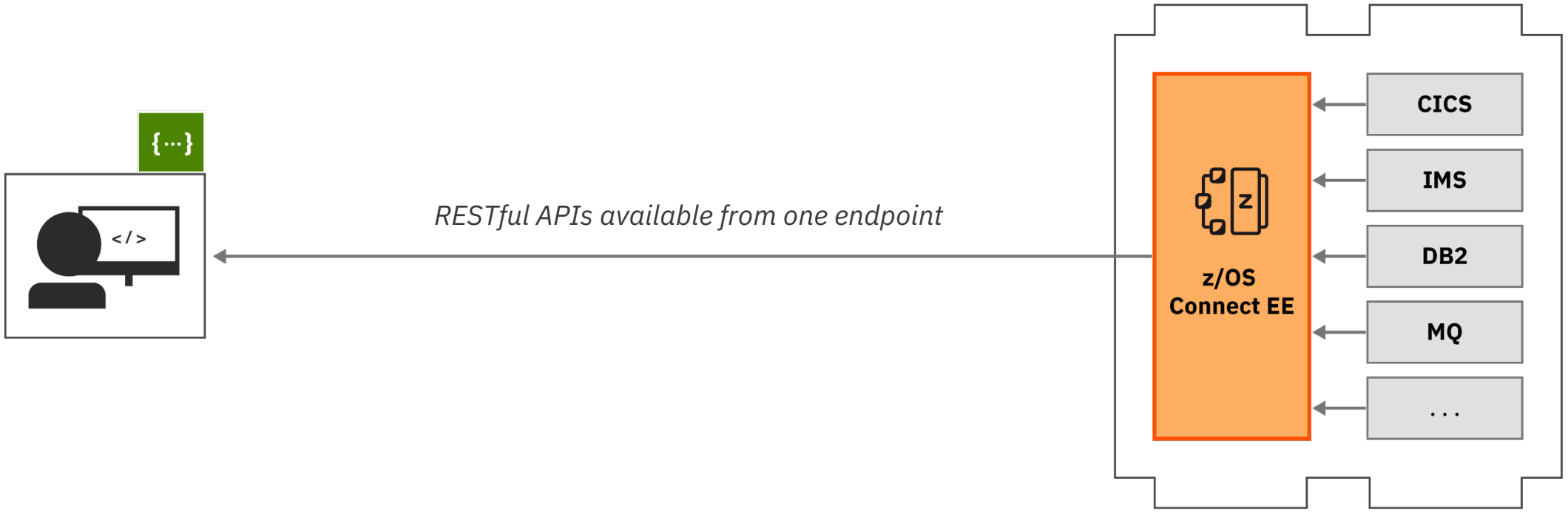
You need a single entry point!

z/OS Connect EE



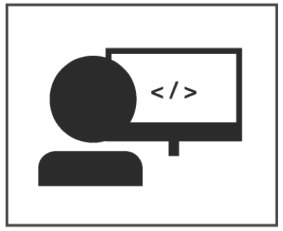
With sophisticated mapping of truly RESTful APIs to existing mainframe and services data.

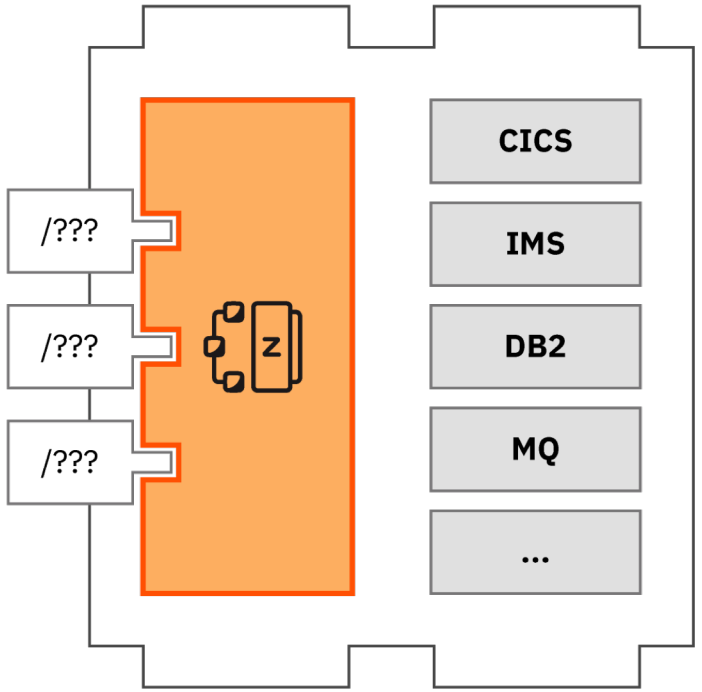
You need a single entry point!



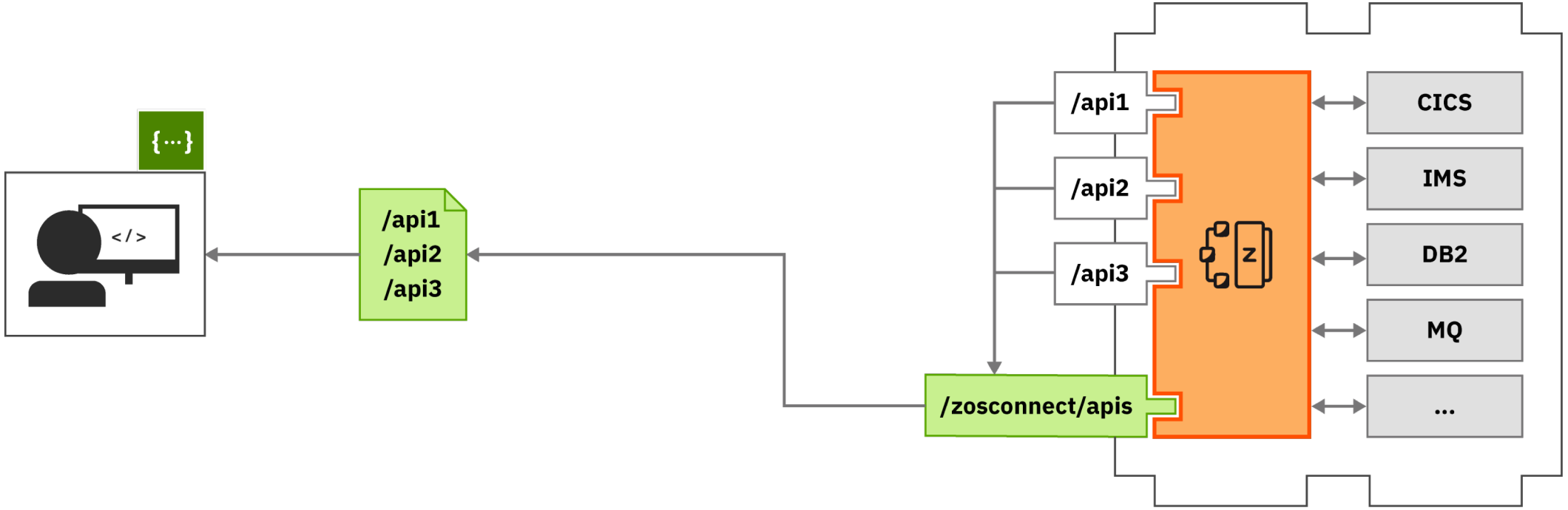
With sophisticated mapping of truly RESTful APIs to existing mainframe and services data.

