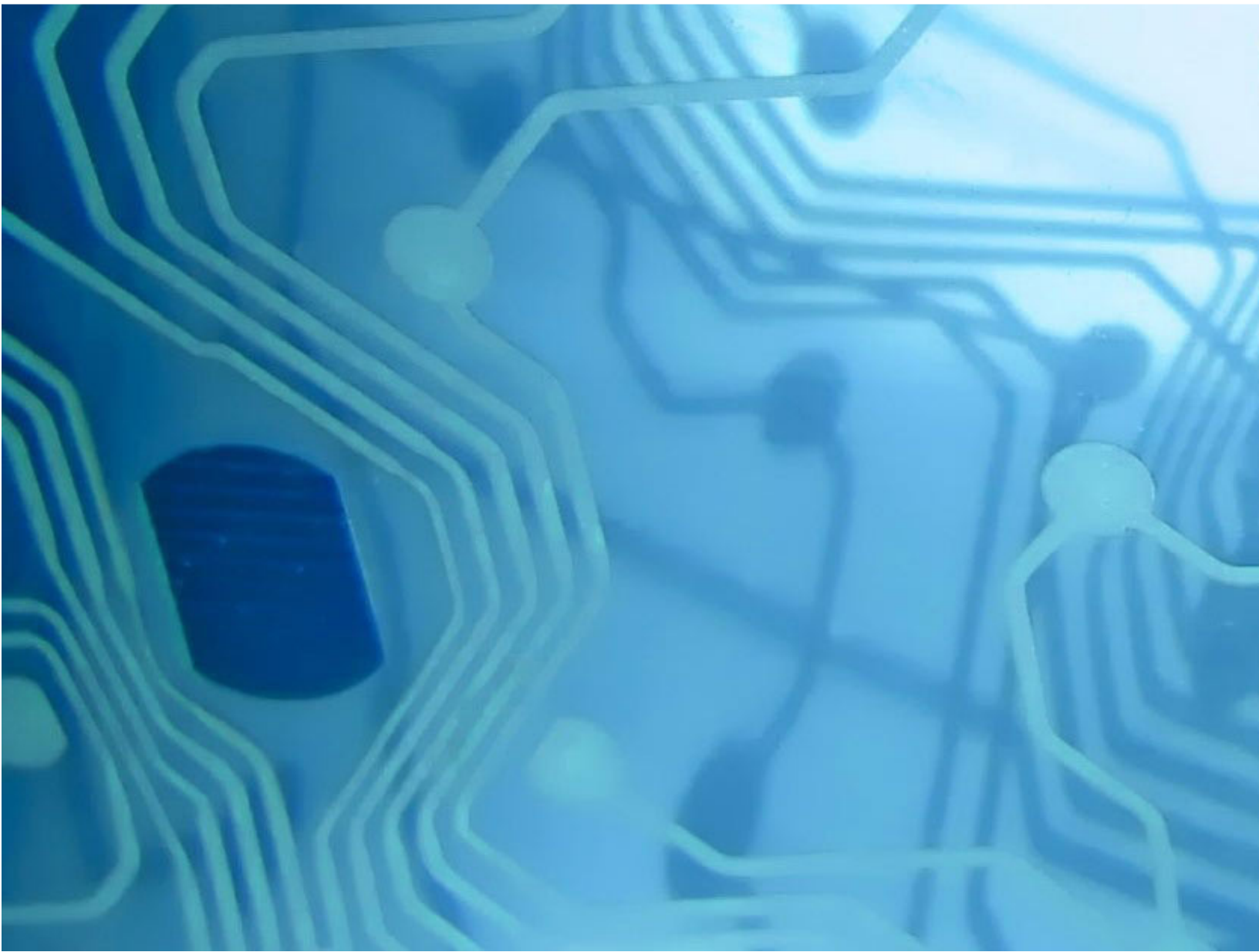
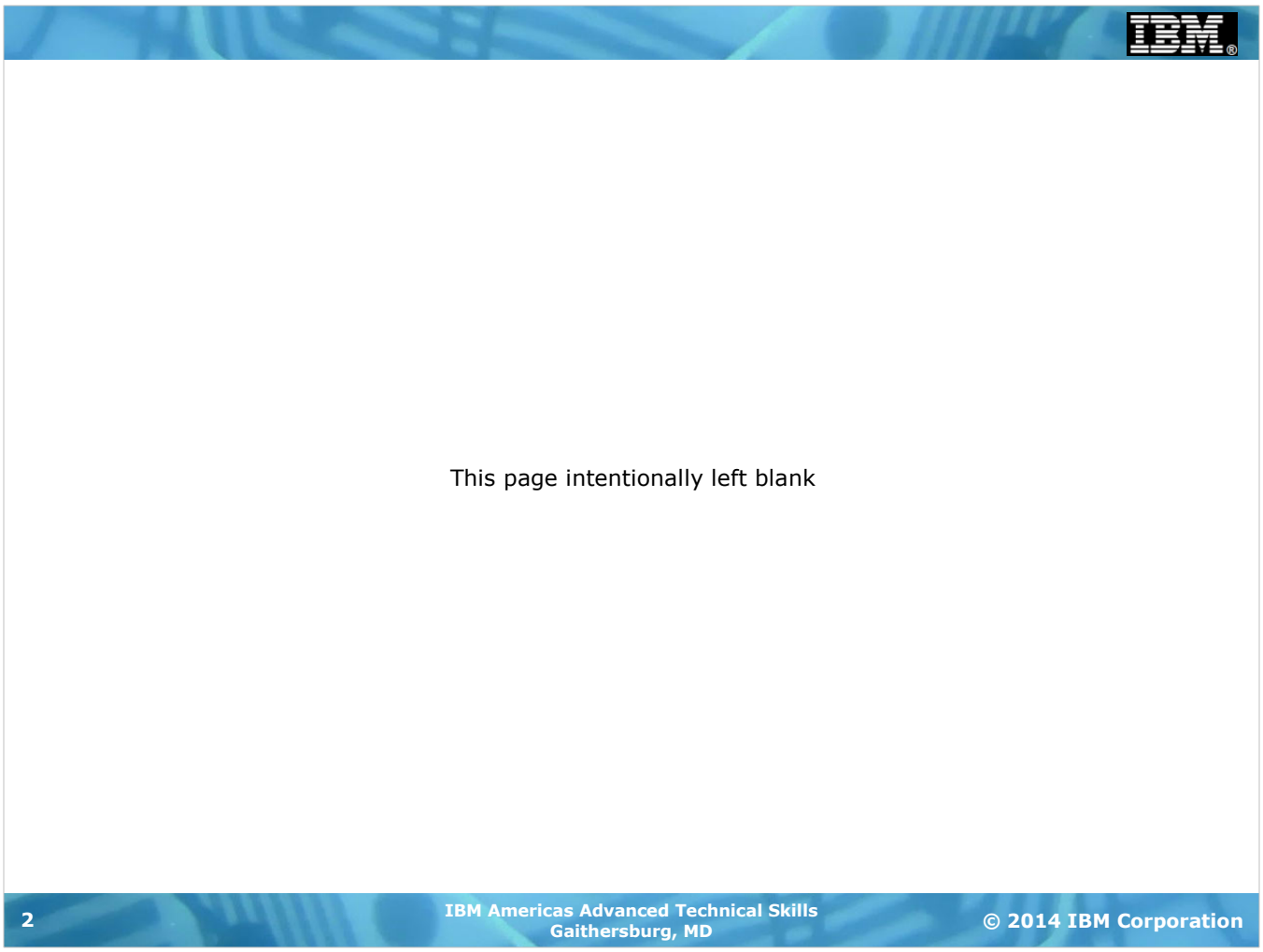
WebSphere Application Server Liberty Profile z/OS

