The ABCs of WAS z/OS

David Follis, IBM

Gary Picher, IBM

Michael Stephen, IBM

Don Bagwell, IBM

#SHAREorg

Sessions

Session	Title	Time	Room
17363	Debug 101-Using ISA Tools for Apps in WebSphere Application Server z/OS	Monday 11:15	Europe 11
17367	WebSphere Liberty on Windows and z/OS (Among Other Things) Hands-On Lab	Tuesday 10:00	Asia 5
17361	ABCs of WAS	Tuesday 1:45	Oceanic 7
17368	z/OS Connect: Opening up z/OS Assets to the Cloud and Mobile Worlds	Tuesday 3:15	Oceanic 7
17362	Configuring Timeouts for WebSphere Application Server on z/OS	Wednesday 8:30	Oceanic 7
17366	WebSphere Liberty and WebSphere Application Server Classic - What's New?	Wednesday 11:15	Oceanic 7
17364	IBM Installation Manager for z/OS System Programmers: Web-based Installs, Fix Packs, and How iFixes Really Work	Thursday 4:30	Oceanic 7
17365	JSR 352 - The Future of Java Batch and WebSphere Compute Grid	Friday 10:00	Oceanic 6

Agenda

- Overview of “application server”
- WAS Classic
- WebSphere Liberty WAS z/OS

Overview

In The Beginning …

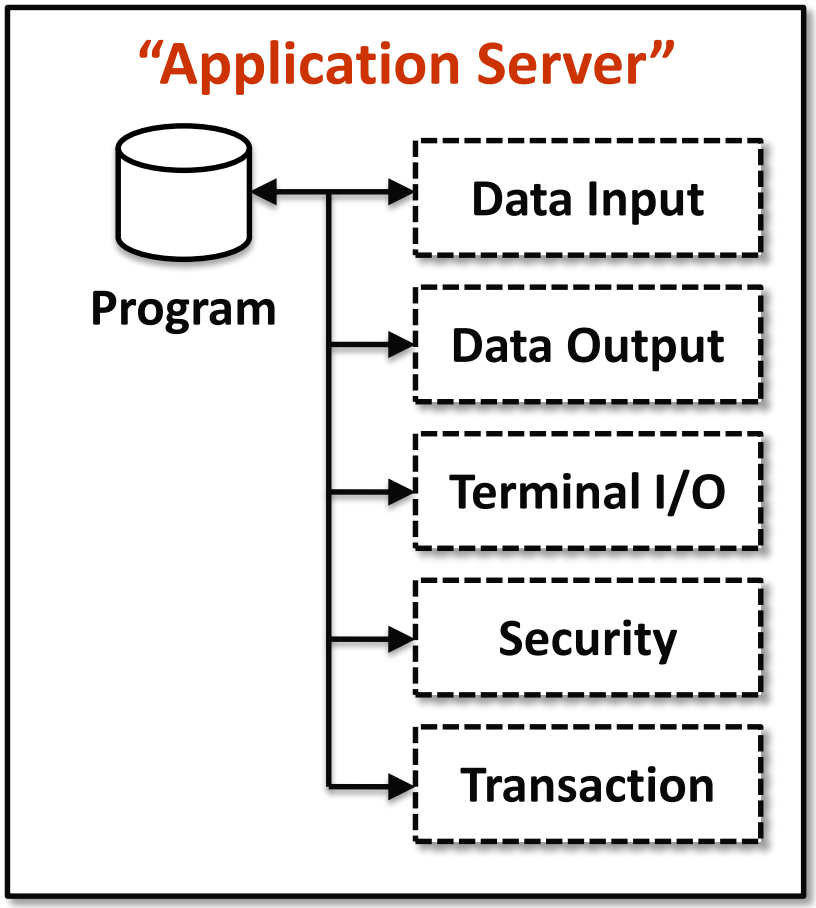
People wrote monolithic programs from the ground up

They didn't rely on pre-packaged routines or frameworks because they didn't exist

As you'd expect … they tended to reinvent the wheel a lot

But that didn't last long … eventually people started sharing code, building in sub-routines … anything to better re-use code that already existed

The Birth of the “Application Server”

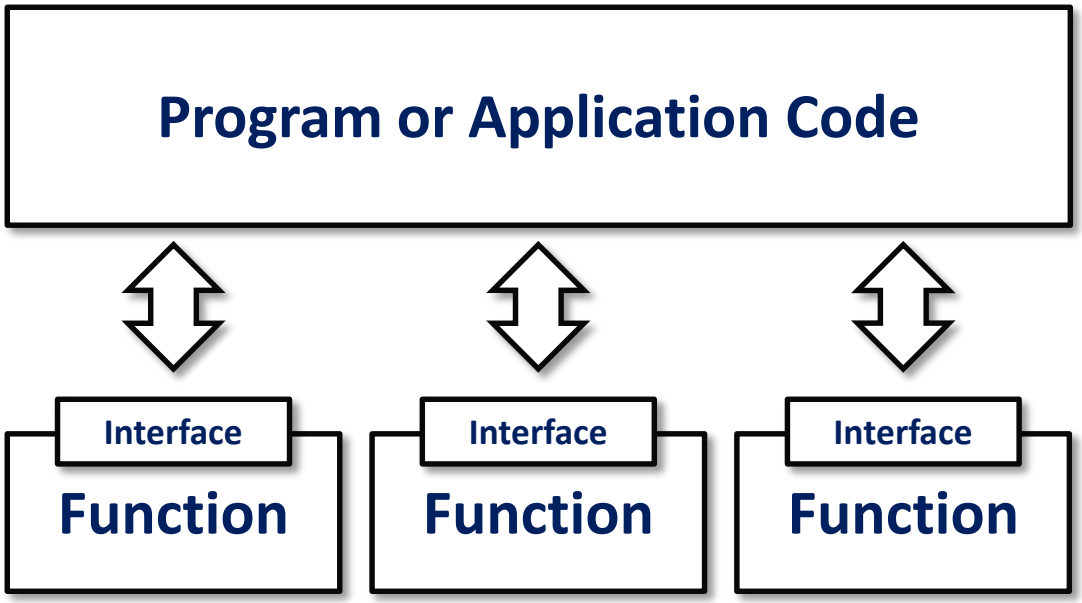


But it wasn’t called this originally

It was called:

- **CICS (1968)**
- **IMS (1968)**

The Purpose of an "Application Server"



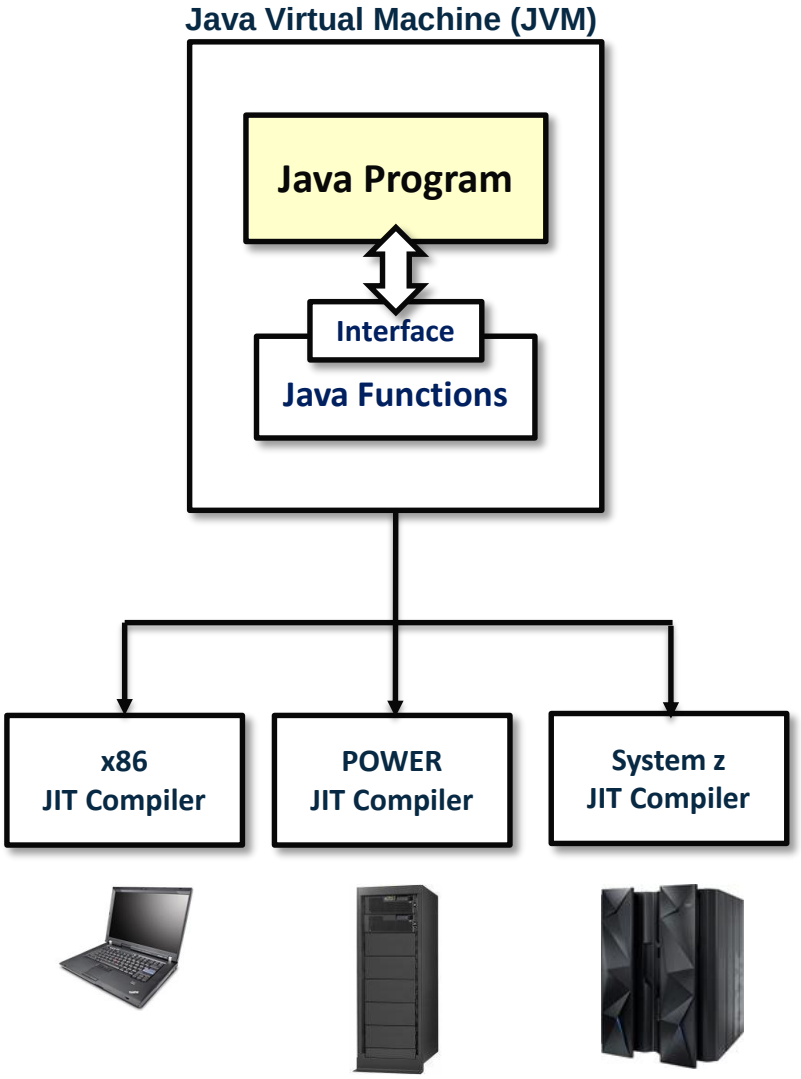
The application server provides common functions exposed through documented interfaces