How do developers know what APIs are available?



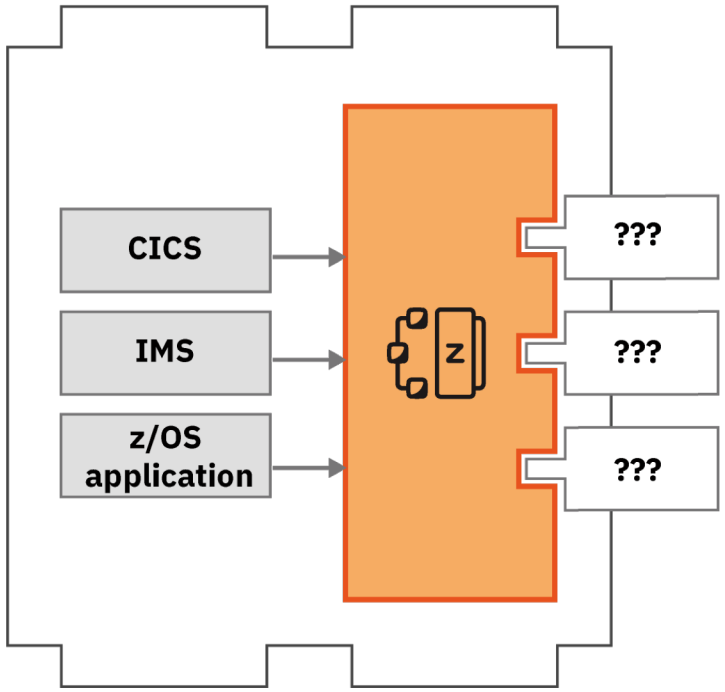


We need some Swagger!

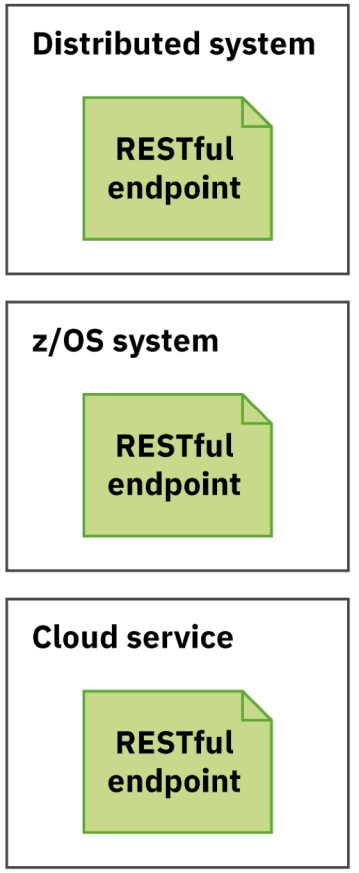


APIs are discoverable through **automatically generated** Swagger documents, served straight from z/OS Connect EE.

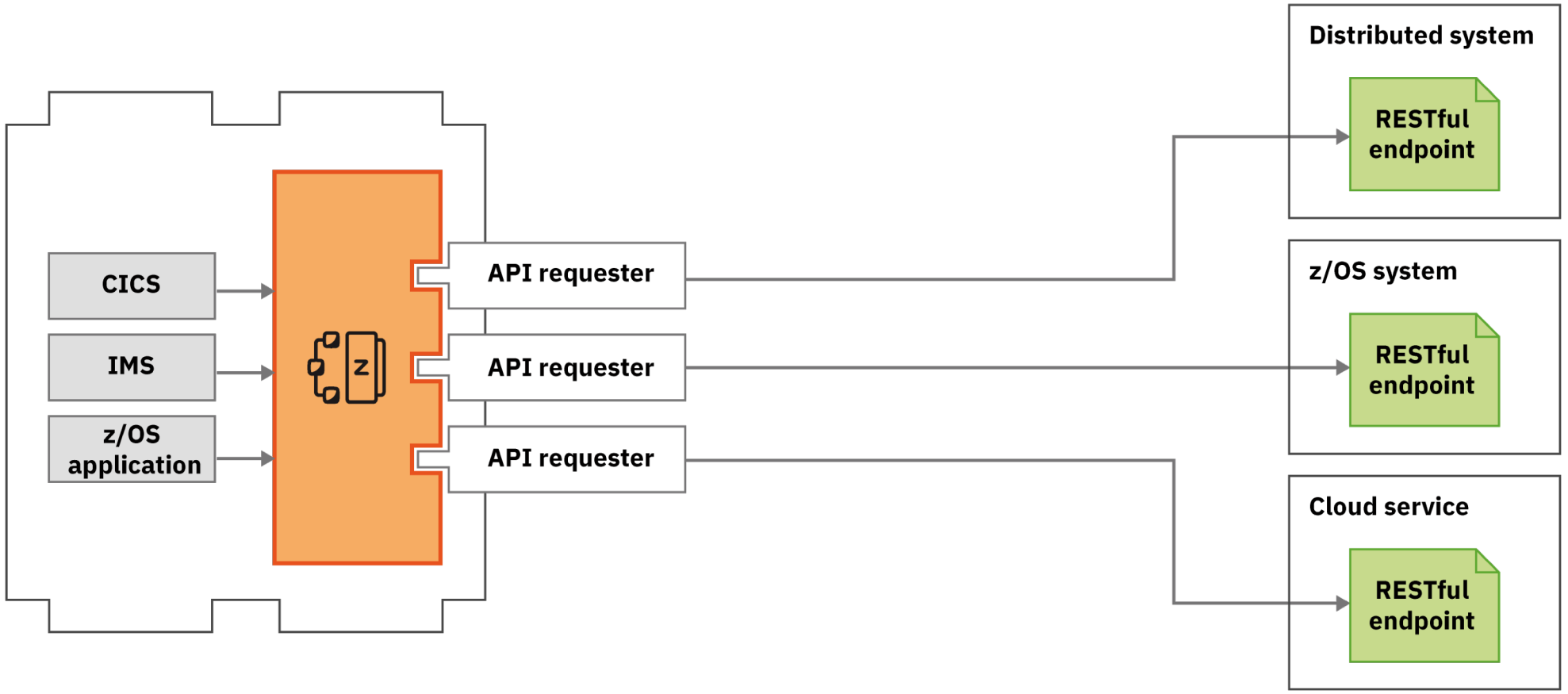
What if I want to call APIs from z/OS assets?



?
?
?



