This guide is meant to educate IBM technical sales, and customers, on all the various ways mobile applications can connect to data and transactions on System z.

Contents

- Summary of z mobile connectivity options, including MobileFirst Platform

- Details
  - Push Notification
  - Why JSON
  - IBM API Management
  - CICS
  - IMS
  - DB2
  - WMB
How to use the z System Mobile guides…
We recommend reading these in this order…

1. Reference Architecture for Mobile Infrastructure on System z
   (March 18, 2014)
   - Components of a mobile architecture.
   - Mobile topology choices.
   - MobileFirst Platform in production.
   - MobileFirst Platform in dev/test
   - Scalability and performance considerations.
   - Conclusion

2. System z Mobile Connectivity Guide
   (June 2014)
   - Summary of z mobile connectivity options, including MobileFirst Platform Foundation
   - Details about
     - Push Notification
     - IBM API Management
     - CICS
     - IMS
     - DB2
     - WMB

3. Reference Architecture for Mobile Security on System z
   (November, 2014)
   - Introduction to the MobileFirst security products – what they do and how they relate to System z.
   - Building a Secure Enterprise Mobile environment using the MobileFirst Security products.
   - Use Cases and Reference Architectures.
Summary of Connectivity Options
What is a Mobile App?

**Browser Access**
Written in HTML5 JavaScript and CSS3. Quick and cheap to develop, but less powerful than native.

**Hybrid Apps - Mixed**
User augments web code with native language for unique needs and maximized user experience.

**Native Apps**
Platform-specific. Requires unique expertise, pricy and long to develop. Can deliver higher user experience.
What is a Mobile App?

Systems of Interaction

Systems of Engagement

Systems of Record

Primary Path
Secondary Path

Web
Hybrid
Native

Application Server

APIs, JDBC, etc.
Mobile App Connectivity to System z – without an app server

Available
Available but unlikely
In Plan
Not Available
Mobile App Connectivity to System z -- without MobileFirst Platform

This is the most typical mobile app configuration.

WebSphere Application Server

Transactions with Business Logic

- JDBC
- JMS
- HTTP

Available
In Plan
Not Available

Web Services
- Salesforce.com
- UPS
- AT&T
- Any SOAP service

z/Linux

- DB2 LUW
- MQ/WMB

z/OS

- DB2/IDAA
- MQ/WMB
- CICS
- IMS

Anything that provides a SOAP or REST interface

Native, Web, or Hybrid apps
MobileFirst Platform Server - Adapters

Universality
• Supports multiple integration technologies and back-end information systems

Read-only & Transactional Capabilities
• Adapters support read-only and transactional access modes to back-end systems

Fast Development
• Defined using simple XML syntax
• Easily configured with JavaScript APIs

Security
• Flexible authentication APIs for back-end connections
• Connected user identity control

Caching
• Leveraged to store data retrieved from back-end

Transparency
• Uniform exposure of back-end data for all adapter types
Mobile App Connectivity to System z -- via MobileFirst Platform

Adapters
- Third-party Adapters
  - AT&T
  - WMB
- Core Adapters shipped with MobileFirst Platform
  - SQL
  - JMS
  - HTTP
  - Cast Iron

WebSphere Application Server

MobileFirst Platform Server

MobileFirst Platform apps

Available
In Plan
Not Available

z/Linux
- DB2 LUW
- MQ/WMB

z/OS
- DB2/IDAA
- MQ/WMB
- CICS
- IMS
- Anything that provides a SOAP or REST interface

Web Services
- Salesforce.com
- UPS
- AT&T
- Any SOAP service

© 2014 IBM Corporation
MobileFirst Platform Server on System z

- z/OS
- z/Linux

MobileFirst Platform Server:
- Server-side Java App Code -- WAS
- JSON Translation
- Authentication
- Adapter Library
- Application Center Enterprise App Store
- MobileFirst Platform Console
- Push Notifications
- Analytics

Device Runtime:
- *Apache Cordova (formerly known as PhoneGap)
MobileFirst Platform Platform support

**MobileFirst Platform Studio**
- Supported Development Tools
  - Rational Application Developer for WebSphere Software 8.5.1
  - Rational Developer for Power Systems Software 8.5
  - Rational Developer for zEnterprise 8.5.1
  - Rational Software Architect 8.5.1
  - Rational Software Architect for WebSphere Software 8.5.1
- Supported OS
  - Windows
  - SOD for x/Linux.
  - SOD for z/OS WAS.