Application developers focus on their business logic and call the functions as needed

By that definition, today's smart phones are "application servers" … they run apps; the apps make use of documented programming interfaces.

Our focus today is on large, multi-user enterprise appservers …

Java as an Application Runtime

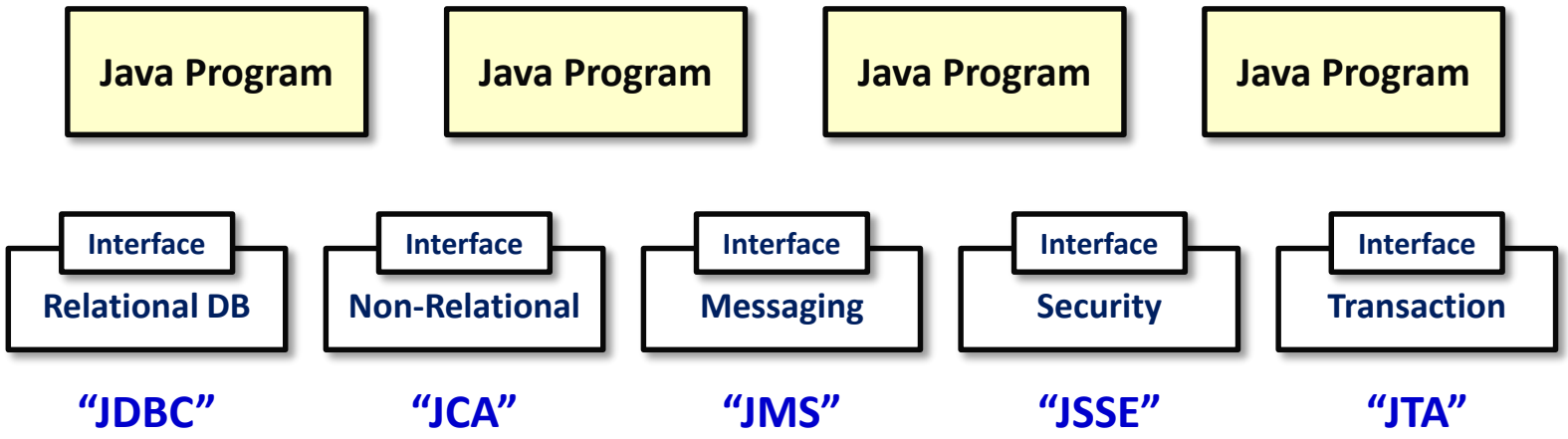


Java is designed to be platform neutral … the Java application code does not know or care about the platform on which it runs

However, the layer just under the Java Virtual Machine (JVM) does care … that's what translates the Java code to the platform code, and what compiles frequently used code into re-usable modules.

This makes Java applications portable *across platforms*

Open Standards Interfaces

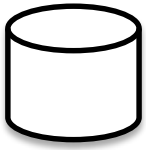


The wide-spread adoption of Java allowed the community of developers to get together and create open standard specifications for a wide array of functional services.

Open standards make applications portable *across vendors.*

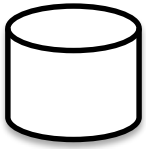
The combination of Java + Open Standards is what IBM WebSphere Application Server – all platforms, not just z/OS – is all about.

Brief Survey of Different Application Models



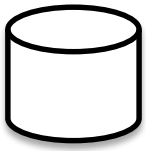
Servlet/JSP

These are often referred to as “web applications.” The most common user interface is the browser.



EJB

Enterprise Java Bean applications are often used to contain the key business logic, such as creating transactions, accessing data, and committing transactions.

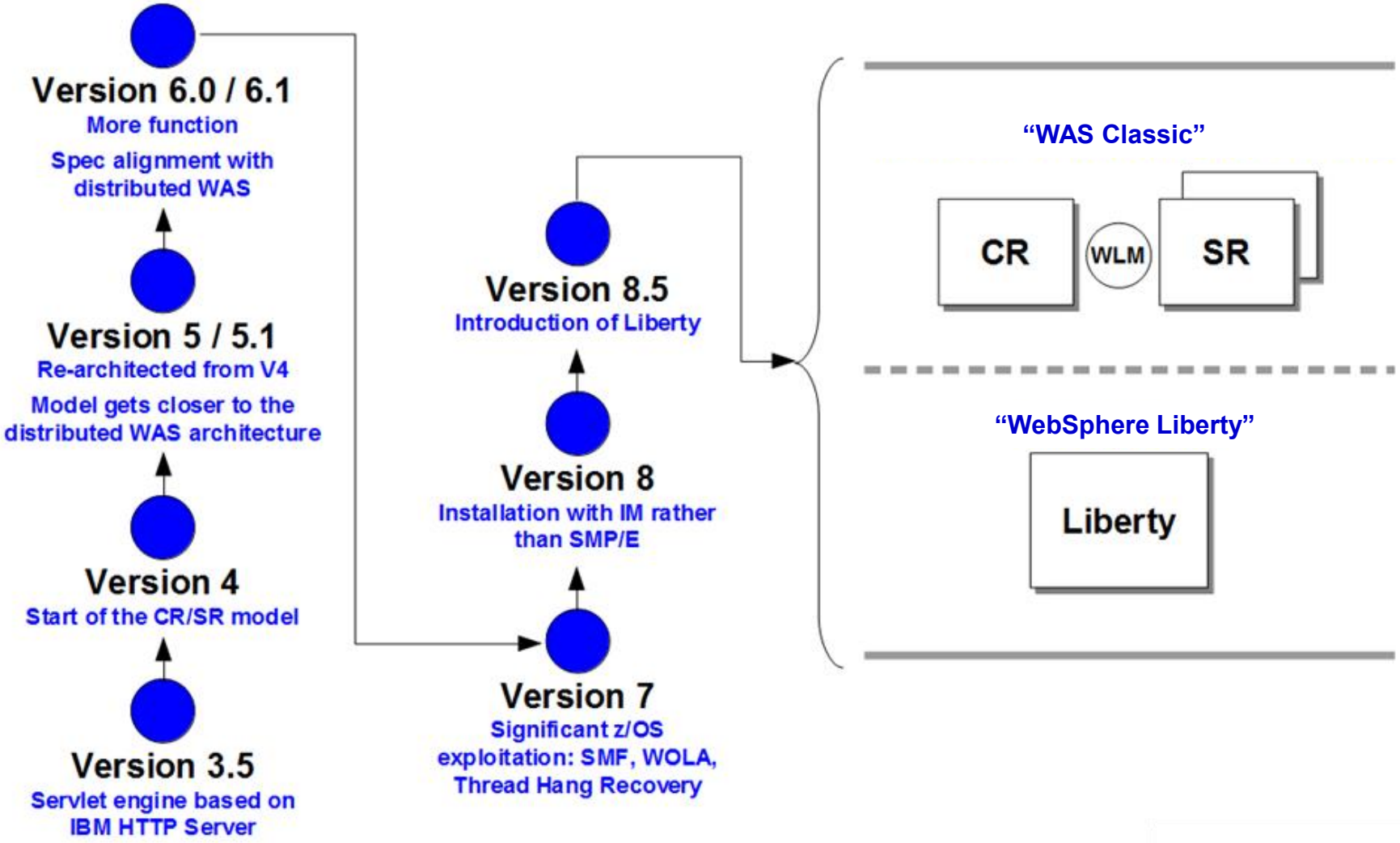


MDB

Message Driven Bean applications are really a form of EJB. They listen on a message queue, and when a message arrives they get the message and process it.