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



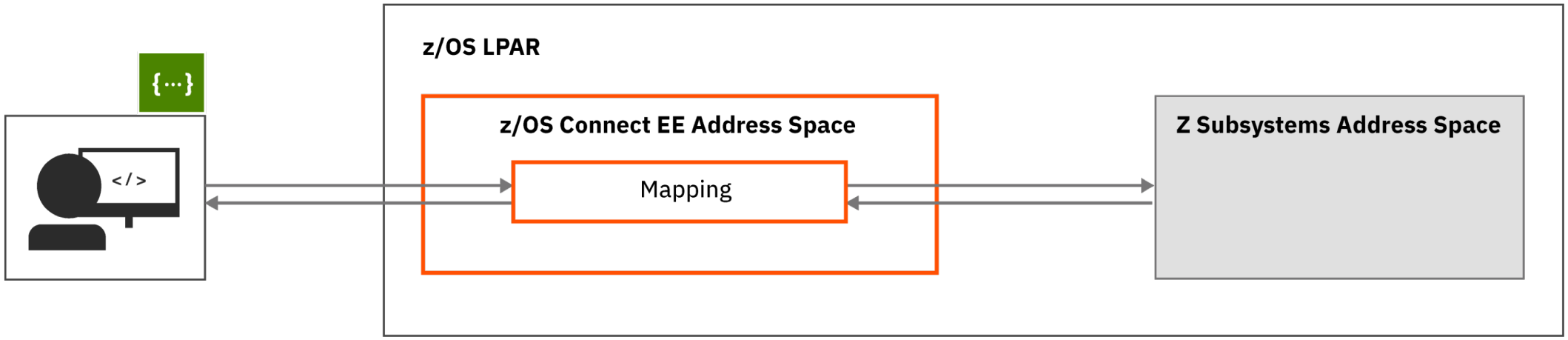
ibm.biz/zosconnect-api-requester-overview

Expose z/OS assets as RESTful APIs without writing any code.