# ZCONN1

## z/OS Connect




This page intentionally left blank
2
IBM Americas Advanced Technical Skills
Gaithersburg, MD
© 2014 IBM Corporation

# Agenda

**The agenda for this workshop is as follows:**

## Overview

**Establish context in which Liberty Profile and z/OS Connect operate**

## Liberty Profile and WOLA

**Understand the operational foundation of z/OS Connect**

**Hands-on Lab**

## z/OS Connect

**Explore the features and functions of z/OS Connect**

**Hands-on Lab**

## Security

**Explore the security considerations around z/OS Connect**

**Hands-on Lab**

**Mobile …**



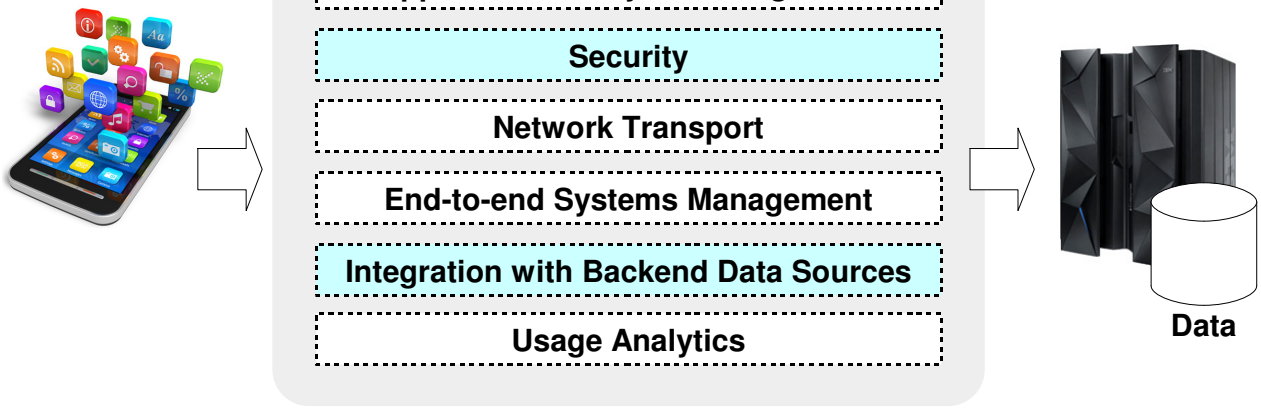

This is the agenda we'll work with in this workshop.

The central focus is on z/OS Connect, but before we get to details of that we need to establish context and provide a foundation of understanding of Liberty Profile and WOLA. Then we move to z/OS Connect.

We finish up with security because when discussing mobile computing and z/OS, the topic of security always comes up. It can't be avoided, and indeed should *not* be avoided. Understanding security is a key part of understanding mobile computing.

# 'Mobile' is a Very Large Topic Space

**The user at the smart phone sees only a very small piece of it … in between that phone and the source of data is a great deal of things going on:**




**The focus of this workshop will be primarily on the topics of integration with backend and security**

SoE, SoR …



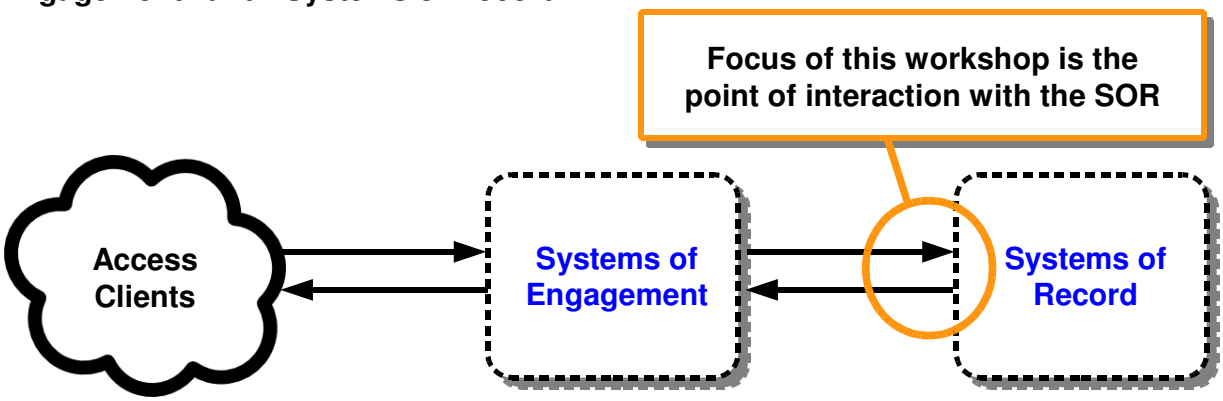

We start with a statement that is likely familiar to most in the room: the 'mobile' topic is about a lot more than just what happens on the smart phone. For the mobile device application that interacts with enterprise backend systems for data, there's a great deal that goes on:

- ***Application Development*** – tools exist for aiding in the development of the application that runs on the mobile device. This includes device-specific code libraries to look-and-feel design tools.
- ***Application Lifecycle Management*** – mobile applications are subject to update and revision just like any other. With a vast number of devices on which the apps run, managing the lifecycle (publishing updates, pushing updates. etc.) becomes very important.
- ***Security*** – no 'solution' chart since the dawn of time was without a box labeled 'security.' It's a topic that is integral to nearly all IT solution discussions, including mobile.
- ***Network Transport*** – this involves firewalls, routers and networks … everything it takes to get the mobile request from the device up into the enterprise and to the system where the data requested resides.
- ***End-to-end Systems Management*** – this involves subjects such as response time management, outage detection and reponse, capacity planning and management, etc.
- ***Integration with Backend Data Sources*** – at some point the mobile request, commonly formatted as a REST URI and carrying JSON data, needs to interface with a backend system such as CICS or IMS. There's a number of ways that can be done. One way is z/OS Connect, which is the focus of this workshop.
- ***Usage Analytics*** – once the mobile system is up and running, there will be a need to monitor usage, analyze the activity, and formulate going-forward plans based on the data.

For this workshop we will focus on the two topics highlighted – integration with the backend and security. The point of integration we'll focus on is z/OS Connect, and we'll cover the topic of security because it's a very important one in this space.

# Systems of Engagement, Systems of Record

**We start our discussion by drawing attention to the concept of "Systems of Engagement" and "Systems of Record":**




**Client systems and devices that interact with SOE**

**Systems that incorporate technologies which encourage peer interactions**

**Systems that host authoritative data sources for a given data element or piece of information**

**Not *just* mobile phones ... any system or device ... including mainframe programs**

**This can be on System z, and in fact System z can make an excellent platform for SOE**

**This doesn't have to be System z, but a great deal of SOR data is on the platform**

**Common architecture ...**




There is a set of terms used with mobile computing that should be brought into context at this point. Those terms are "System of Engagement" and "Systems of Record." The common definitions for each is shown on the chart.

We add to that the clients themselves, which is what's shown on the left side of the chart. This can be mobile devices, but it can also be another computer or even another mainframe. These clients access the "Systems of Engagement," (SoE) which serves as an intermediary to the request content required. The SoE then goes to one or perhaps several "Systems of Record" (SoR) to get the information needed to satisfy the request.

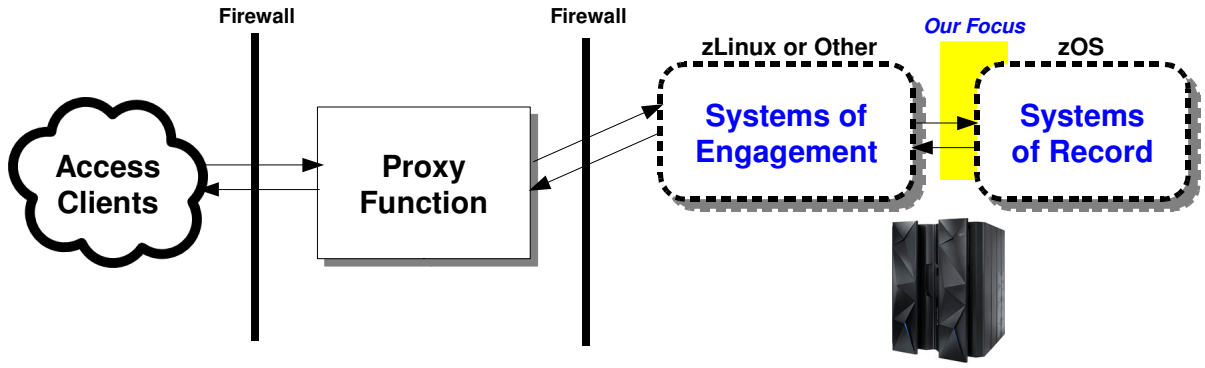
In the past the interaction was between client and SoR; for example, a person sitting at a terminal issuing a command to get the stock on hand in a warehouse. The terminal was connected to CICS, and CICS controlled the data that was used to satisfy the request.

More and more the interaction between client and data is mediated by a system that sits in the middle. Often that's because requests are satisfied by data from multiple sources. Or that's because the protocol used by the client needs to be modified before the backend data system can understand it. In any event, what we see more and more is the inclusion of "Systems of Engagement" in the architectural topology.

For this workshop our focus is going to be on the entry point to the SoR, which for this workshop will be assumed to be z/OS. It does not have to be z/OS, but because so much of the world's business data resides there we will focus there. Systems of Engagement will interact with Systems of Record to get data. What that point of interaction is and how it works is what we'll focus on.

# What We Anticipate to Be Common Architecture

**Nobody is going to allow mobile devices to access the z/OS mainframe directly. We anticipate the following to be a common architectural model:**




**The proxy function provides a secure intermediary in the DMZ**

**How identity flows back is the subject of the unit on security**

**The SOE (IBM MobileFirst Platform) will be back in the secure zone**

**In many cases the SOR will be on z/OS**

IBM MobileFirst Platform ...




This chart shows a generic topology we believe will be in common use for mobile access. There are variations to this general picture, but we believe it represents a general principle of design.

The sub-title states that “nobody is going to allow mobile devices to access the z/OS mainframe directly.” Access to the z/OS system is fairly tightly controlled in all cases, and opening it up to hundreds or millions of mobile devices just isn't going to happen. It *could* be done (technically speaking), it just won't likely happen.

**Note:** an exception to this would be tightly-controlled devices used for in-house applications. For example, rental car devices used to check cars back into inventory. Or hand-held devices used for warehouse or factory floor use cases. In those cases the number of devices and the users of those devices is controlled, and therefore access can be better controlled.

Devices outside the secure inner network are going to come through a set set of firewalls, and between those firewalls will likely be a proxy function that is the first to handle the request. That will provide an intermediary function in the DMZ to provide an additional layer of protection for the backend systems in the secure zone. How and where authentication takes place depends on the design of the system; how it flows back (and *if* it flows back) is the subject of the unit on security.

