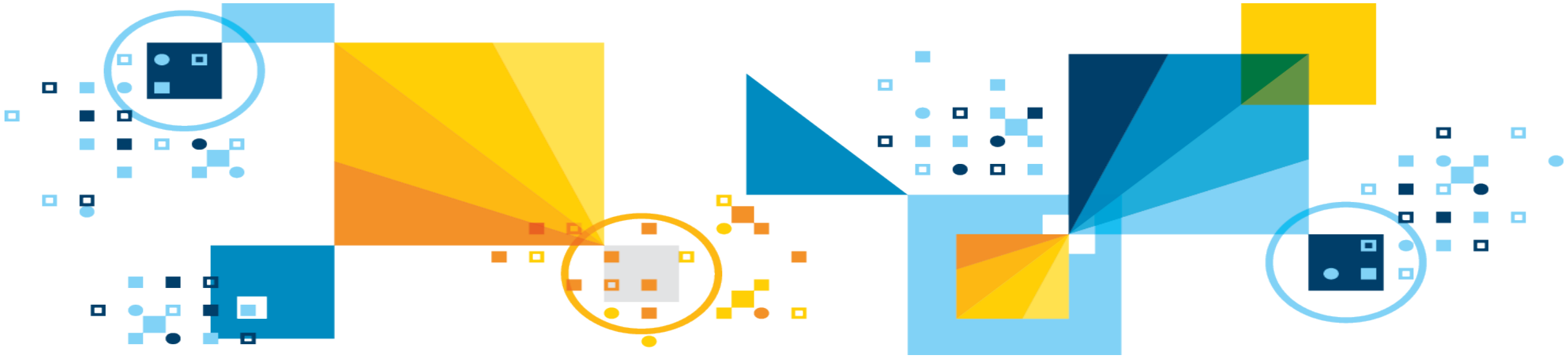
Robert Catterall
IBM Senior Consulting Db2 for z/OS Specialist
rfcatter@us.ibm.com

Data-as-a-Service: the REST Interface to Db2 for z/OS

TRIDEX
June 11, 2019



Agenda

- The distinction between data-as-a-service and database-as-a-service
- Data-as-a-service in a Db2 for z/OS context: the REST interface
- Where z/OS Connect fits in the picture

The distinction between data-as-a-service and database-as-a-service

Data-as-a-service / database-as-a-service – not the same thing

- Key distinction is in the name
  - With database-as-a-service, you want the functionality of a database management system, provided as a service
    - In a Db2 for z/OS context, that usually means easy requisitioning and fast, automatic provisioning of schema services (e.g., CREATE TABLE, ALTER TABLE, etc.) and data services (e.g., load data into tables, back up and restore data, etc.) for application developers in a development environment
    - Capabilities in this area got a major boost with the recent debut of an IBM offering called Db2 DevOps Experience for z/OS (product ID: 5698-DEX)
  - Data-as-a-service is about the programmatic interface to a data server
- Another way to think of data-as-a-service versus database-as-a-service:
  - Data-as-a-service impacts the code (the data server interface) that a developer writes
  - Database-as-a-service impacts the speed with which a developer can create or modify an application

Data-as-a-service: a higher level of data server abstraction

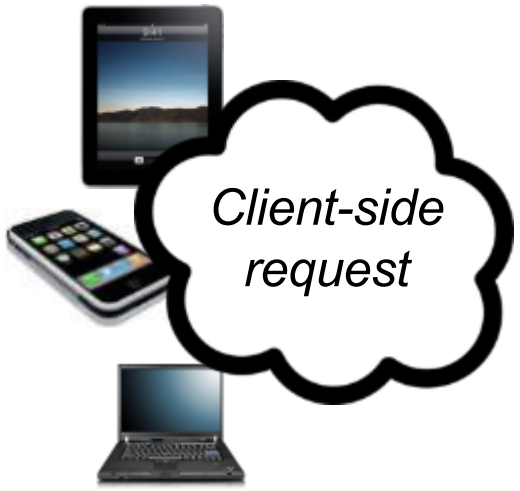
- "Database" is not part of the term, because there is no need (often no desire) on the part of a programmer to know that a database is on other end of a data request
  - Might be database (maybe relational, like Db2, or hierarchical, like IMS)
  - Might be a file system (such as VSAM in a z/OS system)
  - Might be a Hadoop-managed data store
  - Might be none of the above