z/OS Connect EE

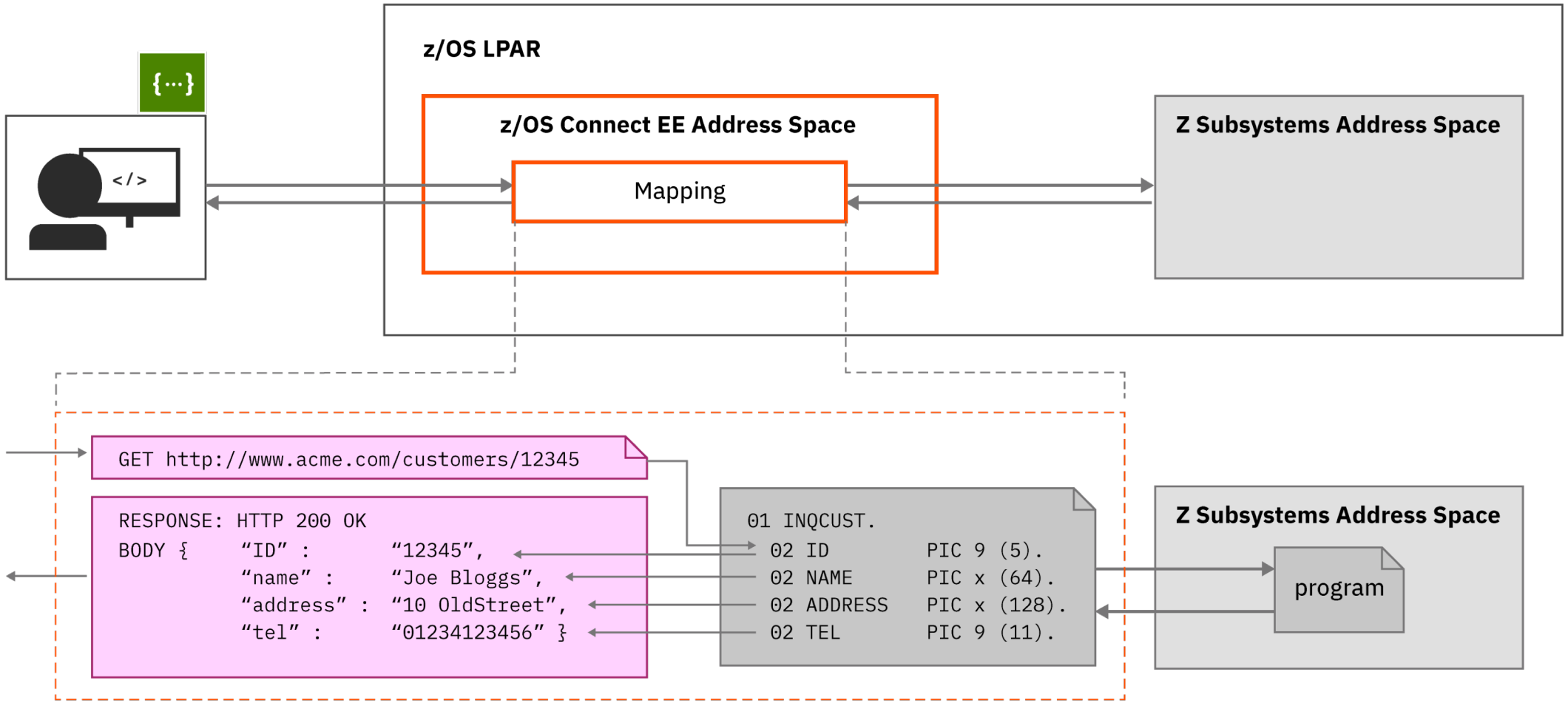
Data mapping

Overview



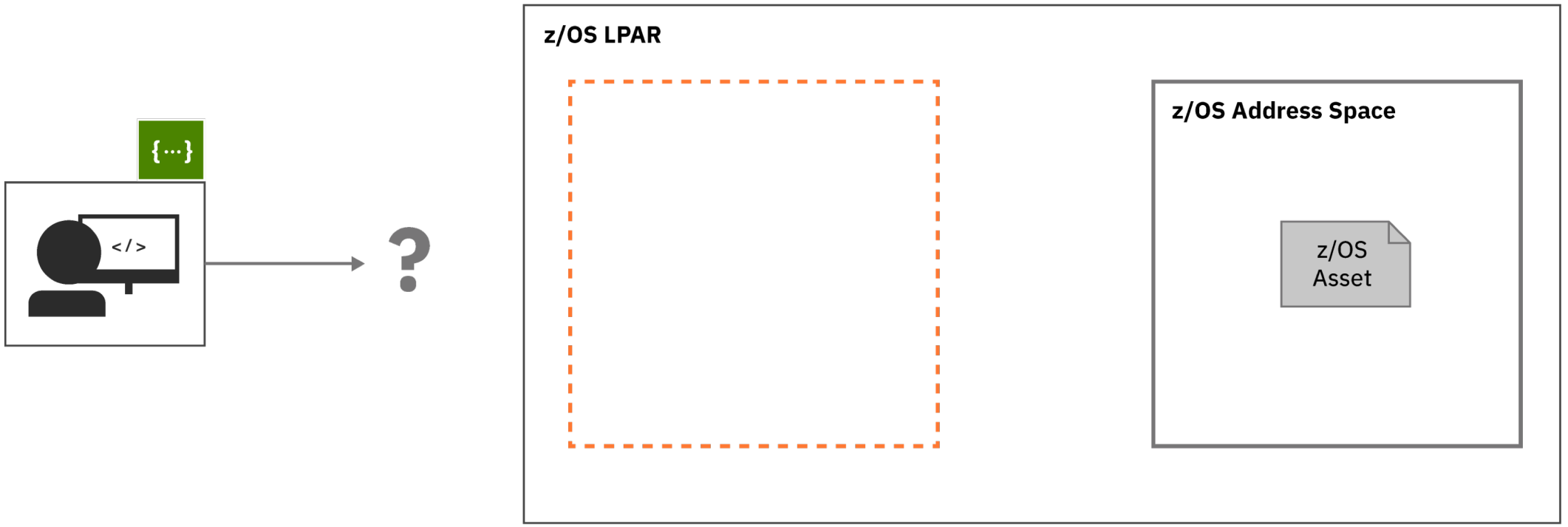
Data mapping

A closer look



Six Steps to expose a z/OS Asset

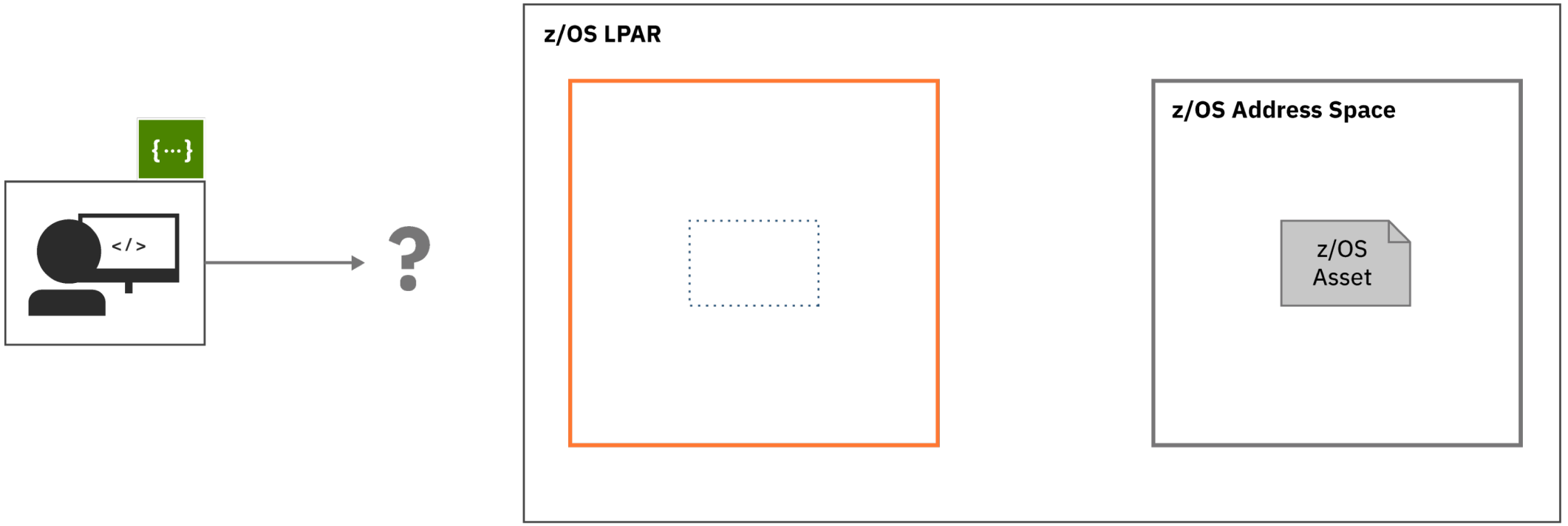
Starting point



You've chosen a z/OS asset you want to expose as a RESTful endpoint.

Six Steps to expose a z/OS Asset

1. Install z/OS Connect EE



Install, set up, and start your new **z/OS Connect EE Server**.

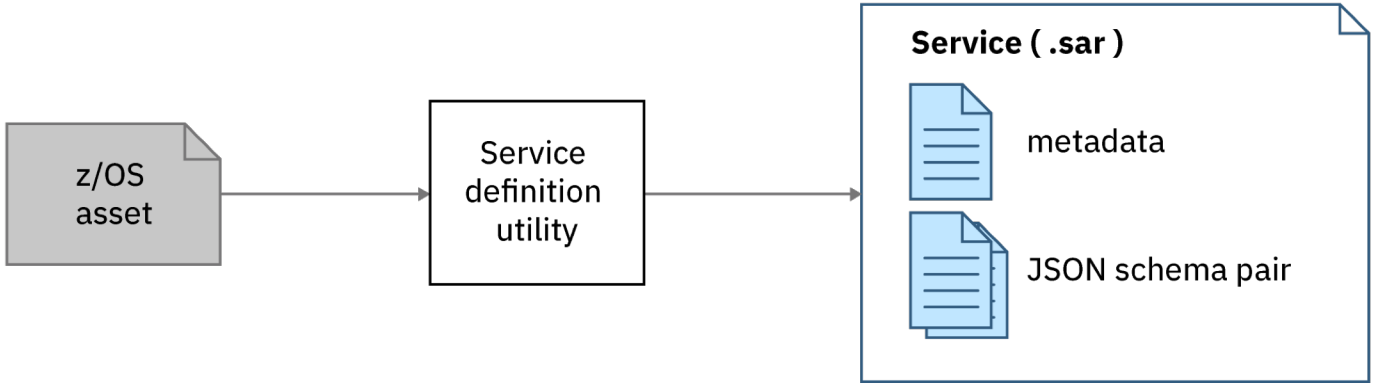
ibm.biz/zosconnect-installation

ibm.biz/zosconnect-start-server

Six Steps to expose a z/OS application

2. Create your 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`).

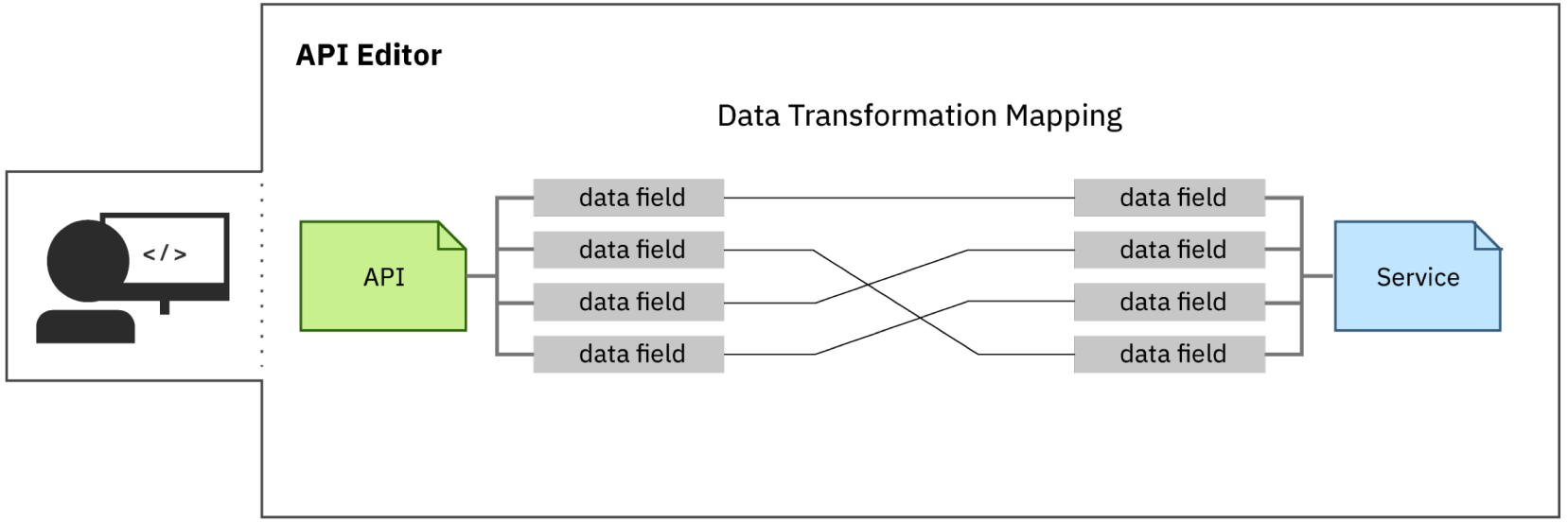


Use a system-appropriate utility to generate a `.sar` file for the z/OS application.