**MobileFirst Platform Server**
- Supported application server
  - WebSphere 7 & 8 (Distributed | zLinux)
  - WebSphere 8.5 Liberty Profile (included for Dev only), ND/Base (not included)
  - Apache Tomcat 7 (Linux | Windows | Mac OS X [development])
- Supported databases
  - DB2 Enterprise Server Edition V9.7 or later (DB2 LUW)
  - Apache Derby, SDK 10.8 (included), or later
  - Oracle 11g Database server, Standard or Enterprise Editions, release 1 or later
  - MySQL 5.1
- Supported OS
  - z/Linux SLES 10,11, RHEL 5,6.
  - See URL below for more.
MQTT Messaging Connectivity to System z

Available
In Plan
Not Available

See notes for rationale for not supporting this path.
System z subsystems are enabled for mobile access today

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>JDBC?</th>
<th>SOAP?</th>
<th>JSON?</th>
<th>Notes</th>
</tr>
</thead>
</table>
| DB2       | YES  | YES  | YES  | ▪ SQL calls from mobile app return data in XML format.  
▪ Support for native store JSON format data, and MongoDB API with DB2 10 and above [https://www.ibm.com/developerworks/data/db2preview/](https://www.ibm.com/developerworks/data/db2preview/) |
| CICS      | N/A  | YES  | YES  | ▪ Web services (SOAP/HTTP) supported with CICS TS V3.1 and above.  
▪ RESTful services (JSON/HTTP) supported with CICS TS V4.2 and above + the CICS TS Feature Pack for Mobile Extensions V1.0, (GA 14 June 2013). |
| IMS       | YES  | YES  | Yes  | ▪ [IMS Mobile Feature Pack (with z/OS Connect)](https://www.ibm.com/servers/z/osconnect) provides REST and JSON support.  
▪ IMS trans can be invoked via SOAP. IMS databases can be invoked today via SQL. |
| TPF       | N/A  | YES  | No   | z/TPF supports web services calls using SOAP over HTTP and MQ. |
| z/VSE     | N/A  | YES  | No   | z/VSE supports web services calls using SOAP over HTTP and MQ. |
Push Notifications
MobileFirst Platform Server - Unified Push Notifications Architecture

- **Polling Adapters**
- **Message based Adapters**
- **Unified Push API**
- **Notification State Database**
- **User-Device Database**
- **iOS Dispatcher**
- **Android Dispatcher**
- **SMS Dispatcher**

- **iOS Push API**
- **Android Push API**
- **Broker API**

- **Apple Push Servers (APN)**
- **Google Push Servers (GCM)**
- **SMS/MMS Brokers**
Users Register for Push Notifications

1. User gives permission to receive push notifications.
2. MobileFirst Platform is notified of user preference.
3. Device OS registers user preference.

MobileFirst Platform Server

MobileFirst Platform Push Service

WAS

DB2

LUW
Push Notification from CICS via MobileFirst Platform

1. User configures event notification with URL to be notified, and data to be sent.

2. An event occurs. CICS invokes the registered URL with data.


Server-side Application Code

MobileFirst Platform Server

WAS

CICS
User configures a data trigger.

MobileFirst Platform Server

MobileFirst Platform notifies mobile devices

Server-side Application Code

Another process updates the data, invoking the trigger. DB2 invokes the URL.

DB2

WAS
Push Notification from IBM Integration Bus (WMB) via MobileFirst Platform

1. User utilizes a pattern to expose backend events via a web service.
2. MobileFirst Platform adapter generated by pattern deployed to MobileFirst Platform.
3. IIB sends notification messages to MobileFirst Platform.
How MobileFirst Platform sends Push Notifications
Why JSON
Why JSON instead of SOAP? Eliminate mapping and transformation through the tiers.