The Systems of Engagement will be hosted in the secure zone. An example of a SoE is IBM's MobileFirst Platform product, which will speak to briefly in a bit. The SoE will interpret the request and flow requests back to Systems of Record, either one or many. As stated, the focus of this workshop is the interaction with z/OS and the use of z/OS Connect.

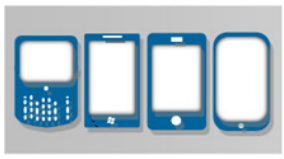
A good deal of this topology will exist on platforms other than System z. The SoE may be on System z or not. IBM MobileFirst Platform, for example, is supported on Linux for System z. But it may also run on a non-Z server platform. The SoR does not need to be z/OS, but in many cases it will be. That is because of the wealth of business data that exists on z/OS systems such as CICS, DB2, or IMS.

# IBM MobileFirst Platform

**IBM MobileFirst Platform is a suite of functions that provides development, connectivity and management for mobile applications:**

**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

**MobileFirst Platform and connectivity …**

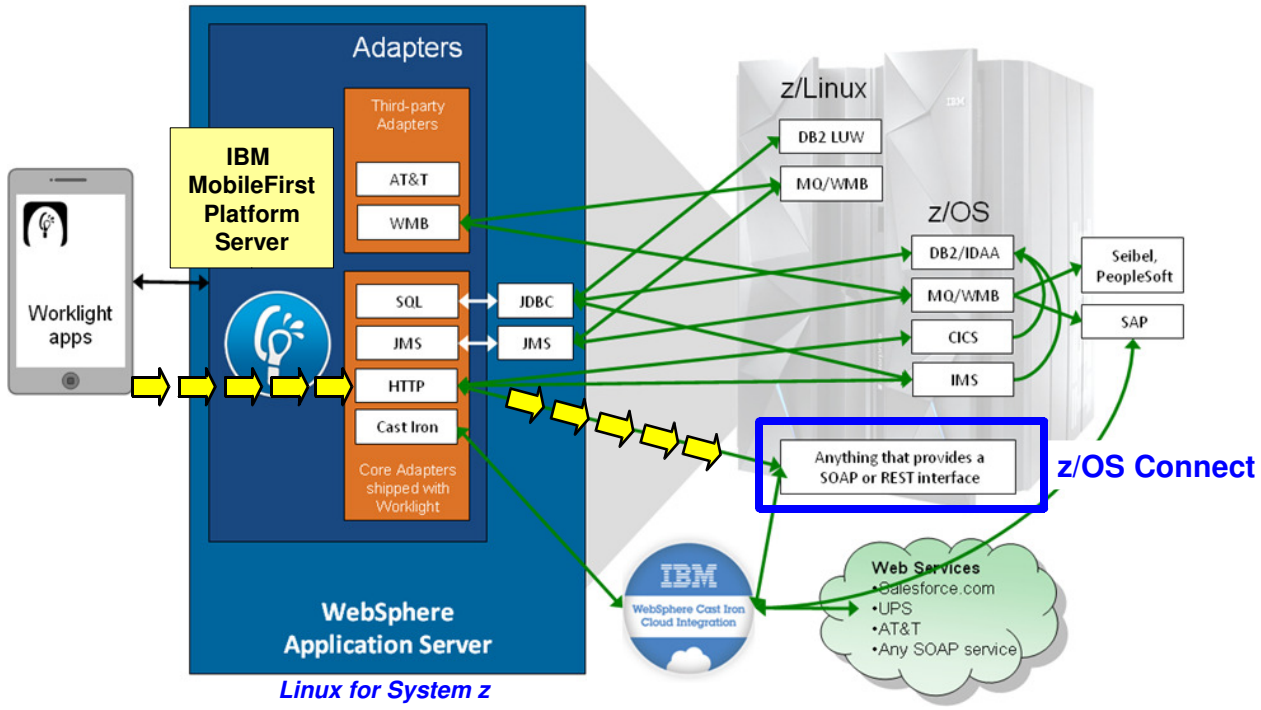



One System of Engagement is IBM's MobileFirst Platform, which is actually a suite of functions that supports mobile application development,deployment, connectivity and updates. Together this is known as a MEAP … Mobile Enterprise Application Platform. The four major categories of function provided by MobileFirst Platform is shown on the chart above.

A good deal of this is related to application development and deployment. That's important, but it's not what we're going to focus on in this workhop. Our focus is going to be on what happens after IBM MobileFirst Platform Server turns to request data from the System of Record. That's where z/OS Connect comes into the picture.

# IBM MobileFirst Platform Adapters and Connectivity

**MobileFirst Platform Server provides connectivity to backend systems via "adapters":**




**Why z/OS Connect …**

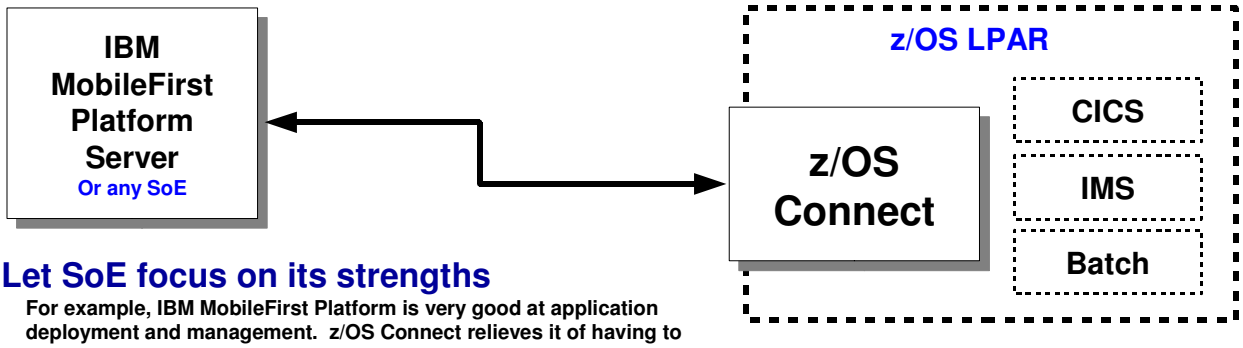



IBM MobileFirst Platform Server has the ability to connect to a wide variety of backend services, using a number of different protocols. It accomplishes this with "adapters," which are software components that know how to take the mobile device request and invoke the backend system using the required protocol.