Not an exhaustive list, but it is representative of common types

Brief History of WAS z/OS



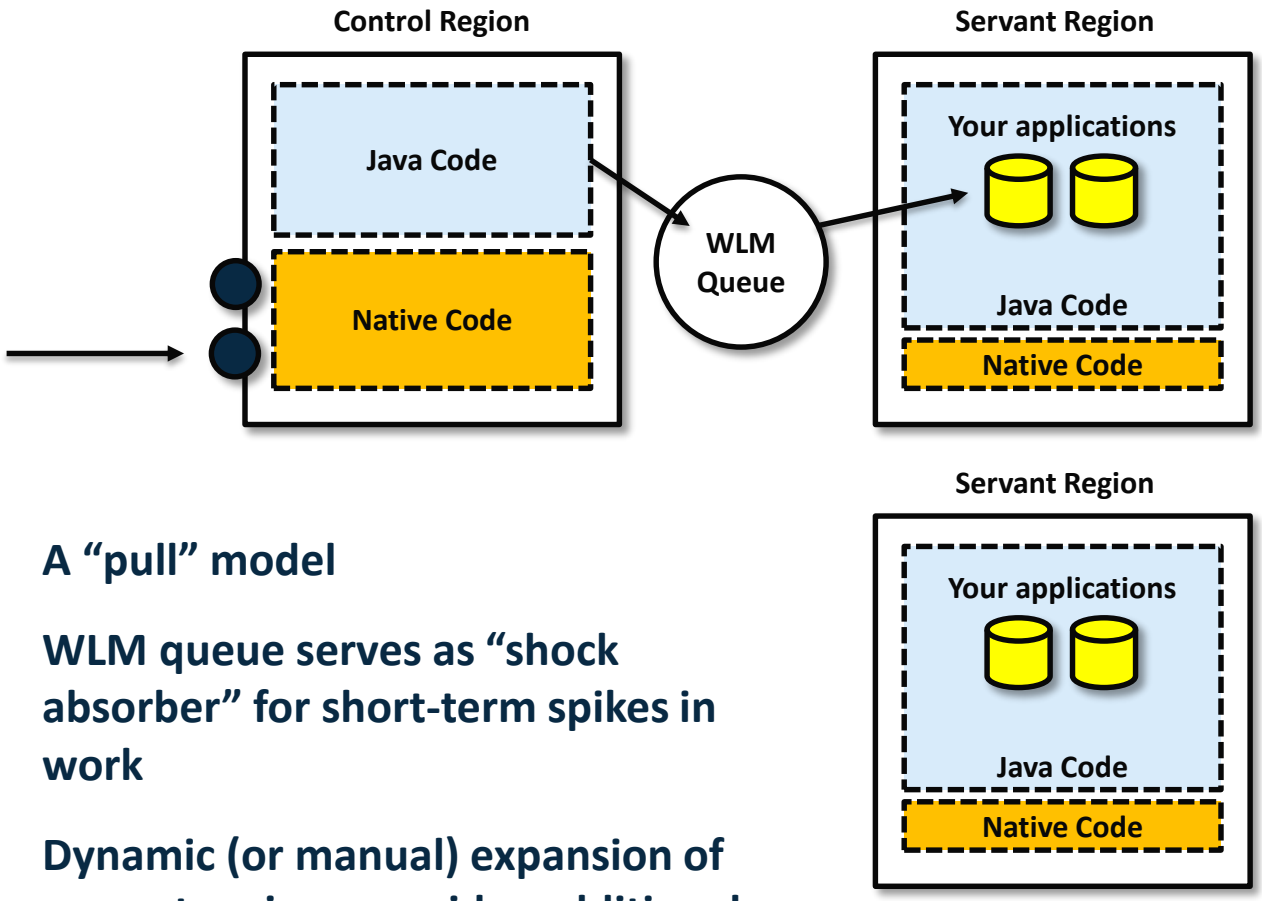
WAS z/OS Classic

Key Points …

- Most current level is 8.5.5.6
- Supports Java EE 6
- Open standard specifications supported by WebSphere Application Server is common across all platforms
- Supports Java 6 or 7 at either 31-bit or 64-bit mode
- Is in use by many large customer accounts … some with very large server topologies serving very high transaction rates
- Can be configured across LPARs for redundancy and availability

Let's take a closer look at how it operates on z/OS

The CR / SR AppServer



A "pull" model

WLM queue serves as "shock absorber" for short-term spikes in work

Dynamic (or manual) expansion of servant regions provides additional JVMs and worker threads

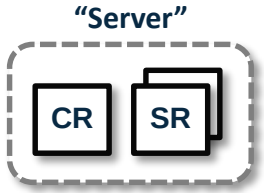
Work requests come into the CR initially

CR queues the work to the WLM queue

WLM dispatches to a worker thread in the Servant Region

May have one or more servant regions

Servant regions may be started at server start, or dynamically expanded by WLM

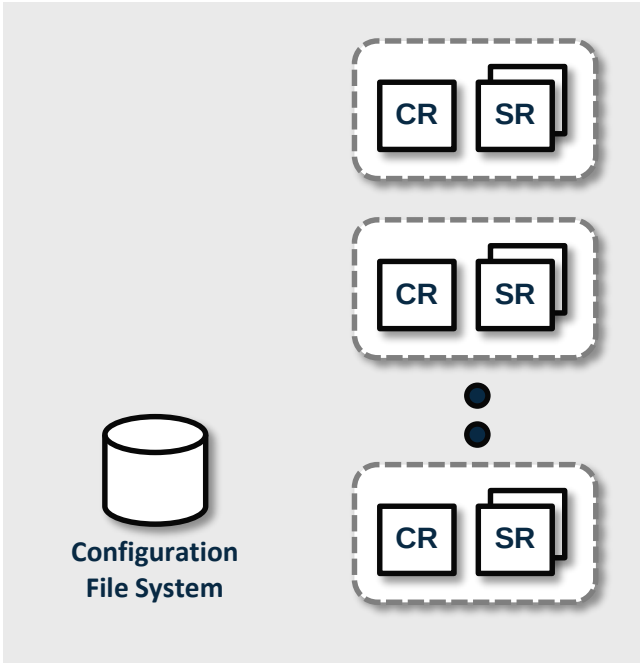


“Nodes” … Collections of Servers

You may configure many servers, depending on your needs

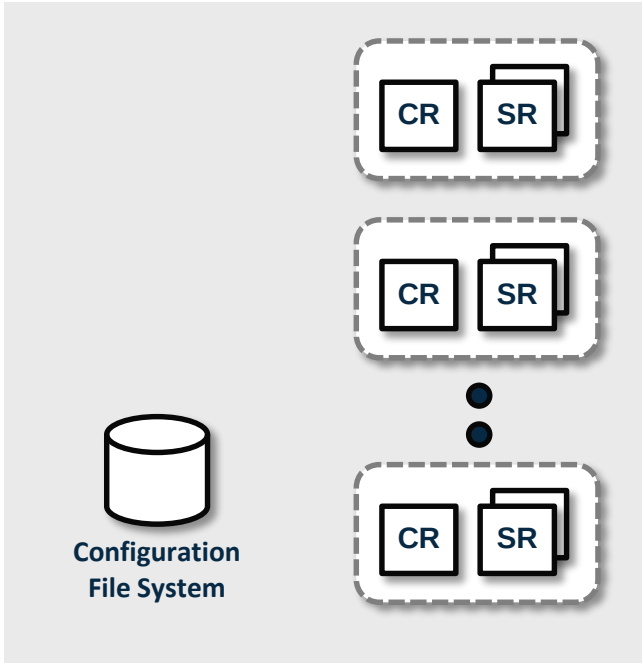
Multiple servers on an LPAR are collected into a “node”