Customer pain points with SOAP
- SOAP and the web services standards can become complex in mobile scenarios
  - RESTful services preferred as it relies on HTTP (verb) and the URI (noun)
- XML can be bandwidth and processing intensive, and pre-req parsers
  - JSON is a lightweight data interchange format, with simple types
  - Suitable to serialise / deserialise a data structure with minimal effort in JavaScript and many other languages
IBM API Management
(aka WebSphere Cast Iron)
IBM WebSphere Cast Iron
Simple connectivity to Packaged Apps and Cloud Services

Total Connectivity

Deployment Flexibility

Repeatable Success

For All Types of Projects

UI Mashups

Process Integration

Data Migration

Cloud Service

Physical Appliances

Virtual Appliances

TIP Exchange

TIP Development Kit

TIP Community
IBM Mobile Foundation – MobileFirst Platform & Cast Iron Bundle
Connects MobileFirst Platform Apps with Cloud & On Premise Applications in Days

*Note: Connectivity to System z is supported but System z is not a supported runtime environment today
### Scenario 1: MobileFirst Platform Application makes request for enterprise data

- **Request from Mobile app for customer data from SAP & salesforce**
  1. User invokes action on Mobile application to request data
  2. MobileFirst Platform Server invokes it’s WL Cast Iron Adapter to send customer data request to Cast Iron
  3. Cast Iron receives request & invokes connectivity with salesforce.com and SAP to extract customer data
  4. Cast Iron sends customer data to MobileFirst Platform Server
  5. MobileFirst Platform Server delivers data to Mobile application on the device
How it works

Scenario 2: Cast Iron provides real-time notification to Mobile applications

- **Instant notification to Mobile apps on enterprise data changes** (e.g. customer shipment delay in SAP)
  1) Manufacturing is delayed and this causes change of customer shipment date in SAP
  2) SAP is configured to push out an IDOC indicating this date change. Cast Iron TIP picks up DOC & transforms the data.
  3) Cast Iron TIP transforms data to MobileFirst Platform format (JSON) and sends to MobileFirst Platform Server via HTTP (REST)
  4) IBM MobileFirst Platform server receives message and sends notification to Mobile app using appropriate protocol for Android and iOS apps
IBM WebSphere Cast Iron

- Invoked via MobileFirst Platform adapter or independently via SOAP. Manages the protocol and data required to interface with third-party APIs that bring valuable services to your web and mobile applications. Cast Iron is not mobile-only, it supports any web applications that need integration of APIs.

- Supports these roles:
  - **API publishers** can create APIs that let web apps use their data and transactions.
    Value-add: Manage subscriptions to your APIs, get statistics, limit entitlements.
  - **External app creators** can use those APIs in their apps.
    Value-add: Use external web APIs without coding. CastIron Studio integrates with MobileFirst Platform Studio (or RDz) to allow drag-and-drop integration with these services.
  - **Internal App creators** can create orchestrations using APIs from different internal applications.
    Value-add: bridge incompatible systems without programming.

- Contains a catalog of available services for app developers to use:
  - SaaS apps: Salesforce.com, Oracle CRM, Taleo.
  - Packaged apps: SAP, Oracle PeopleSoft EBS.
  - Web apps: UPS, Amazon, etc.
  - DBs, flat files/FTP

- Runs in either:
  - An appliance run on-premises.
  - A VMware image run on premises.
  - A hosted service in the IBM cloud (Cast Iron Live)
CICS Connectivity Options without MobileFirst Platform

- z/OS
- CICS

- HTTP
- MQ
- Atom Feeds
- REST
- TN3270
- 3270
- SNA
CICS Connectivity Options with MobileFirst Platform

- HTTP
- Web Services
- Atom Feeds
- REST
- TN3270
- 3270
- HATS Service Flow
- Web Services
- MobileFirst Platform Server

© 2014 IBM Corporation
MobileFirst Platform gives a Mobile UI to CICS Services

NEW – CICS Mobile Feature pack provides direct JSON connection to CICS.

IBM CICS TS feature pack for Mobile Extensions (CICS TS 4.2+)
Further simplifies MobileFirst Platform-CICS connectivity with JSON support in CICS.
NEW -- Run the WAS Liberty profile in CICS to give JSON capabilities to CICS Java apps.
NEW -- Run z/OS Connect in the WAS Liberty profile in either z/OS or CICS to give access to CICS apps.
Why JSON?