ibm.biz/zosconnect-sar-creation

Six Steps to expose a z/OS application

3. Create your API



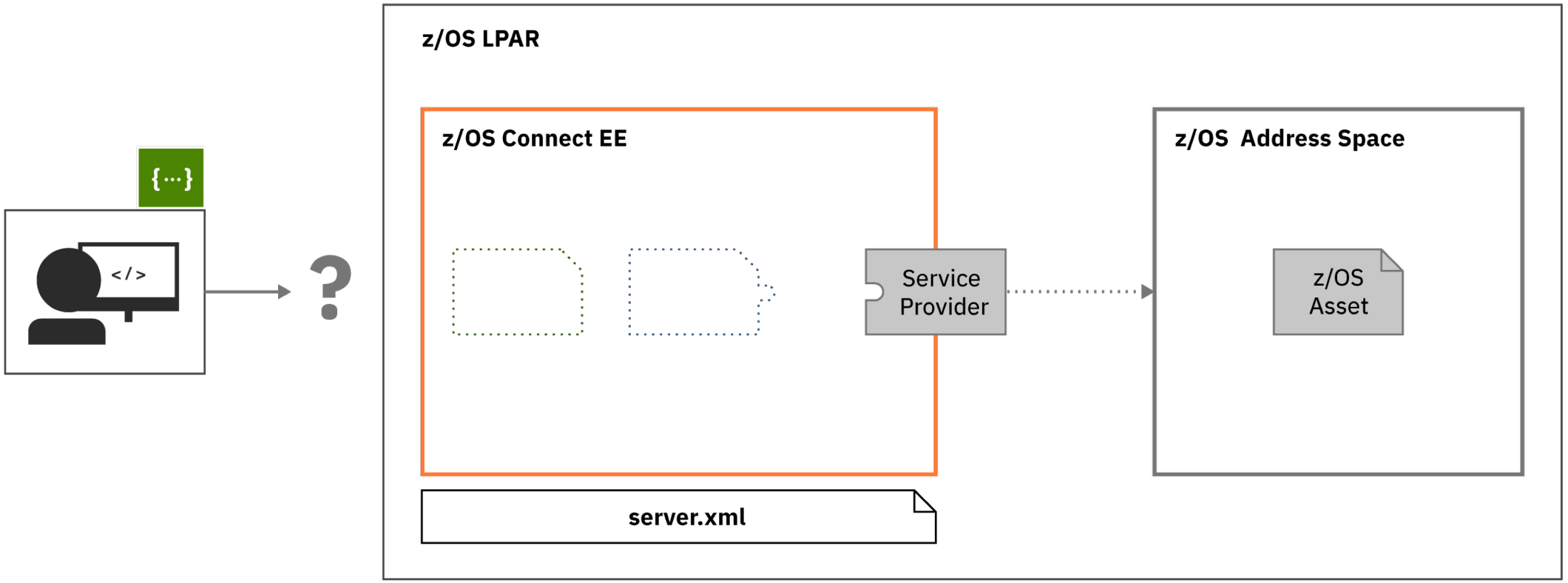
Import your `.sar` file into the **API toolkit**, and start designing your API.

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

Six Steps to expose a z/OS application

4. Configure your service provider

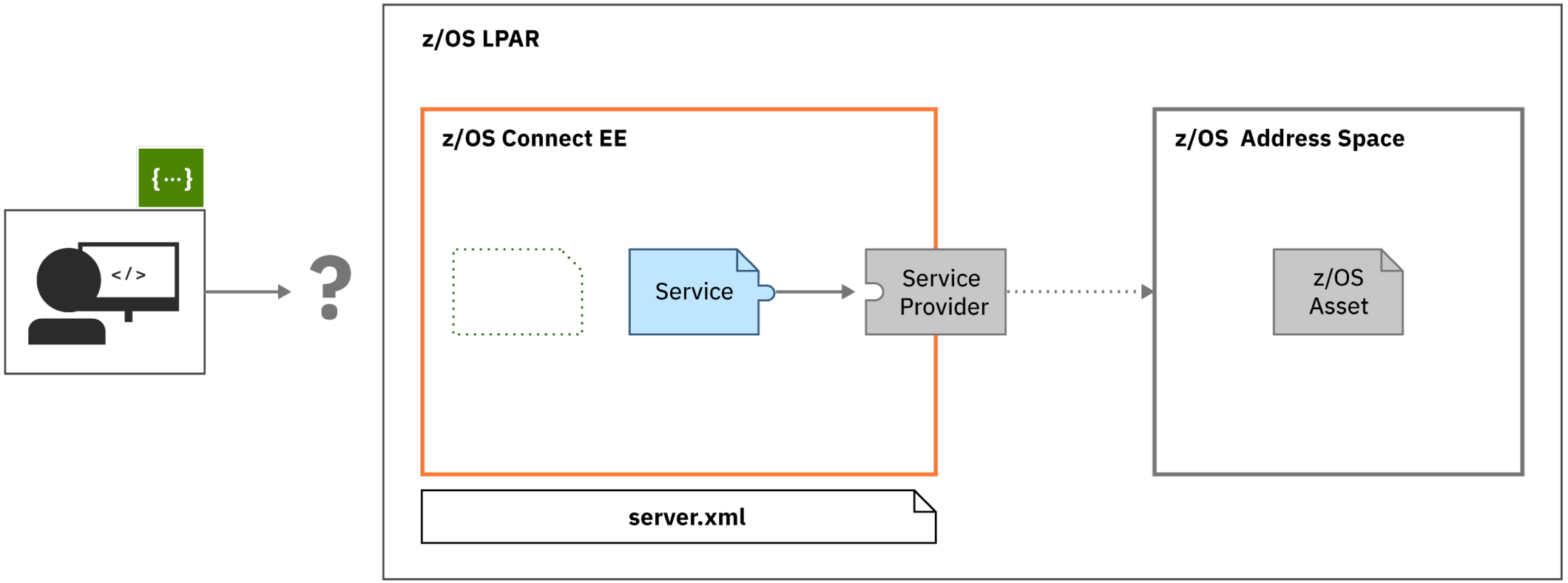


Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.