=

It doesn't matter. Many application developers just want to invoke a data service of some kind (create, read, update, delete data) via a straightforward and consistent interface, *regardless of the mechanism by which the request is executed.*

REST (REpresentational State Transfer) is an architectural style that enables use of such an interface

RESTful services – client-side perspective

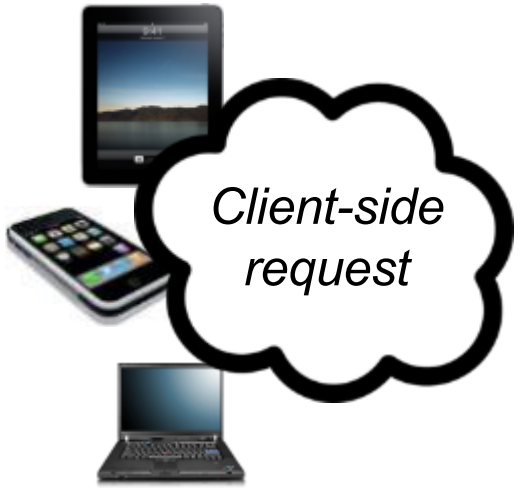


URI = Uniform resource identifier

`https://mysite.com/CustomerApp/getCustomer?cn=1234`

- With REST, a service is invoked by way of a URI, which is appended to the URL of an HTTP request
- If the URI is understood by the receiving server, the requested action is taken

What about data “payloads” (input/output) for REST calls?