The focus of this workshop will be on the indicated path – the HTTP connect that speaks to any device that supports REST and JSON. That's exactly what z/OS Connect is – a system that understands REST/JSON and communicates to CICS, IMS, or long-running batch behind it.

# Why z/OS Connect?

**We have not yet introduced z/OS Connect, but it's important at this point to answer the question – "Why z/OS Connect?"**




## Let SoE focus on its strengths

**For example, IBM MobileFirst Platform is very good at application deployment and management. z/OS Connect relieves it of having to do protocol and data conversion.**

## Let z/OS Connect be 'gateway' to z/OS

**It provides a single, common and consistent entry point. And yes, z/OS Connect can be duplicated for HA. This can be part of plan to expose z/OS programs as a 'service' through an API layer.**

## Manage data conversion close to source

**The target programs and their data structures are on z/OS. This allows all activities related to conversion to be kept in one place. Data conversion is Java-based and therefore off-loadable.**

## Capture usage statistics at the 'gateway'

**z/OS Connect cuts SMF records on request/response statistics**

**May not be apply in all cases, but it may make sense in some. It is an option to consider.**

**API Management and Mainframe as a Service …**




The previous chart showed a picture of IBM MobileFirst Platform connectivity options that was quite impressive. That naturally brings up the question what z/OS Connect brings to the table. This is asked before we've even introduced z/OS Connect.

z/OS Connect does not replace the functionality of IBM MobileFirst Platform or other Systems of Engagement. What z/OS Connect provides is an opportunity to leverage the strengths of those SoE platforms and offload some of the protocol and data tasks to z/OS Connect. It also provides a single "common and consistent" entry point to a z/OS LPAR for REST/JSON requests. Some like to call this a 'gateway' function.

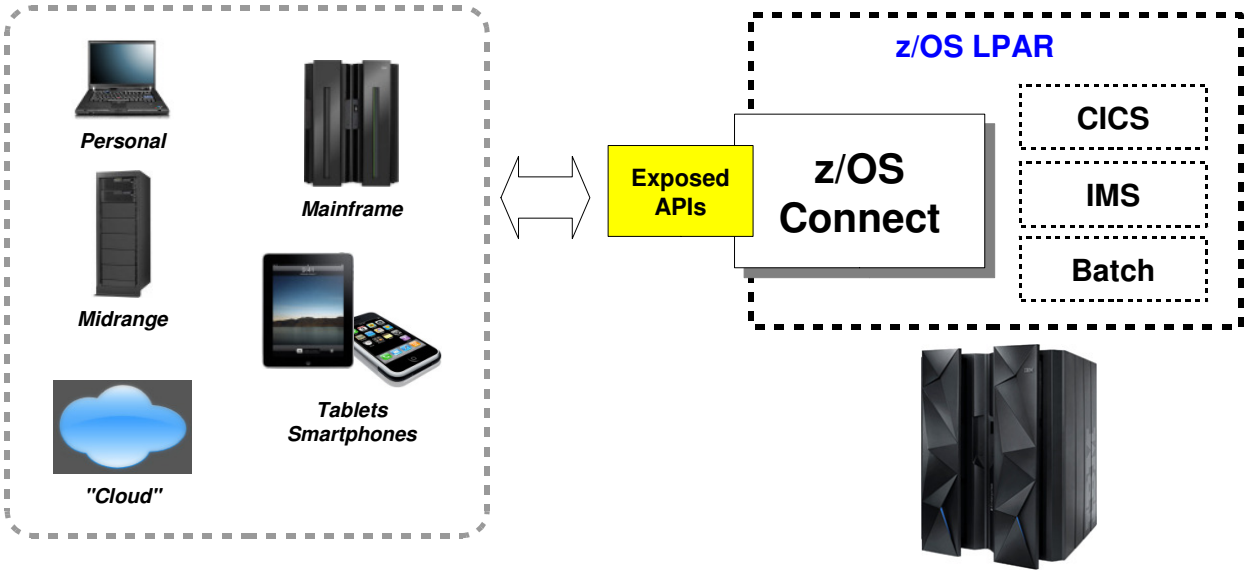
**Note:** this 'gateway' function may play into a strategy to expose z/OS programs as services through an API layer. In that case, the accessing devices may not be mobile devices at all; they may be other mainframe programs for example. That is one of the benefits of z/OS Connect – since it consumes REST/JSON (a common standard) it can expose z/OS programs as services represented by an open and discoverable API.

Further, the data conversion that z/OS Connect does can exist close (meaning: same LPAR) as the backend program data structures. Processes can be developed to update data conversion bind files every time the data structure changes, which eliminates the need to push files out to other platforms. Finally, z/OS Connect captures SMF records on usage, which can be folded into other SMF analysis routines to get a better handle on how the mainframe functions are being used.

How strong this argument is for using z/OS Connect is a function of many things. z/OS Connect is an option to consider when crafting an architectural design.

# Mainframe as a Service

**Another use-case for z/OS Connect is as a standard gateway into the z/OS LPAR to expose programs as a service:**




**z/OS Connect provides a way to do this with a single entry point (HA *is* possible) and common protocol (REST/JSON)**

REST/JSON ...




Another use-case for z/OS Connect that comes up frequently is using it to expose programs on the mainframe as services behind an API layer.

The role z/OS Connect plays in this is a configurable mechanism to expose backend programs through a common and consistent set of APIs. There are many ways this could be done – CICS can expose CICS programs through REST/JSON; IMS can do the same – but with z/OS Connect you have the opportunity to expose them together through a unified interface layer.

**Note:** the chart mentions that HA is possible. The key to keep in mind is that REST is a *stateless* protocol. We do not need to worry about affinity routing or maintaining state awareness one request to the next. Therefore, we can stand up multiple copies of z/OS Connect – either on the same LPAR or across LPARs – and *provided the configuration of z/OS Connect is properly done*, a client accessing any copy of z/OS Connect will be able to get to the same backend resources.

What do we mean by “properly done?” Imagine you had two instances of z/OS Connect running, one on each LPAR in a Sysplex. Now imagine that the configuration of LPAR “A” z/OS Connect was *completely different* from the LPAR “B” z/OS Connect. In that case a client cares very much which instance of z/OS it comes to, since the two are not the same. That is an example of an *improperly* configured HA setup for z/OS Connect.