Existing options to connect from IBM MobileFirst Platform to CICS TS V3.1+
- SOAP, or write your own WEB API wrapper program

Customer pain points with SOAP
- SOAP and the web services standards can become complex in mobile scenarios
  - RESTful services preferred as it relies on HTTP (verb) and the URI (noun)
- XML can be bandwidth and processing intensive, and pre-req parsers
  - JSON is a lightweight data interchange format, with simple types
  - Suitable to serialise / deserialise a data structure with minimal effort in JavaScript and many other languages

CICS TS Feature Pack for Mobile Extensions (CICS TS 4.2+):
- RESTful service pipeline to accept JSON in the HTTP body
  - Converts JSON to COBOL, C/C++, PL/I application data for easy consumption
  - Tooling to enable conversion from JSON schema or language structures
- General service for JSON conversion, called using LINK command
  - Used in combination with WEB commands to send RESTful service requests
- Support JSON feature in Liberty profile provided in CICS TS V5.1
  - Natural for Java applications using Servlet as the entry point
IMS
IMS Mobile Enablement
IMS Mobile Enablement With z/OS Connect

Mobile Devices
IMS Mobile Apps

Web / Desktop

z/Linux
MobileFirst Platform Server
HTTP Adapter
SOAP Adapter
SQL Adapter

IBM z/OS
WAS Liberty Profile
IMS Mobile Feature Pack
z/OS Connect

Transaction Manager
IMS Application
Database Manager

IMS Connect

JSON

HTTP Adapter
SOAP Adapter
SQL Adapter

IMS DB

ISPF

JSON

SOAP, JDBC, WAS

JSON

HTTP Adapter
SOAP Adapter
SQL Adapter
DB2 z/OS
DB2 LUW on Linux
New Era Application Characteristics

- Applications evolve rapidly as the needs for mobile and Web presence try to keep pace with internet user needs

- Application developers are increasingly looking for solutions that allow nearly continuous integration of application changes
  - Amazon.com allows 1000’s of their developers to check in product code changes daily…
  - Developers resist solutions that require delays to sync up with DBA change windows

- NoSQL JSON stores are appealing to these developers:
  - JSON schema can be evolved rapidly without intervention by DBAs or data modelers.
  - Objects like “shopping cart” in these applications really aren’t used outside the Web application, so there is no need to interlock closely with the rest of the enterprise data model.
  - JSON offers a very simple and elegant model for persisting Java or JavaScript objects, without needing a heavy-weight persistence solution like OpenJPA or Hibernate.
  - Performance and scalability is very good for JSON
    - Store a single JSON document representing the object versus
    - Store “n” rows in relational as a “normalized” object.
NoSQL DB2 -- The Best of Both Worlds: Agility with a Trusted Foundation

- Interoperate seamlessly with modern applications
  - Flexible schemas allow rapid delivery of applications
- Preserve traditional DBMS Capabilities, leverage existing skills and tools:
  - Multi-statement Transactions
  - Management/Operations
  - Security
  - Scale, performance and high availability
- Extend with Advanced features (future)
  - Temporal semantics
  - Full Text search
  - Multi-collection joins
  - Combine with Enterprise RDBMS data
JSON (NoSQL) API

**Java Driver** that translates API calls to SQL + function invocations

- Supports Transactions
- Batches insertions
- Fire-forget inserts (fast)
- Indexing
- Time travel query
- Smart Query re-write
- Good performance with Inline LOBS
- Java command line
NoSQL JSON API vs SQL: Simple Example

1) Create a customer collection / table.
   \[\text{createCollection("customers")} \quad \text{CREATE TABLE customers \(_id\) VARBIN(12) data BLOB(50K)}\]

2) Insert all your customers as JSON documents. For example, one insert might contain this document:
   \[
   \{ \text{name:"Joe", age:25, phone:["555-666-7777", "444-789-1234"],} \\
   \text{ address:{ street:"ABC st",} \\
   \text{ zipcode:"95141" } } \}
   \]
   \[\text{db.customers.insert({name:"Joe"") \quad \text{INSERT INTO customers \quad VALUES ( <binary JSON> )}}\]