URI = Uniform resource identifier

`https://mysite.com/CustomerApp/getCustomer?cn=1234`

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "1542 Main Street",
        "city": "Anytown",
        "state": "NY",
        "postalCode": "10021-1004"
    },
}
```

- Data associated with REST calls can be sent in JSON format (JavaScript Object Notation) – a series of name/value pairs

← **Example of output data payload in the form of a JSON document**

Data-as-a-service in a Db2 for z/OS context: the REST interface

Db2's native REST interface

- Introduced with Db2 12 for z/OS, retrofitted to Db2 11 via APARs PI66828, PI70477
- An extension of the Db2 distributed data facility (DDF)
  - DDF is Db2's interface to applications that access data via TCP/IP connections (application does not have to be outside of z/OS system – could be Java program in z/OS using type 4 driver)
  - The REST interface to Db2 is relatively new, but DDF is not – it has been powering high-volume, mission-critical client-server applications for 25+ years
  - A DDF cost-of-computing benefit: SQL statements executed by way of REST calls to Db2 run under preemptible SRBs in the DDF address space, and SQL executing under DDF preemptible SRBs is up to 60% zIIP-eligible
- Designed for high performance
  - IBM tests: over 100,000 trans/second through the Db2 REST interface

More on the Db2 for z/OS REST interface…

- It enables execution, by way of a REST call, of the package associated with a single static SQL statement (Db2 package: compiled form of SQL)
  - Single static SQL statement could be a data manipulation statement (e.g., SELECT, INSERT, UPDATE, DELETE) or a CALL to a Db2 stored procedure (sever-side data processing program)
    - If a stored procedure, probably want to use native SQL procedure, if possible (that's a stored procedure written in SQL PL, and it us up 60% zIIP-eligible when invoked through Db2 DDF)
- Not just REST service creation – also support for service discovery
  - Allows client-side developers to get information about available Db2 data services
- Also access control
  - Db2 REST request must be authenticated via ID and password, or with client-provided certificate (certificate is associated with an ID on server side)
  - ID must also have EXECUTE privilege for package associated with REST call

Turning a SQL statement into a RESTful service: two options

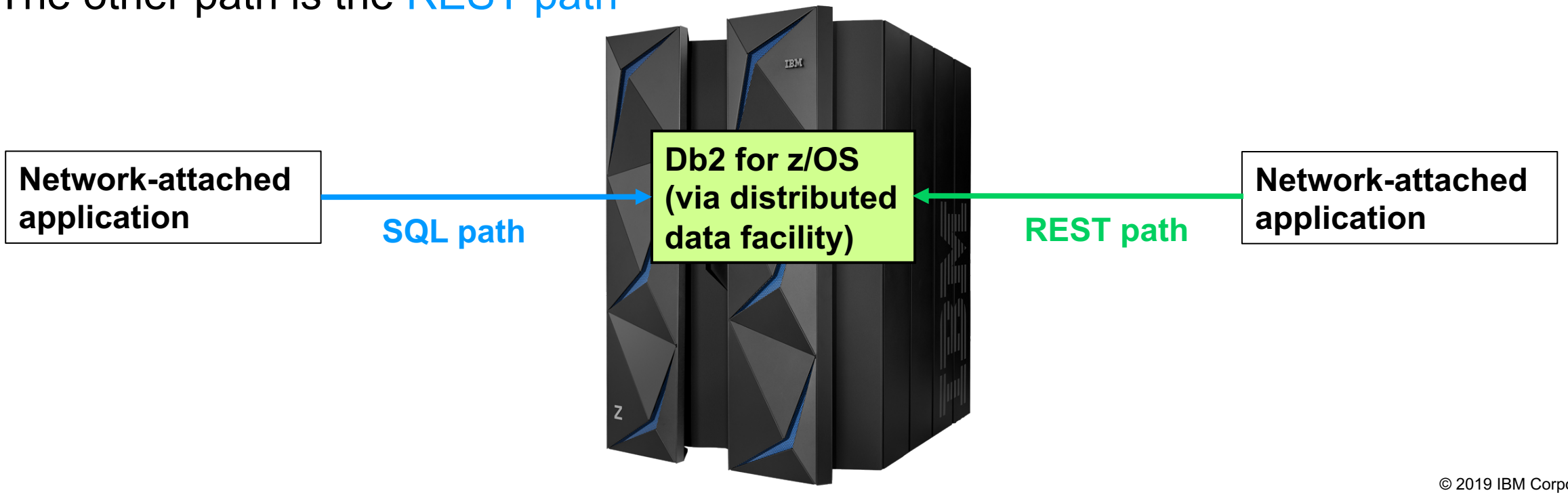
- Use the Db2-provided RESTful service called DB2ServiceManager
  - A RESTful service that can be used to create a RESTful service
- Use Db2 command BIND SERVICE
  - Enables creation of RESTful service by way of a batch job

(example)

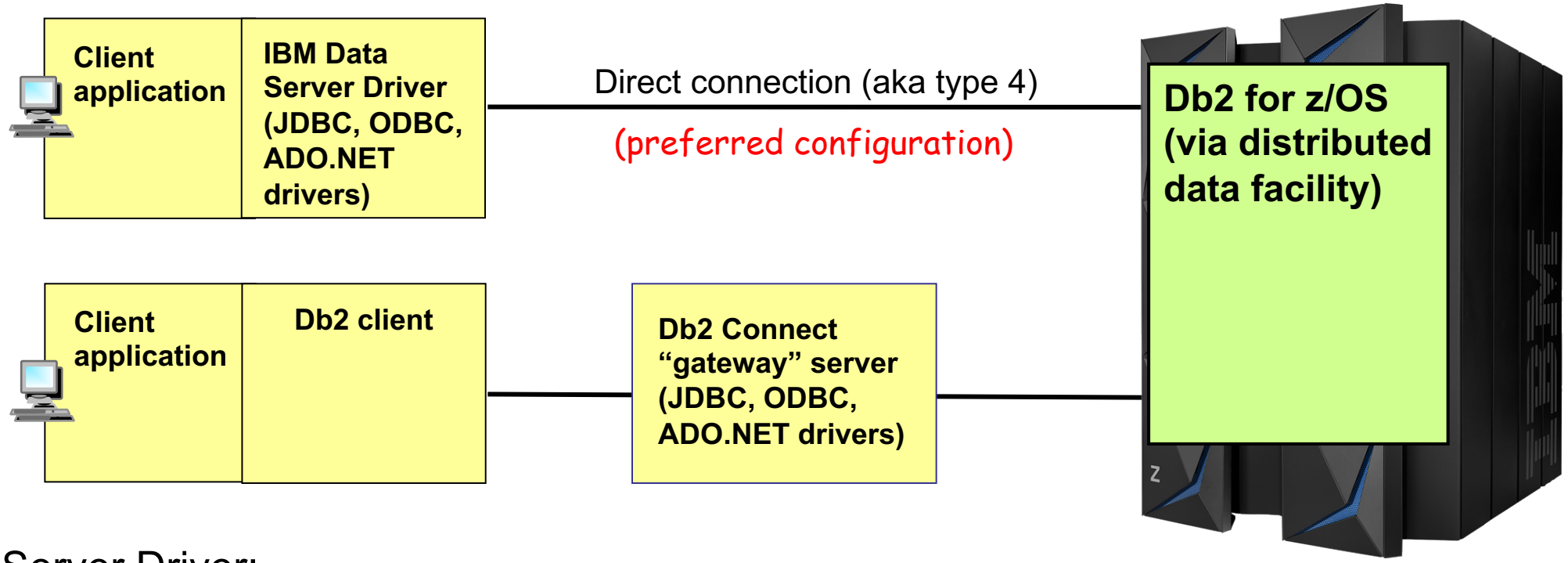
```
//CR8SRVC EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD  DSN=DB2A.SDSNEXIT,DISP=SHR
//         DD  DSN=DB2A.SDSNLOAD,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//DSNSTMT  DD  DSN=SYSADM.SERVICE.SQL(SELECT1),
//             DISP=SHR
//SYSTSIN  DD  *
 DSN SYSTEM(DB2A)

