#SHAREorg

SHARE

Getting Started With WebSphere Liberty Profile on z/OS

David Follis
IBM

August 12, 2013
Session Number 13597

SHARE in Boston

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

CICS*
DB2*
GDPS*
Geographically Dispersed Parallel Sysplex
HiperSockets
IBM*
IBM eServer
IBM logo*
IMS
On Demand Business logo
Parallel Sysplex*
RACF*
System z9
WebSphere*
z/OS
zSeries*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Oracle.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
MIB is a trademark of MIB Group Inc.
* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Disclaimer

- The information contained in this documentation is provided for informational purposes only. While efforts were many to verify the completeness and accuracy of the information contained in this document, it is provided “as is” without warranty of any kind, express or implied.
- This information is based on IBM’s current product plans and strategy, which are subject to change without notice. IBM will not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation.
- Nothing contained in this documentation is intended to, nor shall have the effect of , creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of the IBM software.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

WebSphere Application Server on z/OS Sessions

Day	Time	Room	#	Title	Speaker
Monday	9:30	203	13597	Getting Started with WebSphere Liberty Profile on z/OS	David Follis
Monday	4:30	203	13600	Managing Server Output from WAS on z/OS	Mike Loos
Tuesday	9:30	203	13644	Using WAS Optimized Local Adapters (WOLA) to migrate your COBOL to zAAP-able Java	Jim Mulvey
Tuesday	11:00	203	13640	Need A Support Assistant? Check Out IBM's! (ISA)	Mike Stephen
Tuesday	3:00	203	13641	zWAS: In Real Life	Rod Feak
Wed.	1:30	202	13601	Lab: WebSphere Liberty Profile on z/OS	everybody
Thursday	11:00	203	13598	Getting Started With Compute Grid	John Hutchinson
Thursday	3:00	203	13645	Configuring Security for Liberty	Mike Loos

Agenda

- Quick Overview of Liberty Profile
- Installation
- Creating the First Server Instance
- Deploying Applications
- Multiple Server Instances and `server.xml`
- Liberty Profile as z/OS Started Task
- z/OS Extensions and the Angel Process
- z/OS MF!

WP102110 - WebSphere Liberty Profile for z/OS

Liberty Profile z/OS Overview

The new Liberty Profile provides a server model that is:

- **Composable** -- the function is very modular and flexibly decoupled, allowing you to specify just what function you need for the applications you are serving.
- **Lightweight** -- the Liberty Profile uses a number of approaches to optimize the loading of functions, whi results in a footprint significantly less than traditiona WebSphere Application Server.
- **Dynamic** -- many of the runtime functions may be dynamically updated by simply changing the configuration XML; applications may be added or updated by simply replacing the application file in th file system directory.
- **Fast** -- due to the composable design and other fac the Liberty Profile is able to execute very quickly.

Liberty Profile z/OS Technical Executive Flyer

The following is a two-page color technical executive flyer that provides an overview of the new Liberty Profile:

Liberty Profile z/OS Quick Start Guide

The following is a step-by-step "Quick Start" guide to assist with establishing early success with the Liberty Profile. The document takes the reader from some very simple initial uses of the Liberty Profile up through more sophisticated uses involving z/OS-exclusive functions such as using z/OS for digital certificate keystore/truststore, classifying work using WLM and JDBC using Type 2 and RRS:

The following ZIP file contains two sample applications referenced by the Quick Start guide:

Step by step guide to creating and using Liberty Profile

Includes focus on running as z/OS started task

Focus on z/OS extensions to Liberty

Overview

What is the Liberty profile?

A lightweight, dynamic, composable runtime

Lightweight

- Server install is only about 55 MB
- Extremely fast server starts – typically well under 5 seconds

Dynamic

- Available features are user selected and can change at runtime
- Restarts are not required for server configuration changes

Composable

- Features are implemented as loosely coupled components with lazily resolved optional and mandatory dependencies
- The availability of features and components determines what Liberty *can* do and what's available to applications

What is the Liberty profile?

An easy to configure runtime environment

- Simple, extensible, and sparse configuration model
  - Configuration can live in a single XML document
  - Configuration is by exception
    - Defaults are provided by contributing feature
    - Only modifications to the defaults are required
- Flexible configuration structure
  - Include mechanism allows for shared configuration elements
  - Variable indirection mechanism allows for customization when distributed across multiple JVMs
  - Easily managed by version control systems if desired

What is the Liberty profile?