3) Look for customers in zipcode 95141.
   \[\text{db.customers.find(} \quad \text{SELECT DATA FROM customers} \quad \text{WHERE JSON\_VAL} \quad \text{(json\_data,'address.zipcode','s:5')} \quad \text{= '95141'}}\]

4) Improve performance by creating index on zipcode.
   \[\text{db.customers.ensureIndex(} \quad \text{CREATE INDEX idx1} \quad \text{ON customers} \quad \text{(JSON\_VAL(json\_data,'address.zipcode','s:5'))}}\]
NoSQL (MongoDB) JSON Wire Listener

- Built on JSON API
- Leverage community
- Immediate reach to more applications and developers
- Presence in “New style apps”
- (Future) Extend existing community drivers with DB specific features:
  - Multistatement commit scope
  - Temporal
  - Geo-spatial
JSON and DB2 – Complementary Technologies

- Does NoSQL mean NoDBA? NoDB2?
  - Definitely Not - The relational database isn’t going away anytime soon.
  - We see JSON as becoming a complementary technology to relational

- Transactional atomicity is essential for mission critical business transactions.
  - DB2 JSON Store solution brings commits, transaction scope

- Future Direction – access JSON data directly with SQL.
  - JSON data co-exists with relational columns in the same table
  - Enables the proper balance between fixed schema and flexible schema
**JSON Store Tech Preview Summary**

**MongoDB Wire Listener**
- Leverage NoSQL community drivers for data access layer
- IBM provided Java Driver for JSON API
  - Java Driver supporting JSON API for data access layer
  - Transactions
  - Parametric SQL statements (Delete, select)
  - Temporal tables
- Insert, Update, Delete, Select support
  - Select projection list
  - Batching, order by, paging queries (API only)
  - Fire and forget inserts
  - Limited aggregate functions (group-by / having)

**Indexing support in API**
- Primary index and secondary single value index

**Import/Export**
- Import/Export from/to MongoDB export JS-files

**Command line tools**
- Execute JSON queries and display results

**Install**
- Files and scripts that are part of server and DS Driver

**Undocumented Server Capabilities**
- JSON_VAL Built-in function
- Index on Expression with BLOB input

**Platforms:**
- Available on DB2 z/OS V10
- Available on DB2 z/OS V11 in June 2014 with “DB2 Accessories Suite for DB2 11”

---

**Applications**
- Java
- PHP
- NodeJS

---

**DB2 Engine**
- IoE w/ BLOB in the expression
- JSON_VAL()
- JSON_UPDATE()
- JSON_TABLE()
- JSON UDFs
  - Built-in, supports extraction of (SQL) values from BSON

---

**DB2 Wire Protocol**
- BSON Wire Protocol

---

**NoSQL JSON Wire Listener**
- IBM extension to enable DB2 features
- Community Provided Drivers
IBM WebSphere Liberty
z/OS Connect
z/OS Connector (z/OS Connect)

What is it and what are the benefits for customers?

z/OS Connect is a Liberty based gateway that provides a secure and simple way to discover and call in to application assets/infrastructure on z/OS from Mobile/Web/Cloud applications using RESTful services.

Benefits include:

- Fast on-ramp for z/OS customers to discover and reach z/OS applications securely/simply using RESTful services. Service references can be copied from z/OS Connect and stored in any repository – cloud based (such as IBM Cloud OE) or mobile based (such as IBM MobileFirst Platform, API Management) or any other web technology

- Light-weight and modular providing flexibility to run multiple copies on the same or different z/OS systems and assign higher/lower priority to specific Liberty servers

- Integrated with z/OS management makes the operations of the environment automated and consistent with the environments it is exposing

- Provides ability to standardize on security access for calling in to z/OS applications in all major environments - CICS, IMS, batch, Unix System Services, and ISV software. Supports SAF-based security integration allowing for individual z/OS Connect services to have unique sets of authorized users.

- Provides ability to track requests from cloud, mobile, web based external requestors using standard z/OS mechanisms like SMF. Fulfills audit/chargeback needs for access to z/OS applications

- Provides ability to prioritize requests within a single Liberty server - based on URIs and z/OS Connect service names – using z/OS WLM.
Introducing WebSphere Liberty z/OS Connect

- There are different ways to get mobile requests to the existing z/OS assets, but each have their own mechanisms for setup, security and data mapping.