 BIND SERVICE(SYSIBMSERVICE) -
  NAME("simpleSelect1") -
  SQLENCODING(1047) -
  DESCRIPTION('return a list of deptname-
 based on input location')
/*
```

Important concept: there are now two paths into Db2 DDF

- The traditional path can be thought of as the SQL path
  - Technically, this is the DRDA path (distributed relational database architecture is Db2's distributed database protocol – IBM driver translates client-issued SQL to DRDA)
  - Client-side SQL typically in non-DBMS-specific form such as JDBC or ODBC form
- The other path is the REST path



SQL path to DDF: IBM Data Server Driver (or Db2 Connect)



- IBM Data Server Driver:
  - Runs on same server as client application
  - To use IBM Data Server Driver, you license Db2 Connect
    - Exception: "concurrent user" license for Db2 Connect requires use of Db2 Connect gateway server

The REST path – no SQL on the client side

```
POST http://mybank.com:4711/services/ACCOUNTS/getBalance
Body: { "ID": 123456789 }
```

REST call

REST client

HTTP + JSON

Discover
Invoke
Manage

DDF (Db2 distributed data facility)

- HTTP and JSON parsing
- Security checks
- Thread management

Db2 for z/OS (database services address space)

- SQL execution

HTTP response (data in JSON format)

```
Body:
{
   "FIRSTNAME" : "John",
   "LASTNAME"  : "Smith",
   "BALANCE"   : 1982.42,
   "LIMIT"     : 3000.00
}
```

```
SELECT C.FIRSTNAME,
C.LASTNAME, A.BALANCE,
A.LIMIT
FROM ACCOUNTS A,
CUSTOMERS C
WHERE A.ID = ?
AND A.CUSTNO = C.CUSTNO
```

SQL statement (in the form of a package)

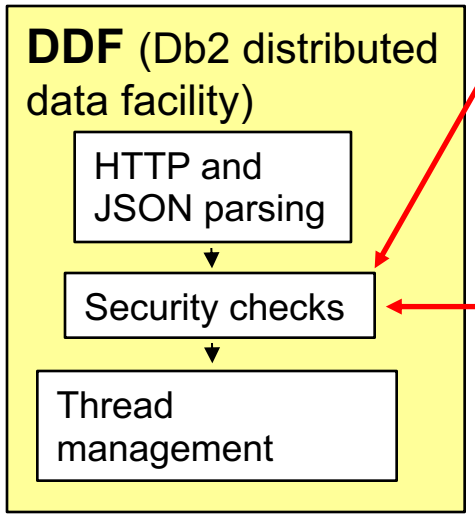
Additional information on the REST path to Db2 for z/OS

This could be Db2 server's secure port, if SSL connection required