A transportable runtime for your applications

Use “server package” to generate an archive that contains a tested, self-contained, pre-configured server instance that includes your application
- Enables an application-centric deployment model that allows for easy scale-out
- Light-touch admin builds on the ND job manager infrastructure to manage Liberty server instances

A runtime environment with fidelity to full WAS

- Liberty *is* WebSphere
- Applications that are developed and tested on Liberty will run on the full profile

Why Liberty on z/OS?

- ***Simplification***
- Liberty environments don't need significant z/OS configuration and customization
  - RRS, WLM, and SAF exploitation and configuration is optional
  - No authorized code is ***required*** to host applications
- Liberty runs in a single process instead of 3+ started tasks
  - Significantly reduced resource consumption
  - No started task definitions are ***required***
  - No need to create users and groups for controllers, servants
- Server instances can be quickly created or cloned
  - `server create serverName [options]`
  - `server package serverName [options]`

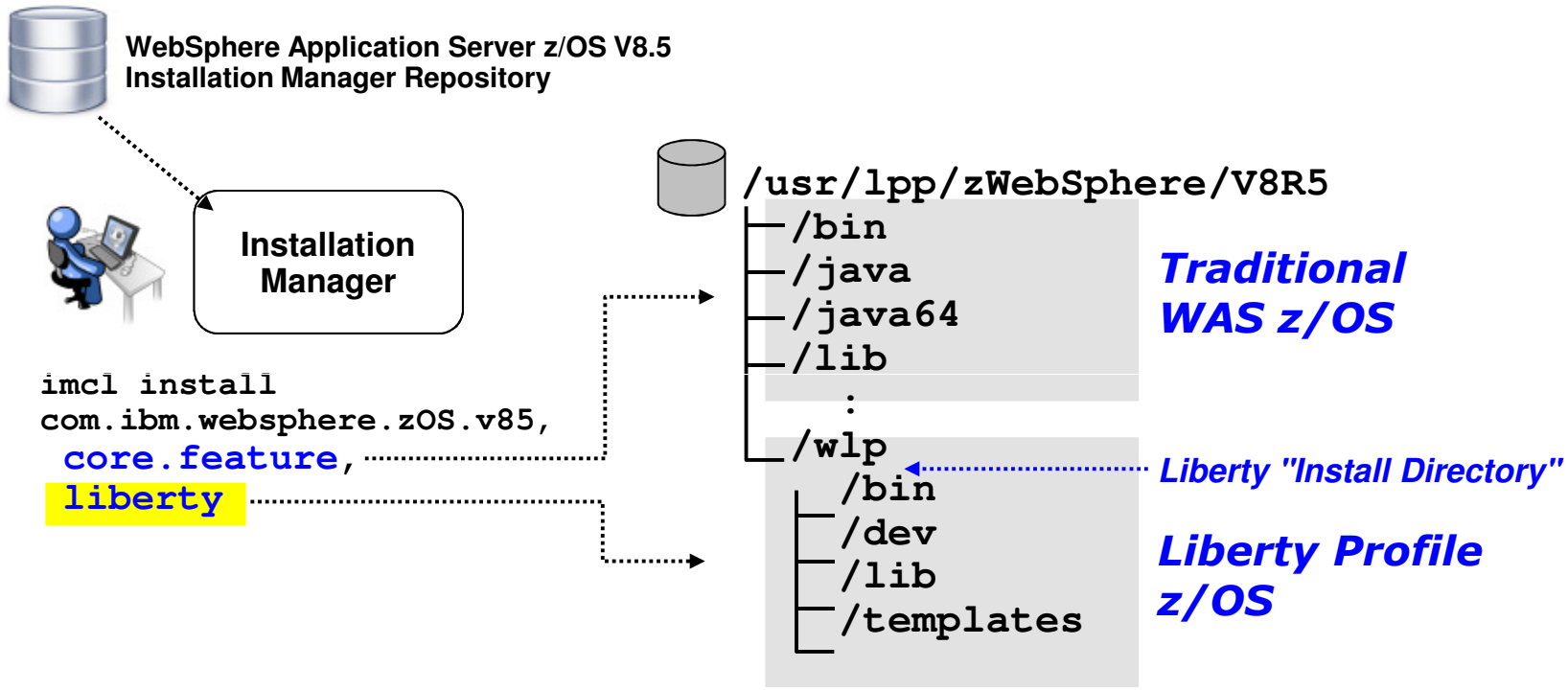
Why Liberty on z/OS?