ibm.biz/zosconnect-configuring

Six Steps to expose a z/OS application

5. Deploy your service

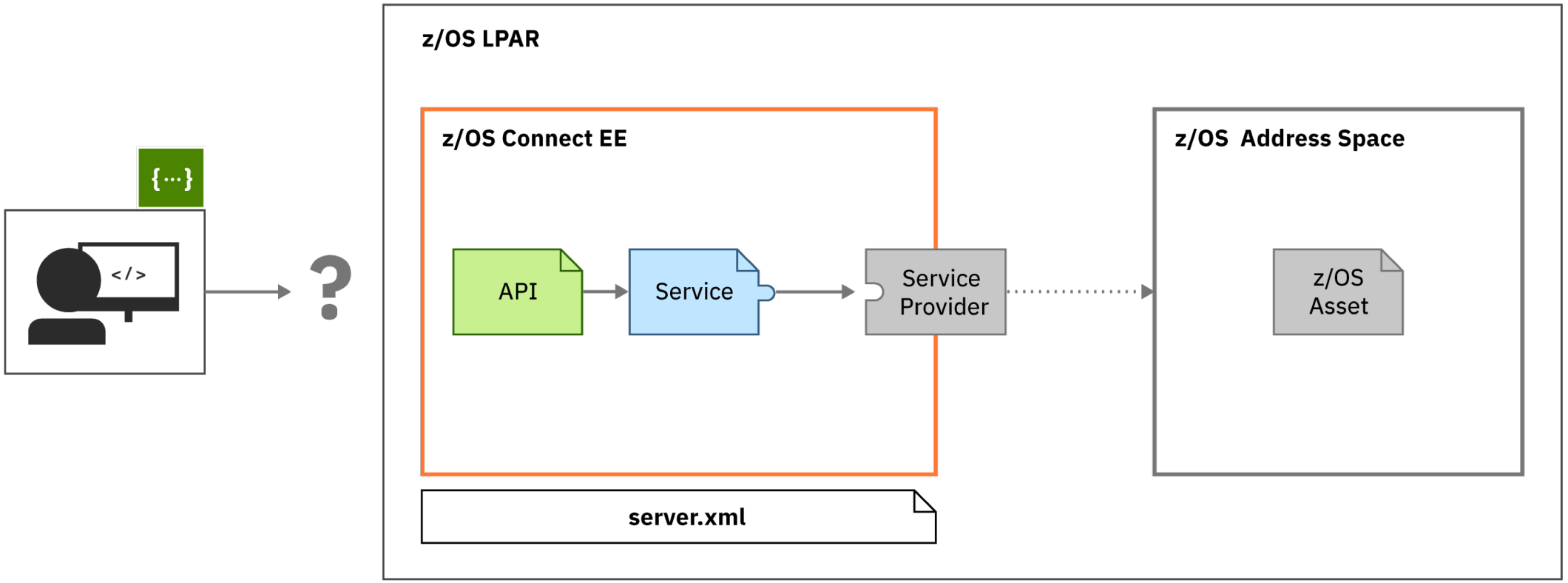


Deploy the `.sar` file generated by the *service definition utility* by copying the `.sar` file to the *services* directory. (This step uses the `.sar` file generated in **Step 2**.)

ibm.biz/zosconnect-define-services

Six Steps to expose a z/OS application

6. Deploy your API

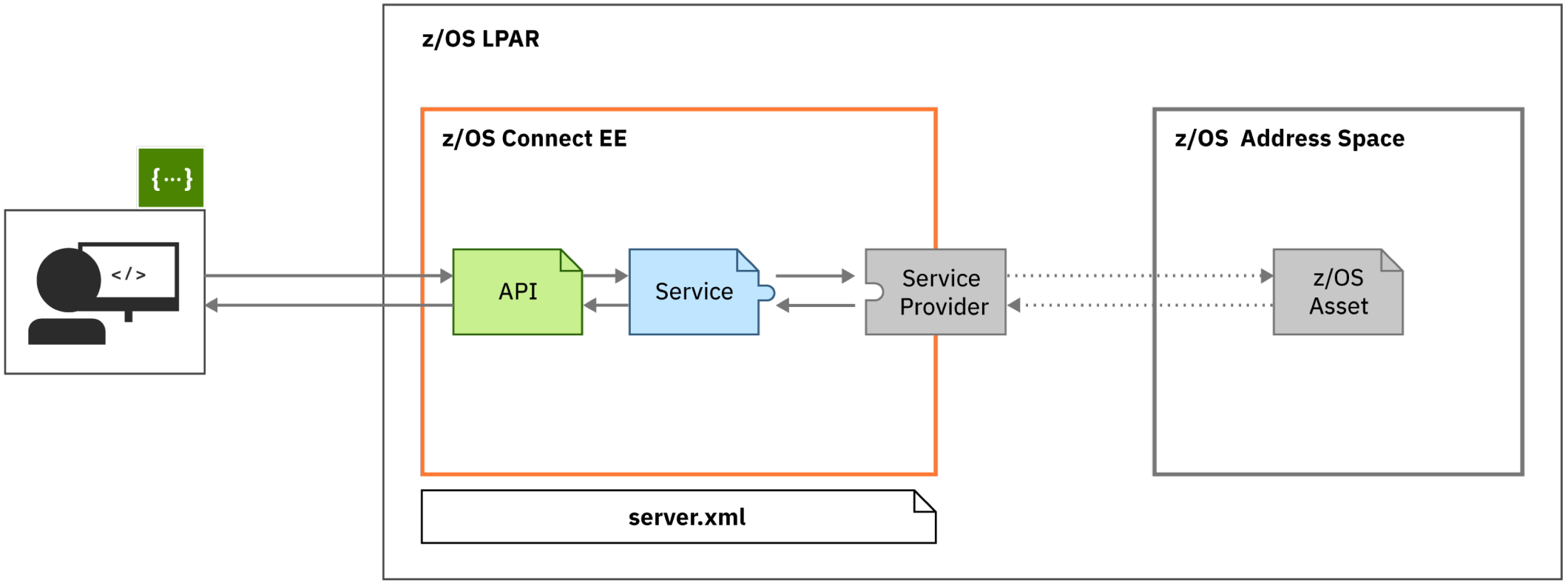


Deploy your API using the right-click deploy in **the API toolkit**,
or by copying the `.aar` file to the *apis* directory.