But imagine the two copies of z/OS Connect are configured so that *both expose the same REST URI patterns,* and *both access backend CICS or IMS configured in a Sysplex-aware setup*. Now a client can be directed to either instance of z/OS Connect and get to the same program and data. That is an example of *properly* configured for HA.

# REST and JSON

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

## Representational State Transfer (REST)

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

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

The application understands what to do based on the URI

## JavaScript Object Notation (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

z/OS Connect at high-level …




Now let's briefly describe what “REST” and “JSON” are, since they're central to the z/OS Connect story.

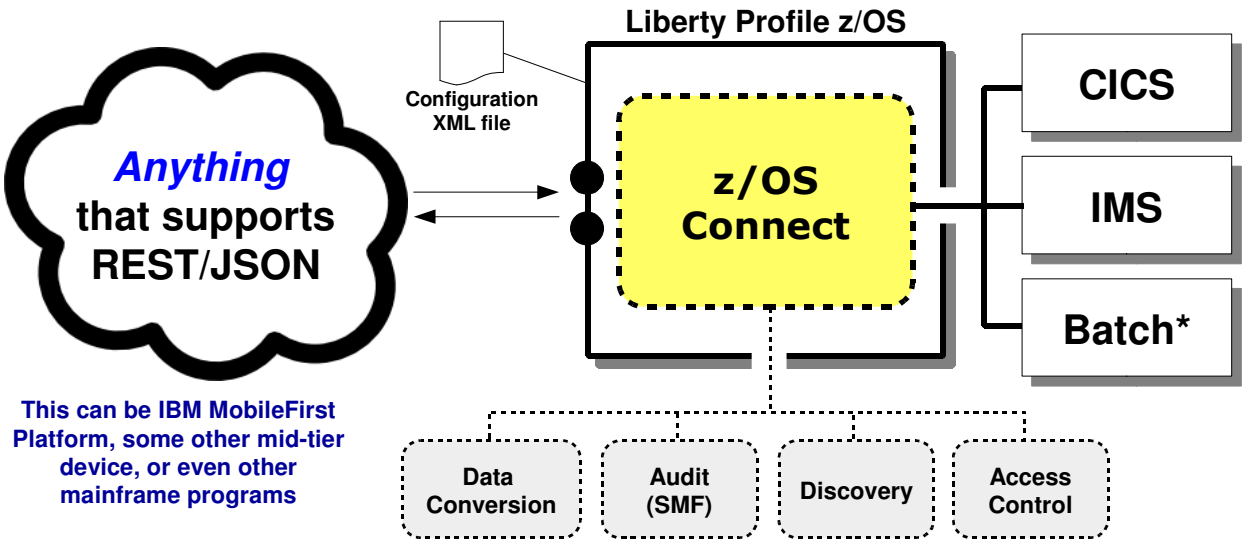
- **REST** – The acronym stands for “Representational State Transfer,” and it is a means of communicating requests via HTTP using URI patterns that the receiving application recognizes. For example, the chart shows a URI of `/account/update`. Assuming the application receiving that request understands what the URI is supposed to do, the server then invokes the program to do an account update. For updates the HTTP verb PUT or POST would most commonly be used. For requests that are designed to simply return data, then the HTTP verb GET is typically used. At its core that's what REST is – a means of requesting actions by a receiving application based on HTTP verbs (GET, PUT, POST, etc.) and URI patters the receiving application is designed to recognize and act upon.
- **JSON** – The acronym stands for “JavaScript Object Notation,” and it is a way of encapsulating data in name/value pairs that is serialized and sent back and forth on the HTTP requests and responses. JSON is enjoying popularity with web applications because it's a lighter-weight means of carrying data than XML.

As noted, z/OS Connect is designed to understand REST/JSON from clients. When z/OS Connect receives the REST and JSON, it either passes it through to the backend system or, more likely, it performs data conversion and sends to the backend the data in the format the target program there understands (for example, COPYBOOK).

**Note:** one of the themes we are presenting here is that z/OS Connect does not really care what the client is, provided it uses REST/JSON and is capable of authenticating. This is why we said earlier that z/OS Connect is not *just* about mobile devices. It can be *any* device capable of REST/JSON.

# z/OS Connect at a High Level

**z/OS Connect provides a z/OS-based solution that handles REST/JSON and connects to backend systems. It performs data conversion, auditing and provides security:**




This can be IBM MobileFirst Platform, some other mid-tier device, or even other mainframe programs

## We have an entire unit dedicated to this topic

* By "batch" we mean a long-running job that uses the WOLA "host a service" API to listen for calls coming over from z/OS Connect

Three ways to get it …




Now we come to a high-level description of what z/OS Connect is. We've hinted at it several times in the earlier discussion, but here we'll start the more direct discussion of it.

At its heart z/OS Connect is an application that runs inside a Liberty Profile z/OS server instance. The z/OS Connect application (it's a servlet) provides the framework to handle REST and JSON sent by clients, and then to invoke the backend program or service associated with the REST request. The relationship between REST request and backend service is defined in the Liberty Profile configuration XML file. That's where you indicate what URI patterns are to be recognized, and then what should be done when that URI pattern is received.

The backend programs or services that z/OS Connect is capable of communicating with is CICS, IMS and Batch. By "batch" we mean a long running COBOL (or C/C++, PLI, or High Level Assembler) program that is "listening" for a call from z/OS Connect.

In addition, z/OS Connect provides four functions within its framework – the ability to perform data conversion from JSON to whatever the backend data format is; auditing using SMF records; a set of REST APIs to provide information and details on the configured services; and the ability to configure granular access control so users have defined limited access.

There's quite a bit of detail we've left unsaid here, of course. We have a whole unit on this where we'll get into that discussion.