- z/OS Connect is a connectivity solution built on Liberty Profile for z/OS. It provides a common and consistent way to access z/OS data, applications and resources.

- z/OS Connect has a RESTful interface and accepts JSON data. It has the ability to convert data and invoke the requested backend program or application.

- z/OS Connect is configurable so only the z/OS resources you want made available are accessible. z/OS Connect has a discovery function so the accessible resources may be included in your broader API Management strategy.
Why z/OS Connect?

It is another component to configure and maintain in your environment. So what value does it bring?

- Provides a common and consistent entry point for mobile access to one or many backend systems.
- Provides simple API Management. Determine the transactions you want “discoverable” by mobile developers.
- Provides single point for security and auditability.
- Simplifies mobile development by providing RESTful and JSON interfaces to z/OS transactions.
- Java, so it runs on specialty engines.

You *can* enable Mobile access without z/OS Connect. z/OS Connect simplifies and makes the environment more consistent and manageable.
Different Delivery Approaches

z/OS Connect is delivered with WAS z/OS, CICS and IMS … objective is to provide different approach paths depending on what you have:

**WAS z/OS**

Delivered as function that runs inside Liberty Profile z/OS. Initially will use WOLA (WebSphere Optimized Local Adapters) to access backend.

**CICS**

Delivered as part of Liberty Profile that runs inside of CICS region. Will use JCICS interface to access CICS functions

**IMS**

Initially this ends up looking just like the WAS z/OS approach: that is, Liberty Profile z/OS with z/OS Connect inside. Difference is this z/OS Connect will be able to talk to IMS Connect for access into IMS
Context Within Overall Mobile Architecture

z/OS Connect is a piece of the overall Mobile architecture

Users of z/OS Connect would access through normal corporate firewall infrastructure.

IBM MobileFirst Platform to provide application management, security and operational governance for mobile applications.

z/OS Connect would be behind the secure firewall, and on LPARs along with backend systems.
NEW -- Run z/OS Connect in the WAS Liberty profile in either z/OS or CICS to give access to CICS apps.
Okay, really … what is “z/OS Connect?”
Let’s take a high-level tour of what z/OS Connect is and how it operates …

1 – Device or systems (including cloud systems, such as IBM Bluemix) seek z/OS resources.

2 – z/OS Connect operates in a Liberty Profile for z/OS server instance. It is configured as a “feature” in Liberty Profile for z/OS.

3 – Liberty Profile server.xml updated with configuration elements to indicate what backend programs or applications you want made accessible.

4 – For data conversion you supply a bind file that allows z/OS Connect to understand how JSON is to be mapped to a backend COBOL, PL/I or C data structure.

5 – Devices or systems may query z/OS Connect and get a list of configured services. They may also query returned service for details on the service.

6 – Devices or systems use the RESTful interface of z/OS Connect and send request data as a JSON payload.

7 – z/OS Connect converts the data to the required format and invokes the backend program or application.

8 – For security, z/OS Connect may use SAF; for activity recording, z/OS Connect uses SMF.
**IBM z/OS Connect under WAS**

**z/OS Connect**
A service that encapsulates calling z/OS target applications using REST calls. z/OS Connect will support JSON payloads for calls from external cloud or mobile-based clients and will enable the conversion of the payload to the target program’s expected format. It will also provide the response payload conversion from a byte array into JSON format before returning the response to the caller.
z/OS Connect Liberty under CICS

Same z/OS Connect implementation – the CICS JCICS service provider handles requests targeted to existing CICS programs. CICS provides interceptors to integrate z/OS Connect with CICS security.
z/OS Connect Liberty for IMS

Same z/OS Connect implementation – WOLA or the IMS Connect service provider handles requests targeted to existing IMS transactions and data.
Request a **list of all services known in a server** – HTTP GET (returns JSON response)

```
http://<hostname>:<port>/zosConnect/services/
<JSON Out>
```

Request **information about a single service** – HTTP GET (returns JSON response)

```
http://<hostname>:<port>/zosConnect/services/<name>
<JSON Out>
```