ibm.biz/zosconnect-deploy-api

Six Steps to expose a z/OS application

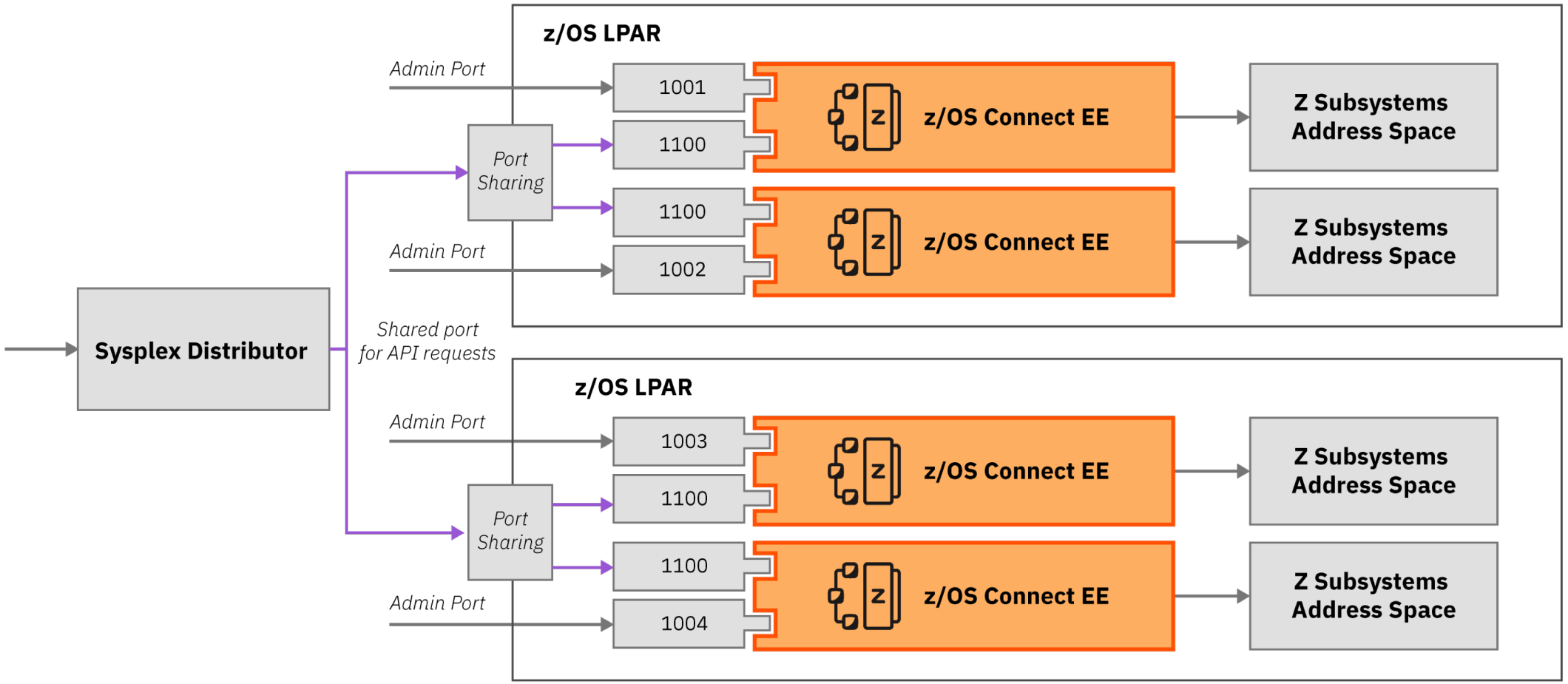
Done



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

High Availability

Topology

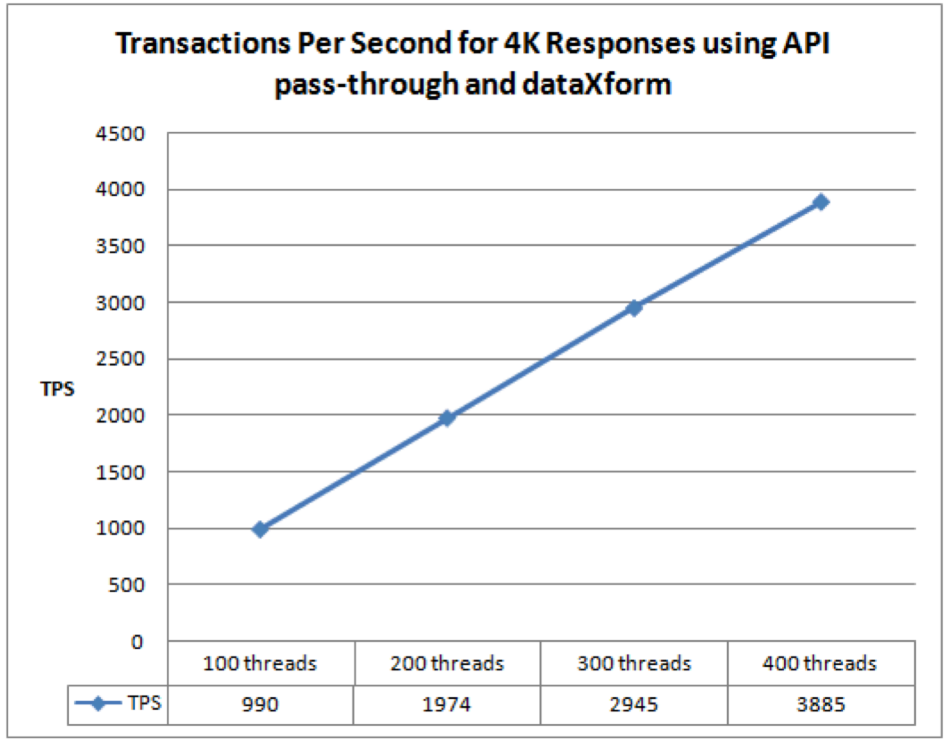


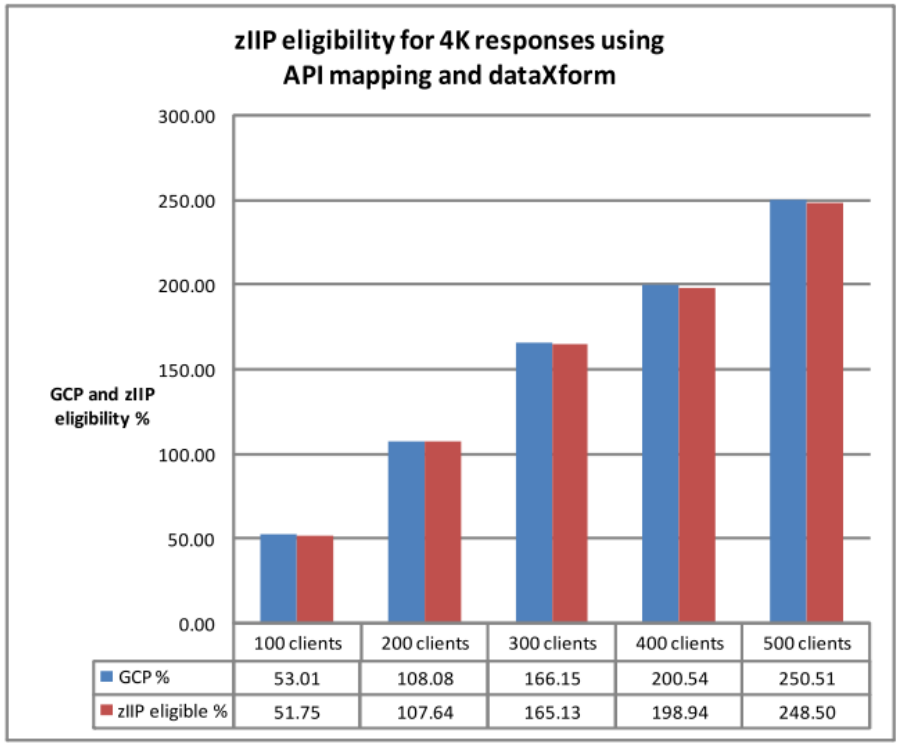
ibm.biz/zosconnect-ha-concepts

ibm.biz/zosconnect-scenarios

Performance

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**.

ibm.biz/zosconnect-performance-report

IBM z/OS Connect Enterprise Edition

Continuous Delivery Updates

[3.0.11] July 2018 **Per API Policy**
- Policies can now be configured for individual APIs

[3.0.11] July 2018 **Upgraded to Liberty 18.0.0.2**

[3.0.10] June 2018 **Persistent connections for API Requester from IMS and z/OS Applications**
- The communication stub for IMS and other non-CICS z/OS applications has been enhanced to automatically enable persistent connections to the z/OS Connect EE server

[3.0.10] June 2018 **Policy for CICS Services**
- Use policy to alter cicsCcsid, cicsConnectionRef and cicsTransId dynamically based on HTTP headers in the API request

[3.0.9] May 2018 **Mapping array type parameters to service fields**
- You can now add array type HTTP headers, query parameters, and path parameters to your APIs and map them to various service fields.

ⓘ See the change history for details ibm.biz/zosconnect-kc-changehistory

IBM z/OS Connect Enterprise Edition

Continuous Delivery Updates Cont.