Each node has a configuration file system associated with it



“Node”

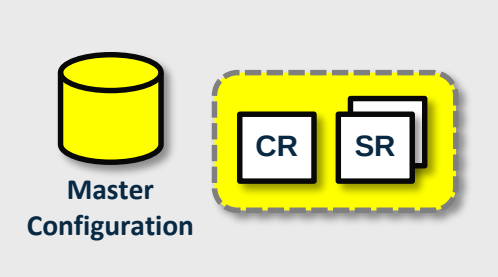
Collection of servers on an LPAR



“Node”

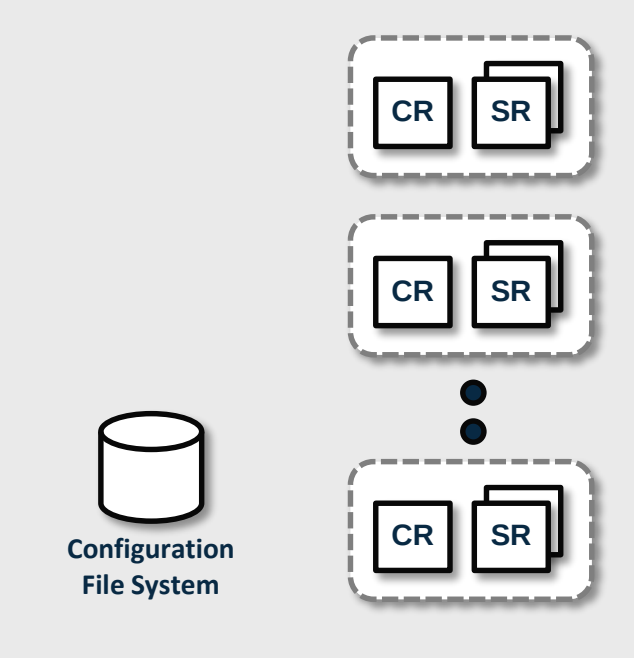
Collection of servers on an LPAR

The Deployment Manager Server



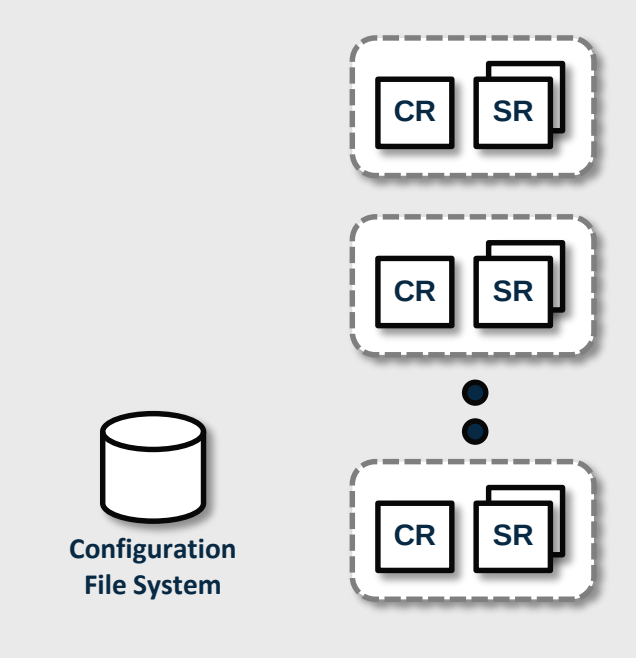
The Deployment Manager is an appserver that runs the IBM admin console

It owns the "master" configuration file system for everything it manages



"Node"

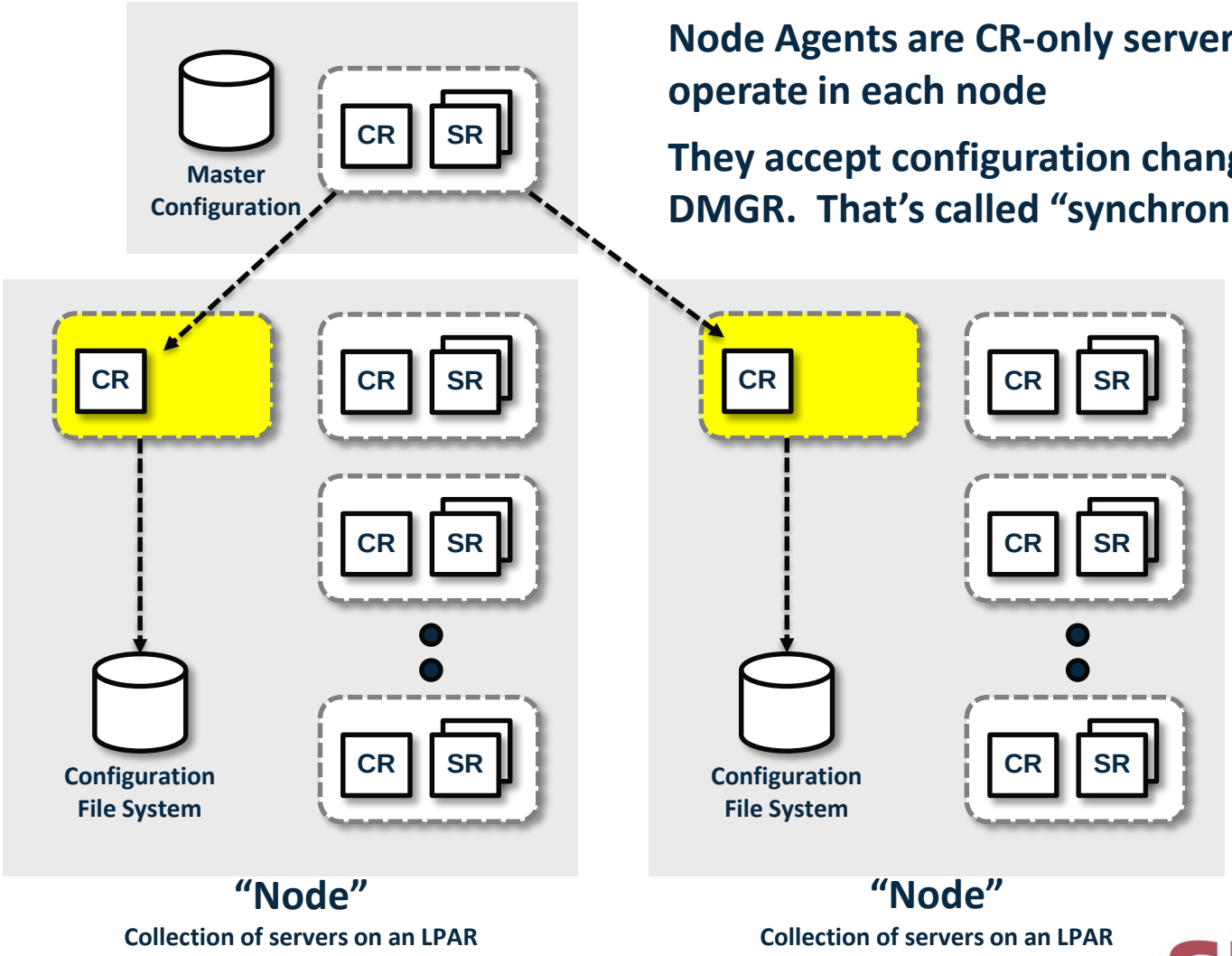
Collection of servers on an LPAR



"Node"