# Three Delivery Mechanisms

**IBM provides z/OS Connect via three mechanisms:**

## With WAS z/OS V8.5

**With WAS z/OS comes Liberty Profile z/OS. z/OS Connect is a feature of that.**

***This is the focus of this workshop***

## With CICS TS 5.2

**CICS has announced z/OS Connect as part of CICS TS 5.2. Liberty Profile will run inside the CICS region. z/OS Connect will run there and use JCICS to access CICS services. This is announced but not yet available.**

## With IMS Mobile Feature Pack

**IMS provides z/OS Connect access into IMS via a supplied instance of Liberty Profile z/OS and a JCA resource adapter to access IMS Connect.**

**It is the same function in all cases. The delivery mechanism is different, and the syntax of the configuration XML will be slightly different (for CICS JCICS and IMS Connect JCA).**

**Liberty Profile z/OS ...**




If you search for “z/OS Connect” you may be confused by references to WebSphere Application Server z/OS, CICS and IMS. It turns out IBM is providing three different delivery mechanisms for z/OS Connect, depending on which license entitlement you have.

The focus of this workshop will be on z/OS Connect as delivered with WAS z/OS. That is configured in an instance of Liberty Profile. The other two mechanisms are with either CICS or IMS.

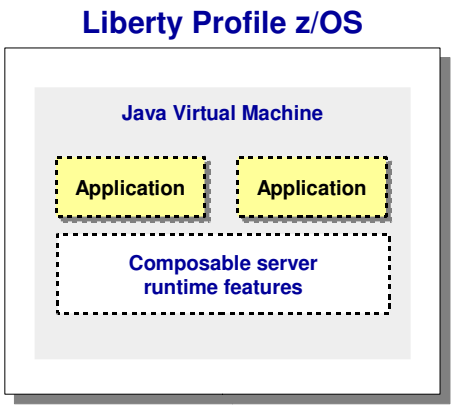
The CICS delivery is somewhat unique in that it runs inside the CICS region. CICS allows Liberty Profile to run inside CICS, and CICS is delivering z/OS Connect with Liberty. This is also somewhat unique in that it accesses the CICS region function through the JCICS interface. It can't be used to access long running programs (“batch”) or IMS.

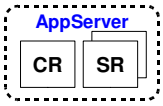
The IMS delivery comes with the IMS Mobile Feature Pack. It provides an instance of Liberty Profile z/OS and an IMS JCA resource adapter that communicates with IMS Connect to access IMS.

In all three cases z/OS Connect is the essentially the same. Some slight syntax differences in the configuration of the server.xml of the Liberty Profile, but functionally it's the same.

# Liberty Profile z/OS

**Liberty Profile is IBM's dynamic and composable server runtime. First shipped with Version 8.5, it is available on many platforms, including z/OS:**




Not this … this is the "traditional WAS" model

- **Single JVM per server model**
  As opposed to the multiple JVM model of traditional WAS z/OS (the CR/SR model)
- **Simple configuration structure**
  One XML file serves as the main configuration file
- **Dynamic**
  Changes to the configuration file or to the applications are detected and dynamically loaded
- **Composable**
  You tell Liberty Profile what features and functions you want and only that code is loaded
- **On z/OS can run from UNIX shell or as a z/OS started task**
  On z/OS we anticipate most will run as started task

**Liberty Profile is the basis for z/OS Connect, so any discussion of z/OS Connect necessarily involves Liberty**

WOLA …




On the previous chart we said that z/OS Connect is an application (a servlet) that runs inside an instance of Liberty Profile for z/OS. On this chart we'll briefly describe what Liberty Profile is, then we'll discuss it in a bit more detail when we get to the unit on Liberty.

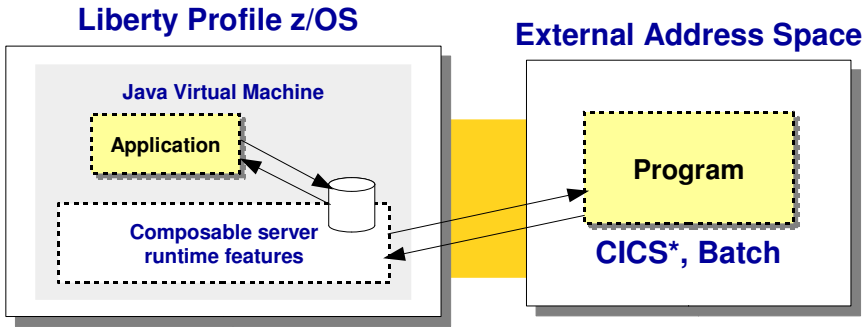
Liberty Profile is an application server that's designed to provide several key attributes:

- **Dynamic** – Liberty Profile is capable of detecting and dynamically enabling changes to its configuration, and changes to deployed applications. This is because Liberty Profile is based on the OSGi model, which has the capability of loading in changing without requiring a server restart. (Some changes do require a server restart, but most changes can be affected dynamically.)
- **Composable** – This refers to the ability of Liberty Profile to be configured to load just those functions you request, and not load everything its capable of doing. By making Liberty Profile composable, it allows you to control how large a memory footprint the server will require. For example, if you intend to use an instance of Liberty Profile for a simple JSP application, then there's no reason to load in functions like JDBC or JMS. Why take up the memory if you aren't going to use the function?

Unlike the full-function WAS z/OS server model with its multiple address spaces and Java Virtual Machines, the Liberty Profile model is a single JVM and a single address space. The server is capable of being run from the UNIX shell, or as a z/OS started task.

# WOLA is a Cross-Memory Exchange Mechanism

**WebSphere Optimized Local Adapters (WOLA) is means of communicating between WAS and external address spaces:**




**The external address space "registers" into the WAS address space. It's over that registration (logical connection) that communications flow**

**For communications "outbound" (WAS to external) a JCA resource adapter is used.**

**The external address space requires some WOLA knowledge. For CICS a set of code is provided that shields CICS programs from needing to know about WOLA.**

## WOLA is the basis for z/OS Connect communications with backend systems such as CICS or Batch

*** For IMS access a JCA resource adapter supplied with IMS is used to access IMS Connect**

**Security …**




z/OS Connect uses WebSphere Optimized Local Adapters, or WOLA, to access backend CICS or Batch programs. (IMS is accessed using a supplied JCA resource adapter that communicates with IMS Connect. For this workshop we will focus on CICS and Batch.) WOLA is a cross-memory data exchange mechanism that has relatively low per-call latency because it involves no network. Data is copied from virtual storage to virtual storage across a logical connection.