Request **z/OS Connect service invoke** – HTTP POST or PUT (received JSON requests / returns JSON response)

```
http://<hostname>:<port>/zosConnect/services/<name>?action=invoke
<JSON In>
<JSON Out>
```

Request **service invoke with parameters** – HTTP POST or PUT (received JSON requests / returns JSON response)

```
http://<hostname>:<port>/zosConnect/services/<name>?action=invoke&<PARMS>
<JSON In>
<JSON Out>
```

Call service **using invokeURI** – HTTP GET (returns JSON response)

```
<JSON Out>
```
Request a list of all services known in a server – HTTP GET (returns JSON response)

```
http://<hostname>:<port>/zosConnect/services/
<JSON Out>

{
    "zosConnectServices": [ 
        {
            "ServiceURL": "http://host:port/zosConnect/services/HelloWorld",
            "ServiceName": "HelloWorld"
        },
        {
            "ServiceURL": "http://host:port/zosConnect/services/GoodbyeWorld",
            "ServiceName": "GoodbyeWorld"
        }
    ]
}
```
z/OS Connect – How it works … sample requests

Request **information about a single service** – HTTP GET (returns JSON response)

```plaintext
http://<hostname>:<port>/zosConnect/services/HelloWorld

<JSON Out>

{
    "zosConnectServiceName": "HelloWorld",
    "zosConnectServiceURL":
        "http://host:port/zosConnect/services/HellowWorld",
    "zosConnectServiceInvokeURL":
        "http://host:port/zosConnect/services/HellowWorld?action=invoke",
    "zosConnectProvider": "WOLA-1.0",
    "registerName": "BATCH01",
    "serviceName": "COBLPGM1",
    "DataXformProvider": "jsonByte-1.0"
}
```
Users of the z/OS Connect Function

• **Mobile phones**
  What people traditionally think of when “mobile computing” is discussed.

• **Tablets**
  Related to mobile phones in terms of how they function

• **Cloud-provisioned services**
  A growing consideration … these are programs that are part of a provisioned cloud service that seek information using standard protocols and data formatting. IBM’s Bluemix is an example of this.

• **Traditional workstations**
  Any network-connected system or device

The key is *not* the specific type of device or program … the key is what service protocol it uses and what data format it passes

If RESTful and JSON, it can use z/OS Connect
More Information

- z/OS Connect Techdocs page
WebSphere Message Broker
IBM integration Bus: MobileFirst Platform mobile services patterns

- Mobile enable any enterprise service in 2 clicks!
- Build robust solutions with integrated caching and security
- Push data to mobile users from enterprise applications
- Create end-to-end mobile solutions for Microsoft .NET

- Four new IIB development patterns
  - Fully integrated in IIB toolkit
  - Generate MobileFirst Platform adapter, test application and supporting IIB flows
- Simple Service to mobile – WSDL based
  - Resource access from mobile - Controlled access to enterprise data as a resource: methods for read create, update and delete
  - Microsoft .Net service pattern
  - Queue based Push Notification

IBM Integration Bus* - formerly WebSphere Message Broker
More
DataPower Gateway

Security, Control, Integration & Optimization of mobile workload

IBM DataPower Gateway Appliance

- SSL Offload
- Threat Protection
- Rate Limiting
- Validation, Filtering
- now with Native JSON Support**
- Authentication
- z/OS identity propagation
- Authorization
- OAuth 2.0
- Security Token Translation (e.g. SAML)
- Transformation
- Content-Based Routing
- Intelligent Load Distribution
- Response Caching Locally or to XC10 **

Enhanced form-based authentication support for quick integration with **MobileFirst Platform applications** running on mobile devices **

Ready-to-use configuration pattern as reverse proxy & security policy enforcement point in front of **MobileFirst Platform Server**

© 2014 IBM Corporation

** Available in DataPower firmware version 6.0
WebSphere Message Broker MobileFirst Platform mobile services patterns

- Mobile enable any enterprise service in 2 clicks!
- Build robust solutions with integrated caching and security
- Push data to mobile users from enterprise applications
- Create end-to-end mobile solutions including for Microsoft .NET

Lightweight MQ for Key Enterprise Messaging

- MQTT clients augment MobileFirst Platform capabilities:
  - Reliable messaging that conserves battery power and reduces network traffic
  - Clients are linked into the app by the app developer
  - MobileFirst Platform Server provides security, app management, statistics gathering etc.