```
POST http://mybank.com:4711/services/ACCOUNTS/getBalance
Body: { "ID": 123456789 }
```

REST call

JSON-format input consists of name-value pairs – "name" part(s) come from column name(s) or host variable name(s) used in SQL statement



Authorization ID always required – either provided in HTTP header or associated with client certificate (if certificate-based authentication used)

Authentication via password (provided in HTTP header) or client-issued certificate

JSON-format output consists of name-value pairs – "name" part(s) come from result set column name(s)

```
Body:
{
   "FIRSTNAME"  : "John",
   "LASTNAME"   : "Smith",
   "BALANCE"    : 1982.42,
   "LIMIT"      : 3000.00
}
```

HTTP response (data in JSON format)

```
SELECT C.FIRSTNAME,
C.LASTNAME, A.BALANCE,
A.LIMIT
FROM ACCOUNTS A,
CUSTOMERS C
WHERE A.ID = ?
AND A.CUSTNO = C.CUSTNO
```

SQL statement (in the form of a package)

Should application team use SQL or REST path to Db2?

- Note: these paths are not mutually exclusive – both can be used to access a given Db2 system
- Different circumstances could favor use of one path over the other

SQL path

- Db2 client software (IBM Data Server Driver or Db2 Connect) required on client side
- Leverages developers' SQL skills
- SQL statements can be dynamically constructed on client side, then issued to Db2
- Client control over transaction scope: client can issue multiple SQL statements, then commit to close out transaction
- Probably delivers best performance, scalability
  - Sysplex workload balancing, connection pooling, …

REST path

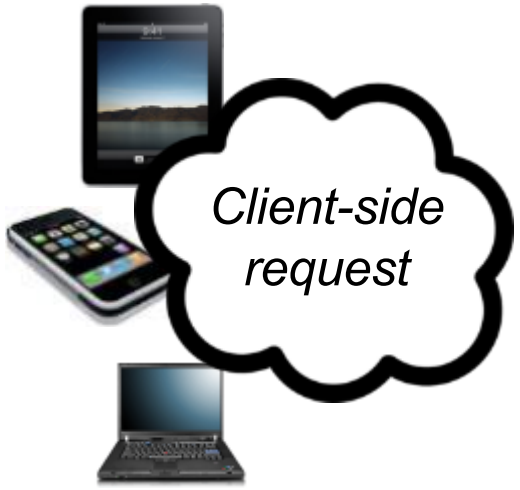
- No Db2 client-side software required for issuance of REST calls (i.e., no driver needed)
- No SQL statements issued from client side: REST calls trigger execution of static, server-side SQL statements
  - Though could pass SQL statement string as input to REST-invoked Db2 stored procedure
- No client control of transaction scope: each REST call is separate transaction from Db2 perspective
  - Stored procedures enable server-side control of transaction scope
- Data-as-a-service programming model is attractive to many developers

Where z/OS Connect fits in the picture

Clients have two ways to access Db2's REST interface

- One option: direct access to Db2's REST interface from client