Collection of servers on an LPAR

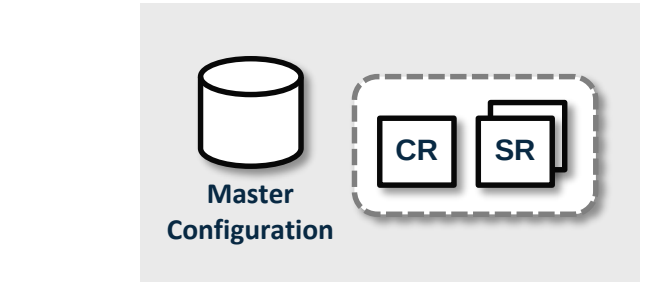
“Nodes Agents”



Node Agents are CR-only servers that operate in each node

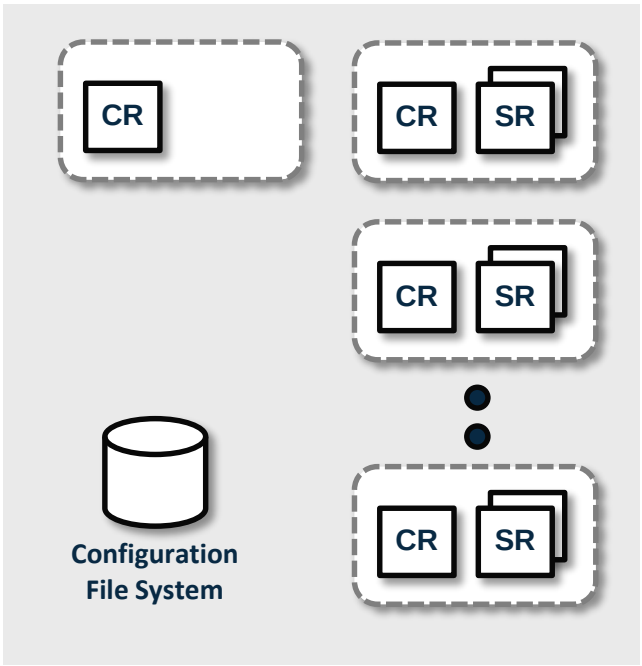
They accept configuration changes from the DMGR. That’s called “synchronization.”

“Cell” … Span of Management Control



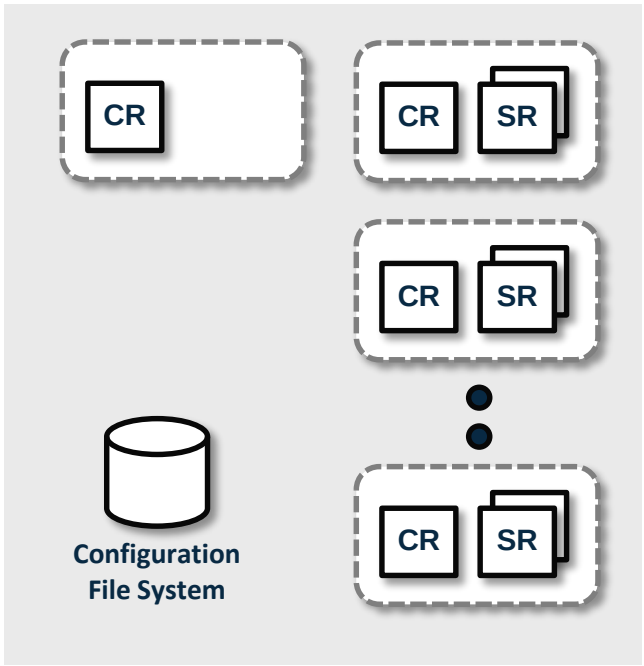
The collection of nodes managed by the DMGR is what constitutes a “cell”

The cell is often used to separate by operational purpose – Test, QA, Production



“Node”

Collection of servers on an LPAR



“Node”

Collection of servers on an LPAR

The Administrative Console

- A web-based application that runs in the Deployment Manager server
- It has sections related to the major configuration areas for WAS – servers, applications, security, etc.
- Think of this as a smart XML updater – it translates your mouse clicks and data into updates to the configuration XML files
- (You don't really want to hand-edit the XML files unless IBM Level 2 support directs you to.)
- You can also start and stop servers, deploy applications, and synchronize changes

z/OS Platform Exploitation by WAS z/OS

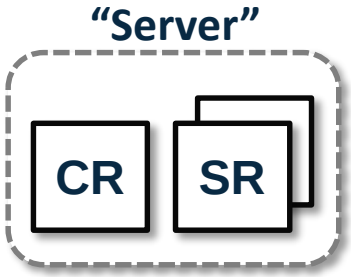
At the application layer WAS is common across all platforms. But under the application layer WAS z/OS takes advantage of the z/OS platform:

- **z/OS Workload Manager**
  - WLM is used to queue work and dispatch to servant regions
  - WLM for Service Classification (priority) or Report Classification (data collection)
- **Cross-Memory Services**
  - TCP stack bypass for call server to server on the same LPAR
  - Cross-memory into CICS (EXCI)
  - Cross-memory into DB2 (Type 2)
  - Cross-memory into MQ (BINDINGS)
  - WebSphere Optimized Local Adapters (WOLA)
- **SAF Integration**
  - SAF (RACF or equivalent) for security registry, keystores, and other security elements
- **SMF 120.9**
  - Request records to capture usage statistics for capacity planning, chargeback, analysis
- **z/OS MODIFY**
  - Dynamically change runtime, or display information about runtime

WebSphere Liberty z/OS

Background on “Why WebSphere Liberty?”

The WAS Classic design loads all the Java EE function for each server, regardless of what the applications actually needed:

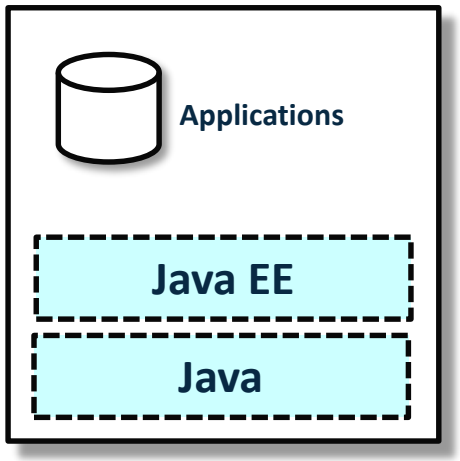


~ 1GB of memory for each server, based on default JVM heap sizes

In development and test environments, where many servers are needed, that memory requirement added up quickly.

WebSphere Liberty was created to address that issue as well as others. Let’s take a tour of Liberty … what it is, how it’s used …

High-Level View of WebSphere Liberty



Single JVM server model

- All platforms, including z/OS

Simple configuration model

- One XML file per server rather than many
- Concept of cells and nodes goes away

Composable

- You configure what features you want loaded
- The fewer features, the less memory
- You tailor Liberty to your specific needs

Dynamic

- Changes detected and dynamically loaded (configurable)
- Server changes and/or application changes
- Reduces the number of server restarts considerably

Liberty z/OS extensions

- Several platform-exploitation features (covered later)

You May Already Have it Running!

z/OS 2.1

Angel Process
(more on this later)

z/OS Management Facility
"zOSMF"

That's WebSphere Liberty z/OS!