DataPower’s Role with Mobile

- What’s new (already GA)
  - OAuth support in DataPower v5.0, enhanced in 2Q12
  - Enhancements to JSON support in DataPower v5.0

- Directions
  - JSON schema validation in DataPower 6.0
  - JSONiq (JSON Query Language) support in DataPower 6.0

MessageSight appliance

- Persist messages across power failure
- Millions non-persistent messages per second
- 400k persistent messages per second
- Developer-friendly
- MQTT and JMS API support
Backup
MobileFirst Platform Studio with RDz -- a complete set of System z Development and Test capabilities for Mobile apps

Integration with Team Concert for Lifecycle and Source Management

Access to typical System z sub-system functionality in z/OS, CICS, IMS, DB2, WAS

Integration with RD&T for flexible access to System z environment

Robust Mobile Development in conjunction with MobileFirst Platform

Integration with Fault Analyzer for Dump Analysis

Integration with File Manager and Fault Analyzer for file and test data handling and Dump Analysis

Integration with Asset Analyzer for Application Understanding and Impact Analysis
MobileFirst Platform architecture

**MobileFirst Platform Server**
- User authentication and mobile trust
- Mashups and service composition
- JSON Translation
- Adapter Library for backend connectivity

**Build Engine**
- MobileFirst Platform Studio
  - HTML5, Hybrid, and Native Coding
- Optimization Framework
- Integrated Device SDKs
- 3rd Party Library Integration

**MobileFirst Platform Application Center**
- MobileFirst Platform Server
- Development Team Provisioning
- Enterprise App Provisioning and Governance
- App Feedback Management

**Client-Side App Resources**
- Direct Update
- Mobile Web Apps
- Unified Push Notifications

**Device Runtime**
- Cross-Platform Compatibility Layer
- Server Integration Framework
- Encrypted and Syncable Storage
- Runtime Skinning
- Reporting for Statistics and Diagnostics

**MobileWeb**
- Mobile Web
- Desktop Web

**SDKs**
- iOS
- Android
- Blackberry
- Windows Phone
- Windows 8
- Java ME
- Mobile Web
- Desktop Web

**Application Code**
- MobileFirst Platform Studio
- MobileFirst Platform Server
- MobileFirst Platform Application Center

**Enterprise Backend Systems & Cloud Services**
- Build Engine
- MobileFirst Platform Studio
- MobileWeb
- SDKs
- MobileFirst Platform Application Center
- Device Runtime

**MobileFirst Platform Console**
- Reporting and Analytics
- Push /SMS Management
- App Version Management

**Development Team Provisioning**
- Enterprise App Provisioning and Governance
- App Feedback Management

**Public App Stores**
- Android
- Blackberry
- Java ME
- Mobile Web
- Desktop Web

**Push /SMS Management**
- Reporting and Analytics
- App Version Management
MobileFirst Platform overview

**MobileFirst Platform Studio**
The most complete, extensible environment with maximum code reuse and per-device optimization

**MobileFirst Platform Server**
Unified notifications, runtime skins, version management, security, integration and delivery

**MobileFirst Platform Runtime Components**
Extensive libraries and client APIs that expose and interface with native device functionality

**MobileFirst Platform Console**
A web-based console for real-time analytics and control of your mobile apps and infrastructure
WebSphere Portal can “refactor” existing Web apps for mobile.

WebSphere Portal/WCM and IBM Worklight used together can extend the capabilities and reach of an exceptional web experience.

IBM Worklight
- Access device specific capabilities (camera, contacts, etc...)
- App Store Presence
- Application management
- Mobile Notifications

WebSphere Portal/WCM
- Multiple Applications, Content, Roles, Personalization, Customization

WCM = Web Content Manager
MQTT - (MQ Telemetry Transport) clients augment MobileFirst Platform capabilities:
- Reliable messaging that conserves battery power and reduces network traffic
- Clients are linked into the app by the app developer

MobileFirst Platform Server provides security, app management, statistics gathering etc.