When discussing WOLA there are a few key concepts to understand:

- **Registration** – for WOLA to be used between Liberty Profile and an address space such as a CICS region or a batch program, the "outside address space" (that is, not the Liberty Profile server) needs to register in to the Liberty Profile server. This act of "registering in" creates the logical cross-memory connection between the address spaces, and is the logical "pipe" through which the data is exchanged. The outside address space is always the one that initiates registration into the Liberty Profile to which it wishes to communicate.
- **Direction of Communications** – this refers to who starts the conversation after the registration is in place. Either the Java program in Liberty Profile starts the conversation (that's "outbound"), or the outside address space initiates (that's "inbound"). The reason we mention this is because the direction of initial conversation has a bearing on the programming model. For z/OS Connect the story is somewhat simplifed because z/OS Connect is an outbound model – z/OS Connect always initiates to the outside address space.
- **WOLA Awareness** – for the outside address space to understand WOLA calls, "something" needs to be in place to handle the WOLA-level protocols. For CICS this is supplied in the form of a long-running CICS task called the "Link Server." This Link Server acts to shield the target CICS programs from requiring any WOLA awareness at all. The Link Server does an EXEC CICS LINK to the named target program, and from that perspective it's business-as-usual CICS.

We'll have more on WOLA in the next unit.

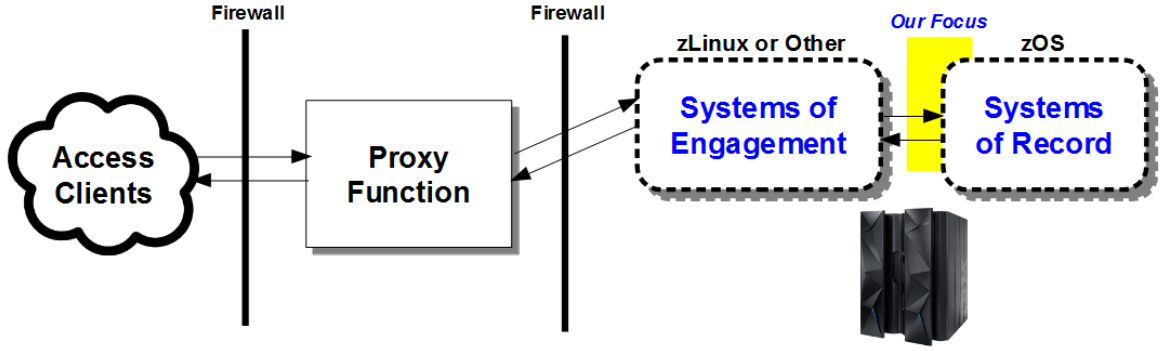
# Security Topic in Context

**The same security topics we've seen for years are present with "mobile":**

**Authentication – validating the user is who they say they are**

**Authorization – allowing the user to access only what they are allowed to access**

**Encryption – protecting network flows from being read or altered**




**How and where is each element of security provided in the architectural topology we showed earlier is the subject of the unit on security**

**Mobile Redbook …**




We have an entire unit on security because it's an extensive topic and an important topic for this subject. In the unit on security we'll cover the issues of authentication, authorization and encryption as it relates to z/OS Connect and other components in the architecture picture shown in this chart.

# Mobile Redbook

**We're going to drill down on z/OS Connect but we don't want to lose sight of the bigger System z and Mobile message ...**

`http://www.redbooks.ibm.com/redpieces/abstracts/sg248215.html?Open`

IBM Redbooks > System z >

Redbooks

## IBM System z in a Mobile World Providing Secure and Timely Mobile Access to the Mainframe

A draft IBM Redbooks publication

**Table of contents**

Part 1. Understanding the business context in a mobile world
Chapter 1. Business drivers for a mobile enterprise
Chapter 2. Introducing IBM MobileFirst for enterprise mobile solutions
Chapter 3. Bridging the gap from mobile to transactional systems
Chapter 4. IBM Worklight: the foundation for mobile solutions
Part 2. Architecting and planning a mobile solution
Chapter 5. Deployment model for a mobile solution on IBM System z
Chapter 6. The mobile enterprise architecture IBM System z
Chapter 7. Designing for resilience
Chapter 8. Designing for security
Part 3. Customer scenario

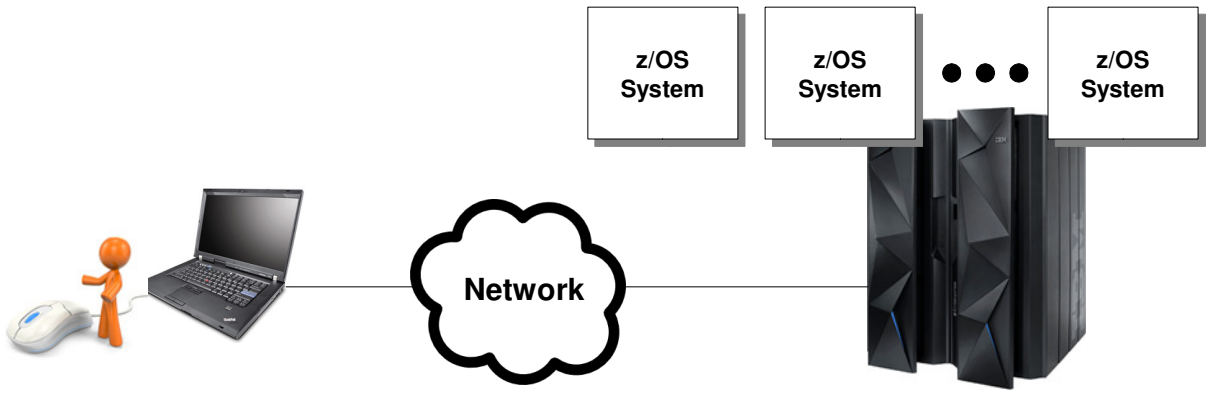
**Very much worth a look for the broader perspective on IBM's Mobile offerings and how System z fits into the picture**

**Hands-on labs ...**




As we prepare to get into the details of z/OS Connect, it's useful to keep an eye on the bigger picture. The Redbook shown above provides that bigger picture. It covers IBM's mobile strategy with respect to System z.

# Hands-on Labs




- Each lab team has their own z/OS System (identical systems except for IP address)
- Lab instructions offer step-by-step guidance
- Lab instructions are more detailed at start and less as labs go on
- Cut-and-paste file provided for commands (eliminates typing errors)





A few notes about the upcomings hands-on labs.

End of Unit