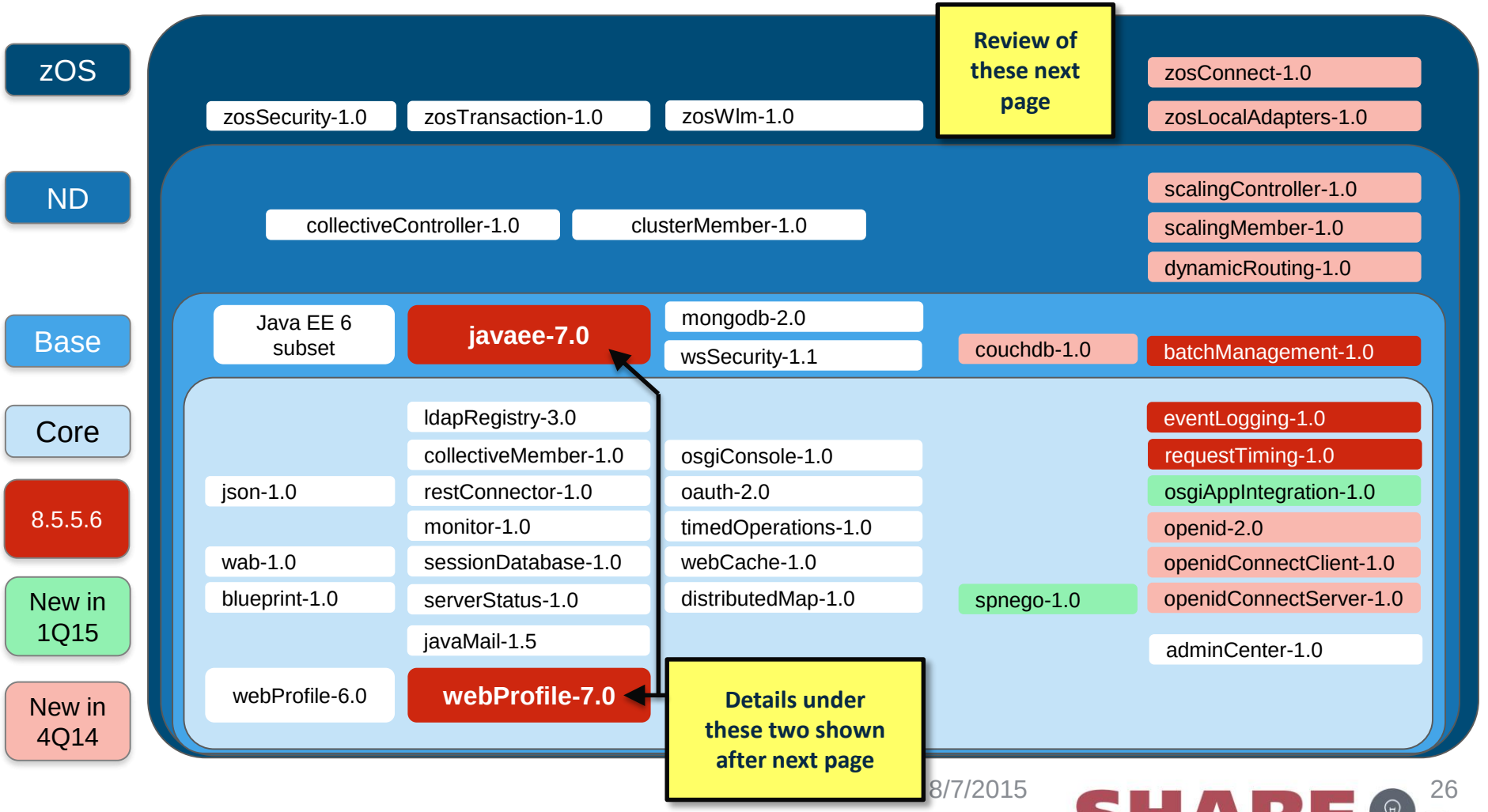
Graphical interface to z/OS system programmer functions

zOSMF is a Java program. Starting with z/OS 2.1 the Java runtime server it uses is WebSphere Liberty z/OS

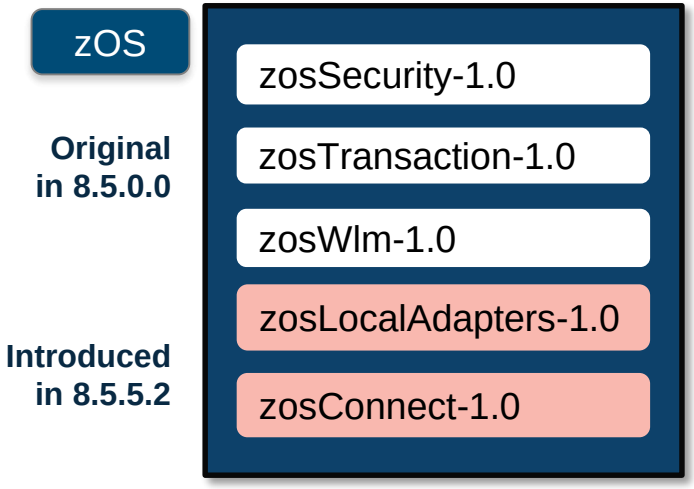
So if you have z/OS 2.1 and zOSMF enabled, you're using Liberty!

WebSphere Liberty Features

"Features" are composable units of function you configure into Liberty:



Focus: z/OS Features of Liberty



This is from the earlier chart

It shows five z/OS-specific features:

`zosSecurity-1.0` – integration with z/OS SAF

`zosTransaction` – integration with z/OS RRS

`zosWlm-1.0` – integration with z/OS WLM

`zosLocalAdapters-1.0` -- WOLA

`zosConnect-1.0` – REST/JSON request handler with integration to backend systems

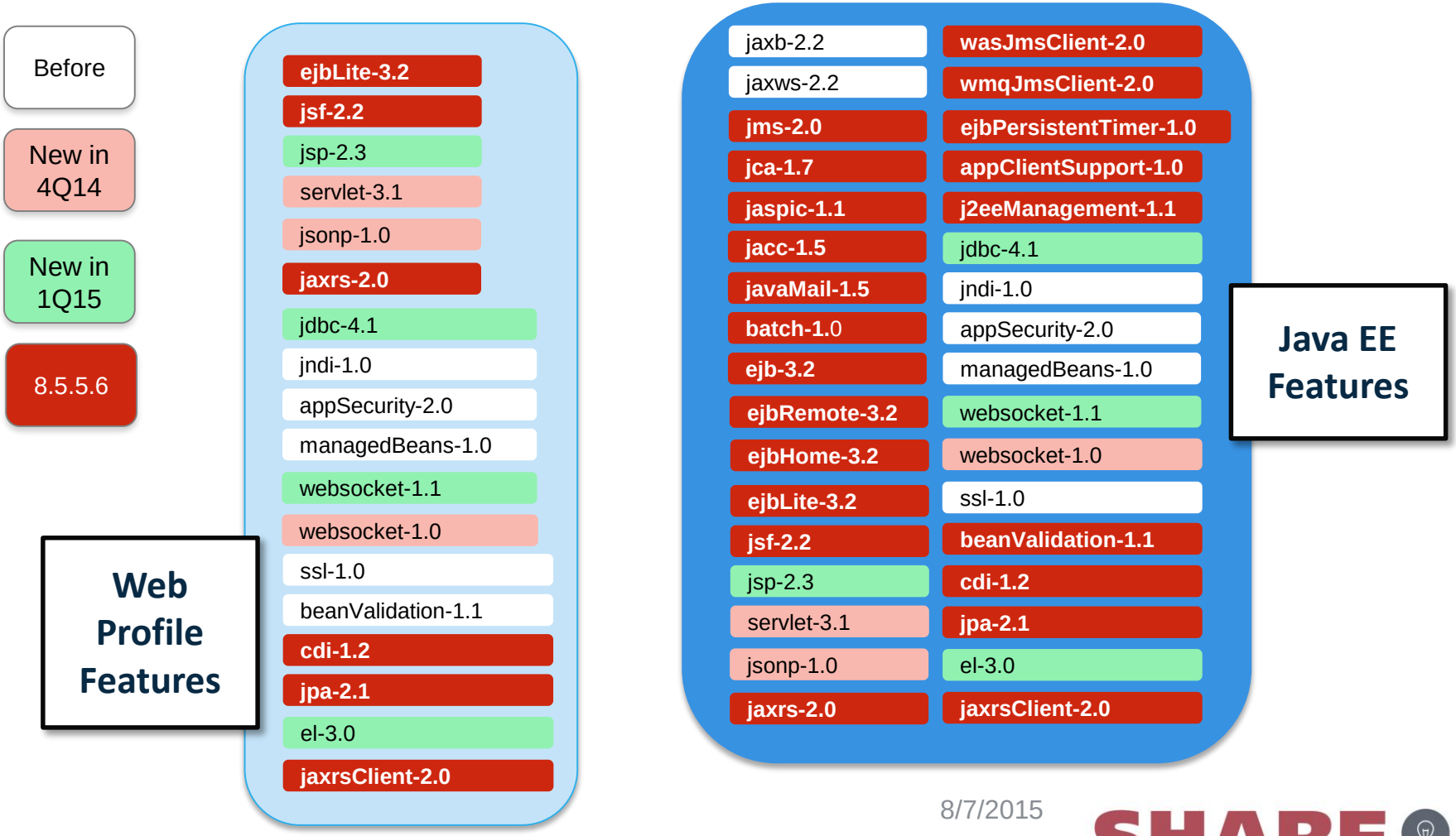
Enabling in a Liberty z/OS server is a matter of updating server.xml:

```
<featureManager>
    <feature>jsp-2.2</feature>
    <feature>zosSecurity</feature>
    <feature>zosLocalAdapters</feature>
</featureManager>
```

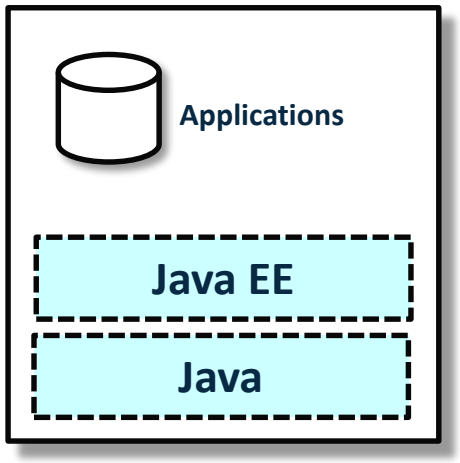
And then making the appropriate updates further down in XML to configure the function

Web Profile and Java EE Evolution

Liberty has been adding new features over time. With 8.5.5.6 it looks like this:



WebSphere Liberty on z/OS



Common features and interfaces

- Liberty z/OS has the same programming interfaces as Liberty on other platforms. Applications portable across platforms.

Start as UNIX process or z/OS STC

- Process = okay for ad hoc testing
- STC = better within z/OS operational environment
- No functional difference between the two (except MODIFY, which requires STC)

Platform Exploitation

- JDBC Type 2 (cross memory into DB2)
- JMS Bindings Mode (cross memory into MQ)
- WLM classification
- SAF integration
- MODIFY (when started as STC)
- WOLA

Creating a Liberty Server

```
/shared/zWebSphere/Liberty/V8R55FP05
└─/bin
    server
```

```
export WLP_USER_DIR=/u/libserv/liberty
export JAVA_HOME=/<path>/java_1.7_64
server create server1
server start server1
server stop server1
```

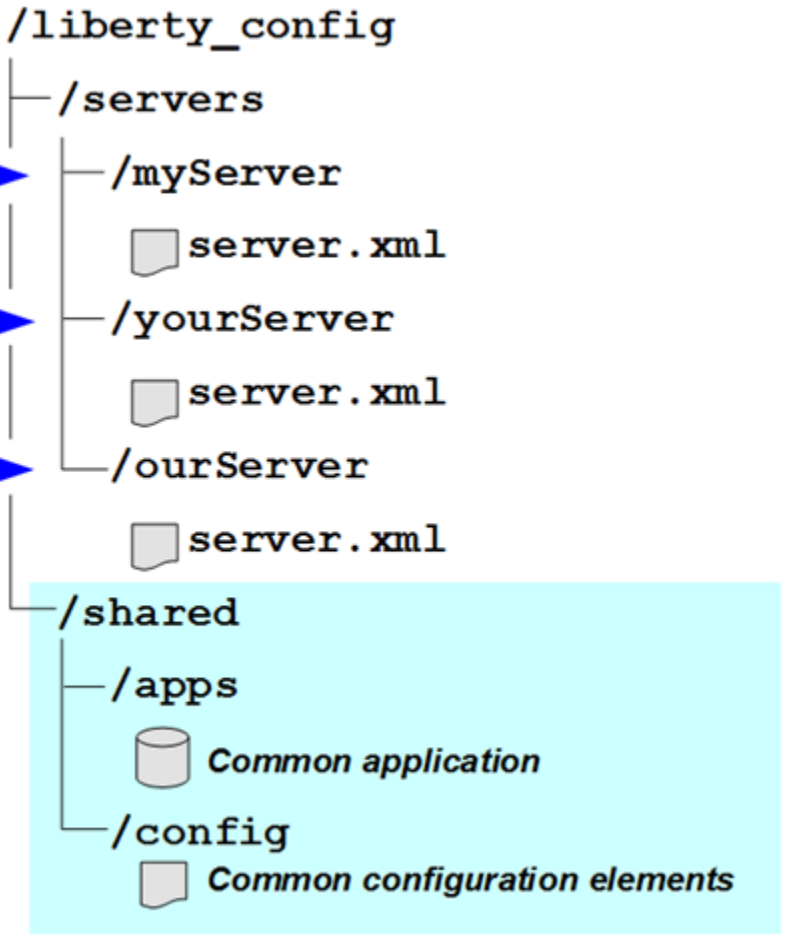
Telnet or OMVS

```
/u/libserv/liberty
└─/servers
    └─/server1
        server.xml
```

Simple but necessary step to create the server configuration structure in the "user directory" you indicate

Creating Multiple Servers

```
WLP_USER_DIR=/liberty_config
server create myServer
server create yourServer
server create ourServer
```



${shared.app.dir}

Use this variable in configuration XML to refer to the `/shared/apps` directory under the user directory where the server operates

${shared.config.dir}

Use this variable in configuration XML to refer to the `/shared/config` directory, and use `<include>` tag to bring in common XML

Starting as a z/OS Started Task

```
//BBGZSRV PROC PARMS='defaultServer'
//*-------------------------------------------------------------------
  :
//*-------------------------------------------------------------------
//  SET INSTDIR='/shared/zWebSphere/Liberty/V8R55FP05'
//  SET USERDIR='/u/user1/liberty'
//*-------------------------------------------------------------------
  :
//*-------------------------------------------------------------------
//STEP1   EXEC PGM=BPXBATSL,REGION=0M,
//  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv &PARMS'
//WLPUDIR  DD PATH='&USERDIR.'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
```

Create SAF STARTED profile so Liberty ID is assigned to this started task when BBGZSRV started

Set `INSTDIR=` and `USERDIR=` to the appropriate values for your instance of Liberty

Create a `server.env` file in same directory as `server.xml` and populate with `JAVA_HOME=` and pointer to 64-bit Java SDK

Then, from MVS Command Extension (to preserve case):

```
START BBGZSRV,PARMS='server1'
```

The `server.xml` Configuration File

```
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
    </featureManager>

    <!-- To access this server from a remote -->
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />

</server>
```

This is the `server.xml` that you'll see when you first create your server.