- ***Application portability and stack consistency***
- Liberty behaves *exactly the same* on all platforms out of the box
  - z/OS specific behaviour must configured if desired
- Administration is the same for all platforms out of the box
  - Server operations are controlled by the same `server` script
  - Logs, trace, and configuration live in the hierarchical file system and are tagged with the appropriate code page for easy viewing and editing
  - Existing server configurations can be brought to z/OS from distributed without modification
- An extremely light-weight, single process runtime
  - Removes deployment and runtime complications introduced by the split process, multi-JVM runtime of traditional WAS for z/OS

Installation

IM and Liberty Profile

Liberty Profile comes with WAS z/OS, but it requires you specify it to have the binaries installed when WAS z/OS itself is installed



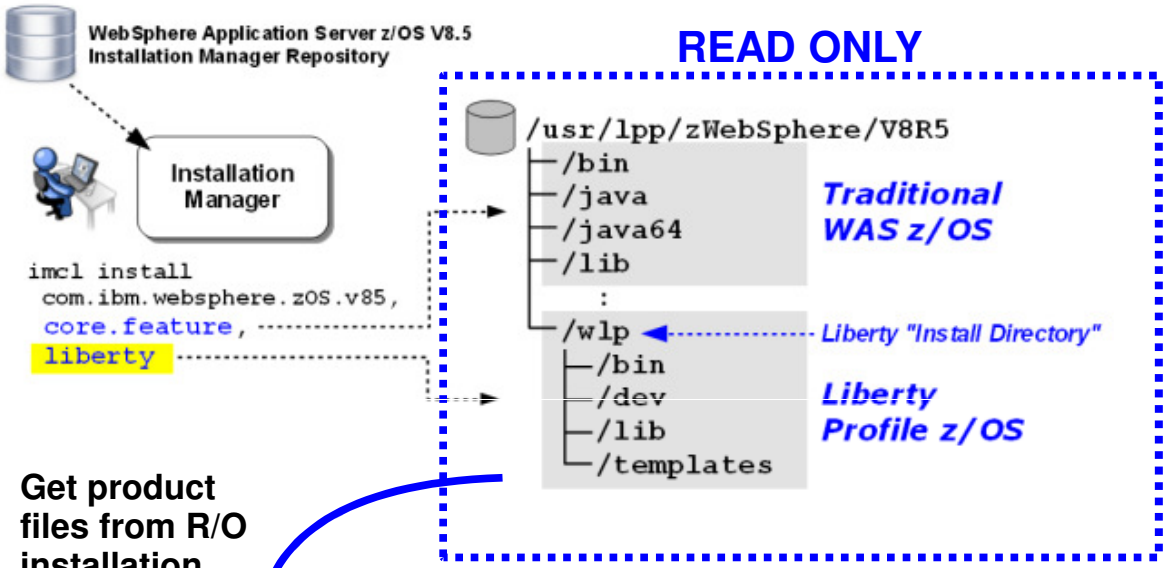
Liberty doesn't take much space ... recommend installing it

This populates the `/wlp` directory with Liberty binaries

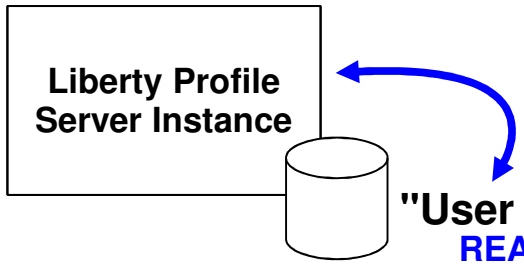
Server configurations are held in a separate "user directory" ...