- Another option: access Db2's REST interface through z/OS Connect
  - In that situation, Db2 is a *REST provider* to z/OS Connect
- Given that you're accessing Db2's REST interface either way, why use z/OS Connect?
  - z/OS Connect provides capabilities beyond Db2's for creating, managing, discovering, securing, and auditing Db2-provided RESTful services
  - z/OS Connect has features that can benefit client-side and server-side developers
    - Client-side: service discovery via the Open API Initiative's Swagger specification
    - Client-side: RESTful services can be invoked via full range of HTTP verbs (e.g., GET and PUT – Db2's native REST interface only supports POST), so REST calls can be more intuitive (see next slide)
    - Server-side: workstation-based tooling that facilitates creation of REST APIs from Db2 SQL statements
    - Server-side: z/OS Connect provides formatting flexibility for JSON output (see next slide)

Remember slide 7?



For Db2-serviced REST request to be in form other than POST, need z/OS Connect

`https://mysite.com/CustomerApp/getCustomer?cn=1234`

For REST request in GET form, input can be appended to URI (for POST-type request, input must be in request body)

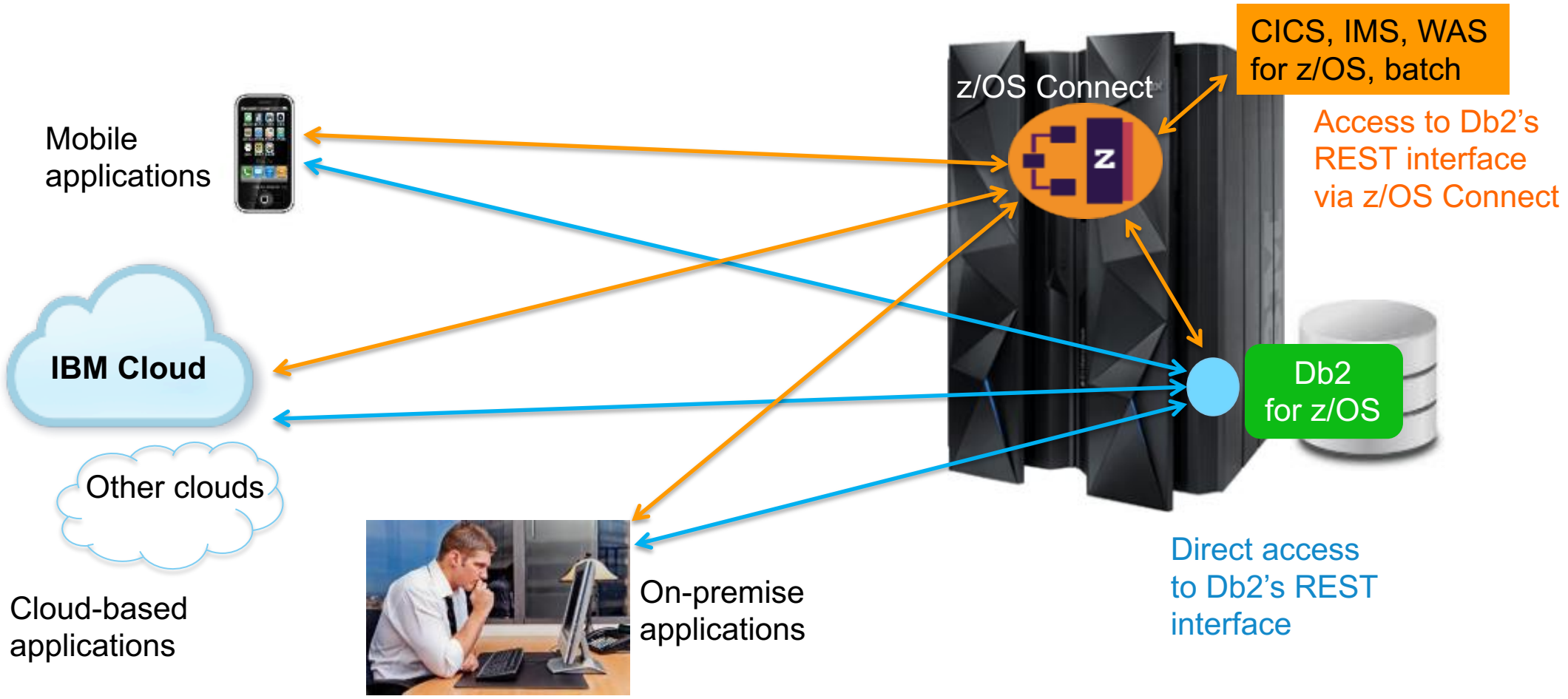
```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "1542 Main Street",
        "city": "Anytown",
        "state": "NY",
        "postalCode": "10021-1004"
    },
}
```

z/OS Connect provides flexibility in formatting output JSON document (for example, showing address components as sub-elements of "address")

A little more on z/OS Connect…

- z/OS Connect provides a single access point through which all kinds of z/OS-based programmatic assets can be exposed as RESTful services
  - In addition to Db2 SQL statements, through z/OS Connect you can REST-enable…
    - CICS transactions (might access Db2 data, might access VSAM files)
    - IMS transactions
    - WebSphere Application Server for z/OS transactions
    - Batch jobs
- Does going through z/OS Connect affect the cost of executing a Db2 SQL statement via a REST request?
  - Not much
    - Some additional CPU consumption, but z/OS Connect is written in Java, so additional mainframe MIPS consumed are zIIP MIPs

Accessing Db2's REST interface – the big picture



Robert Catterall
rfcatter@us.ibm.com