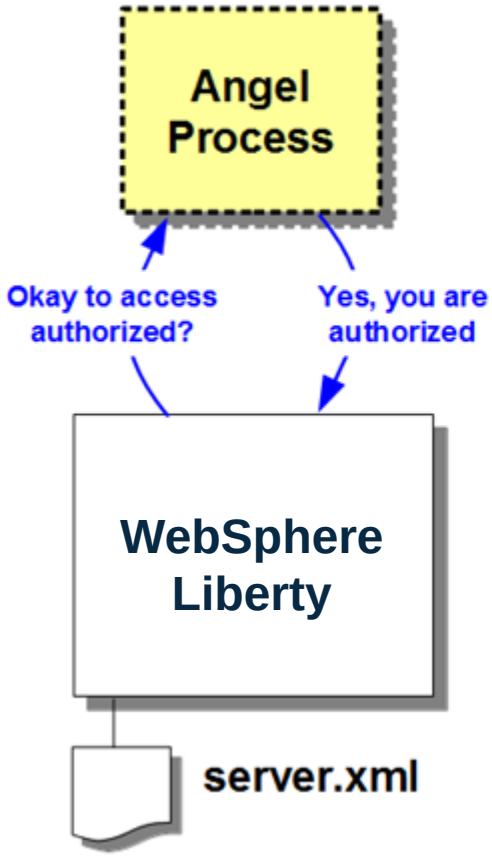
For features such as WOLA and z/OS Connect, you update this section of XML and add appropriate keywords.

You update XML here for various Liberty and z/OS Connect function ... such as using SAF, or configuring z/OS Connect services.

This is where you configure your host and ports

Configuring the server is a matter of updating the XML. Liberty will dynamically load changes when file is saved.

The “Angel” Process on z/OS



Not a 'server'

We deliberately call it a 'process' and not a server because it has no configuration, no ports, and uses no CPU once started. It's just an 'anchor point' for authorized service access.

Not required

Only required when server needs access to authorized services. WOLA is an authorized service that *requires* the Angel.

One per LPAR

When required, only one Angel is needed, regardless of the number of Liberty servers on the LPAR.

Designed to start and leave up forever

The design of the Angel is such that it should not need to be stopped and restarted. There are exceptions – move to WOLA and 8.5.5.2 required stop and restart with new level of the code.

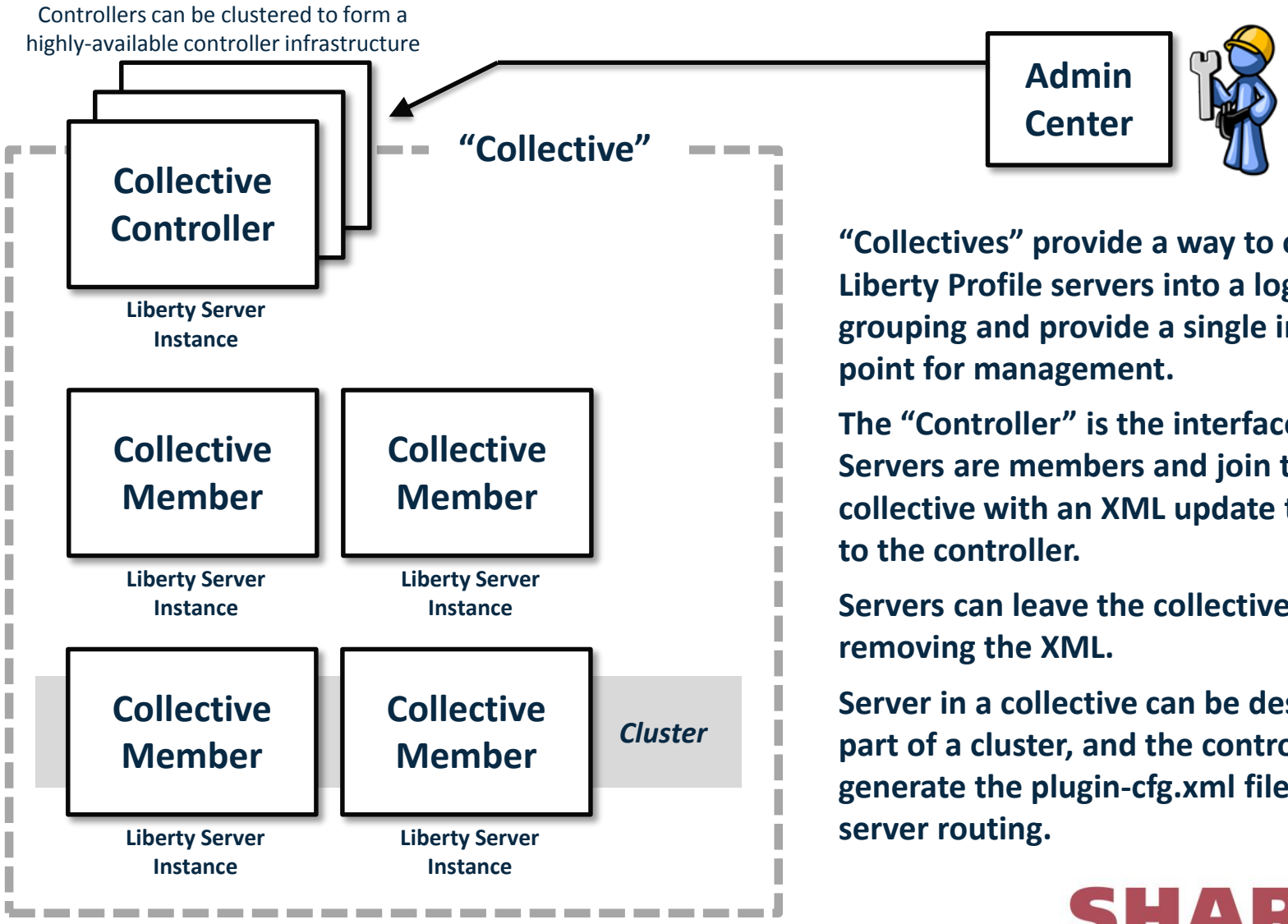
Access through SAF SERVER profiles

This is the key ... each authorized service has a SAF SERVER profile associated with it. You grant a server access to the authorized service by granting the server ID. For example:

```
BBG.AUTHMOD.BBGZSCFM.WOLA
```

Grant server ID READ to that. There's a bit more to it, but that's the basic idea.

Collectives – A Management Infrastructure



"Collectives" provide a way to organize Liberty Profile servers into a logical grouping and provide a single interface point for management.

The "Controller" is the interface point. Servers are members and join the collective with an XML update that points to the controller.

Servers can leave the collective by simply removing the XML.

Server in a collective can be designated as part of a cluster, and the controller can generate the plugin-cfg.xml file for HTTP server routing.

Using collectives is optional. When first starting out it's better to have standalone Liberty servers. It's simpler. Collectives later.

For Further Reading and Study …

WebSphere Application Server 8.5.5 Knowledge Center

www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/welcome_nd.html

IBM Techdocs

"Why WebSphere Application Server z/OS"
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101532

"WAS z/OS Wildfire Workshop Material"
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS4848

"Liberty Profile for z/OS"
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102110

"WebSphere Optimized Local Adapters (WOLA)"
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490

"Hidden Gems"
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101138
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101464
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101992
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102371

IBM developerWorks

https://developer.ibm.com/wasdev/

Questions?

We just went through a lot of material

Nobody can become expert in all this with one presentation

IBM Knowledge Center has a wealth of information on this subject

IBM Techdocs does as well