[3.0.8] Apr 2018 **DevOps for services, APIs, and API requesters**

- You can now automate the development, test, and deployment of services, APIs, and API requesters for continous integration and delivery.

[3.0.8] Apr 2018 **Support for allOf keyword for API Requesters**

- For users generating artifacts for an API requester, the z/OS Connect EE build toolkit now can process the allOf keyword in a Swagger file.

[3.0.8] Apr 2018 **API toolkit updated for CICS Context Containers**

[3.0.8] Apr 2018 **Upgraded to Liberty 18.0.0.1**

[3.0.7] Mar 2018 **CICS Context Containers**

- CICS service provider services can now be configured to send context containers to CICS programs, providing the program with information about the context in which the service was invoked.

[3.0.6] Feb 2018 **Upgraded to Liberty 17.0.0.4**

[3.0.6] Feb 2018 **Dynamic refresh of Policies (update policies without server restart)**

ⓘ See the change history for details ibm.biz/zosconnect-kc-changehistory

IBM z/OS Connect Enterprise Edition

Continuous Delivery Updates Cont.

[3.0.6] Feb 2018 **Support for API key security when using API Requester**

[3.0.5] Jan 2018 **Enhancements to API requester to support multidimensional arrays and JSON additional properties in the OpenAPI document**

[3.0.4] Dec 2017 **Enhancements to the RESTful service administration interface**

- Version 1.1.0 of the interface adds support for POST, PUT, and DELETE methods to support service deployment, service updates, service status updates, and service removal.

[3.0.4] Dec 2017 **Support for z/OS applications to call an API that is protected by OAuth 2.0 on a request endpoint**

[3.0.4] Dec 2017 **Enhancements to the API toolkit**

- Version 3.0.3 of the API toolkit supports right-click deployment of service projects, and allows you to view deployed services, examine service properties, update a service, and start/stop a service from the API toolkit.

[3.0.4] Dec 2017 **Support for imsTranCode override in policy-based API processing**

- You can now override the IMS transaction code (imsTranCode) that your service invokes at run time.

[3.0.3] Nov 2017 **Policy-based API processing for APIs that target an IMS application**

See the change history for details ibm.biz/zosconnect-kc-changehistory

IBM z/OS Connect Enterprise Edition

Version 3.0 recap + Continuous Delivery Updates

[3.0.3] Nov 2017 **CICS® IPIC High availability**
[3.0.2] Sept 2017 **Support for distributed identity propagation to CICS**
[3.0.2] Sept 2017 **New PL/I sample that demonstrates handling large IMS™ data structures**
[3.0.1] July 2017 **Call external APIs from z/OS assets (CICS, IMS, and other z/OS applications)**
[3.0.0] June 2017 **API-enable more IBM z/OS applications**

- Support for applications with complex data structures, such as `REDEFINES` and `OCCURS DEPENDING ON` clauses in COBOL, and `REFER` option in PL/I.
- Support for COBOL and PL/I applications in IMS with large data structures that go beyond traditional message segment limits.

New IBM CICS Service Provider

- Support for multiple CICS containers, including any mix of BIT- and CHAR-type container.
- Exploits IP interconnectivity (IPIC) for cross-LPAR connectivity.

z/OS Connect EE API toolkit (Eclipse tooling)

- Create and dynamically deploy service artefacts (`.sar` files) for CICS and IMS.
- Rename, redact, and add descriptions to data fields in services.

Resources

Downloads

- z/OS Connect EE open beta runtime — ibm.biz/zosconnect-open-beta
- z/OS Connect EE workstation tooling — ibm.biz/zosconnect-tooling-download

Explore the docs

- z/OS Connect EE Knowledge Center — ibm.biz/zosconnect-kc
- z/OS Connect EE Developer Center — ibm.biz/zosconnectdc

Where to get help

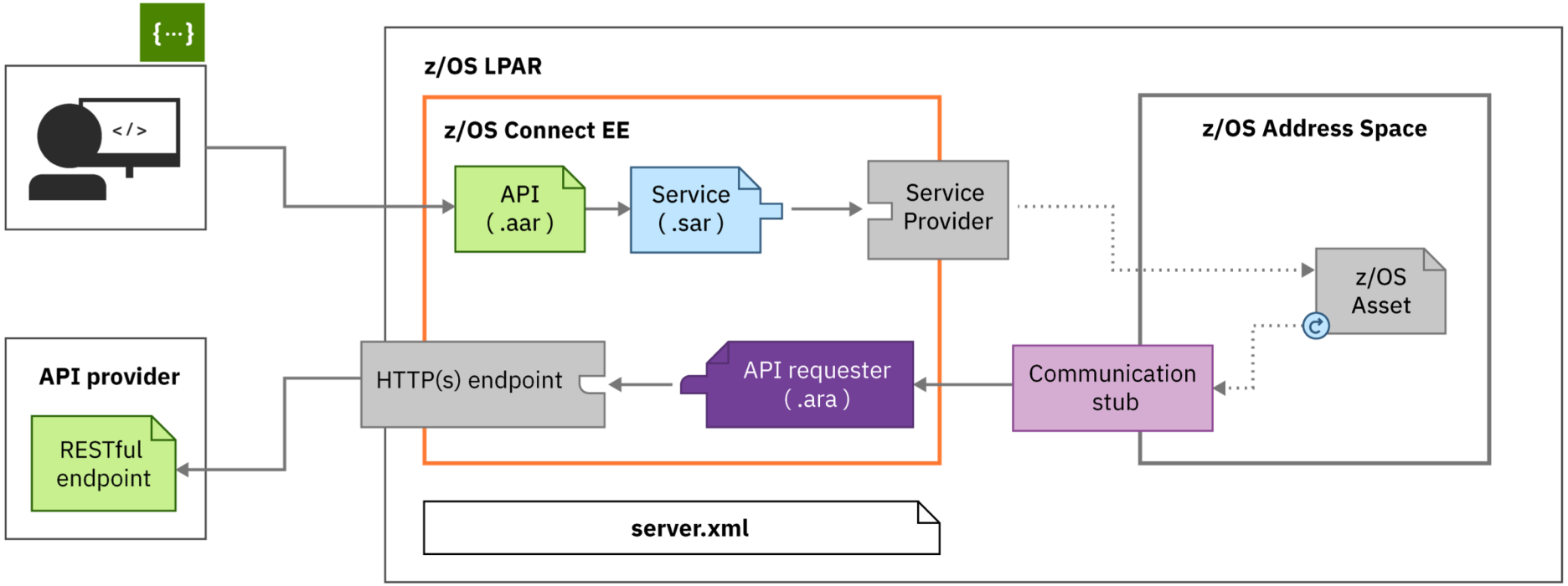
- dW Answers — ibm.biz/zosconnect-dw-answers
- z/OS Connect EE open beta forum — ibm.biz/zcee-beta-forum

IBM z Systems Trial Program

Try the latest z/OS Connect EE capabilities today at zero cost, and with no installation required.

Find out more, and sign up now at ibm.biz/ibmztrial

Make IBM Z the heart of your API strategy with truly RESTful APIs to and from your mainframe



Get started with the open beta today: **ibm.biz/zosconnectdc**

/summary

Developers are your new customers.

Expose z/OS assets as RESTful APIs without writing any code,
and call external APIs from your z/OS assets.

By using **z/OS Connect EE**.