Install vs. User Directory

Your install directory will very likely be mounted READ only ... which means the Liberty Profile configuration files are held in a different location:



Get product files from R/O installation directory



Configuration is read from user directory, updates made to files in user directory

"User Directory"
READ/WRITE

We'll see how this directory is populated in the next section

First Server

UNIX Environment Requirements

The following UNIX environment variables are needed or recommended:

`JAVA_HOME=` ← Liberty requires a 64-bit Java at level Java 6 or Java 7.
The Java that comes with traditional WAS z/OS works very well

`WLP_USER_DIR=` ← This variable tells the shell script where the user directory is located

`_BPXK_AUTOCVT=ON` ← This variable enables auto conversion in z/OS USS for tagged files

Set variables at command prompt or in .profile

Liberty Profile file are tagged ASCII which is what tells z/OS editors (OEDIT) to auto-convert. The variable above is what enables this.

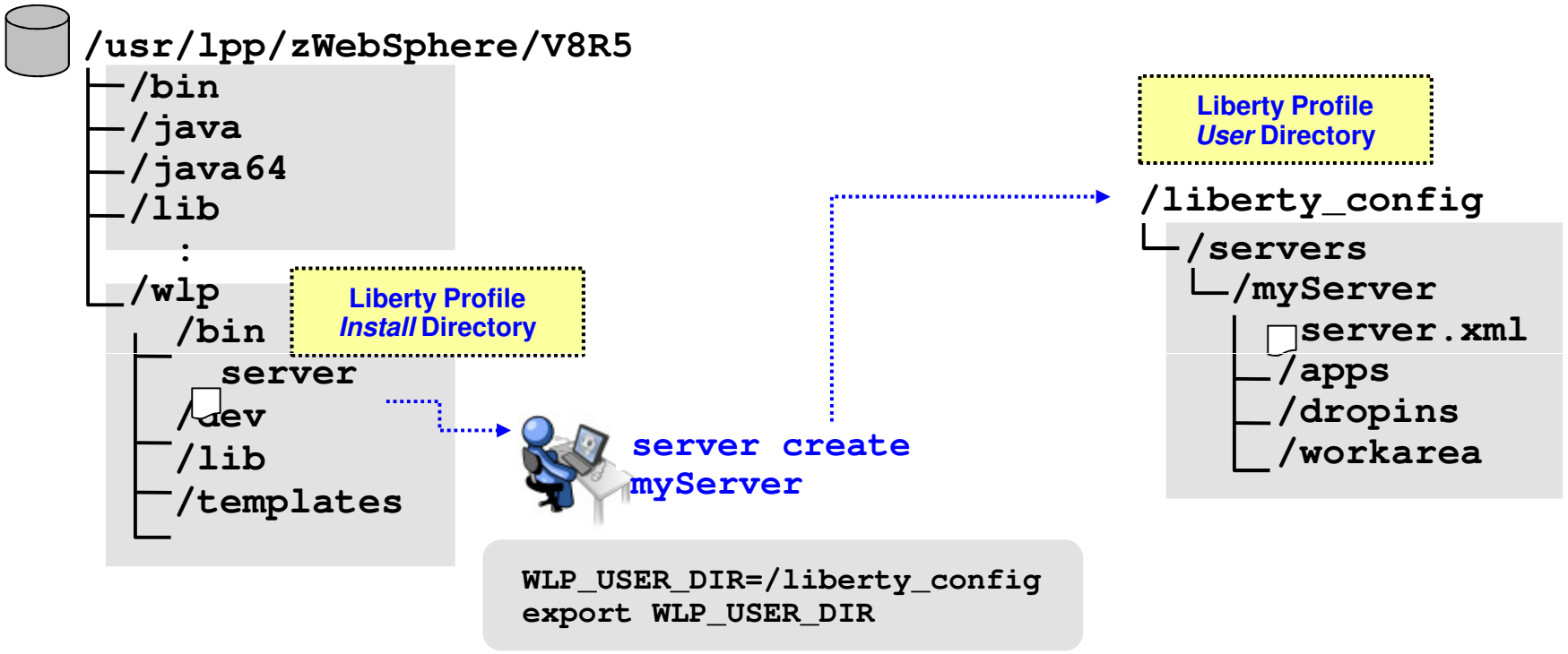
Do not use conversion tools like "a2e" or "viascii" to work on tagged files. You could harm the tagging, which Liberty Profile relies on.

You can tag a file with: `chtag -t -c iso8859-1 <filename>`

Command `ls -lT` on a directory will show what files are tagged

Using "server" Shell Script to Create Server:

The "server" shell script will populate the user directory with the key files and directories needed by Liberty Profile:



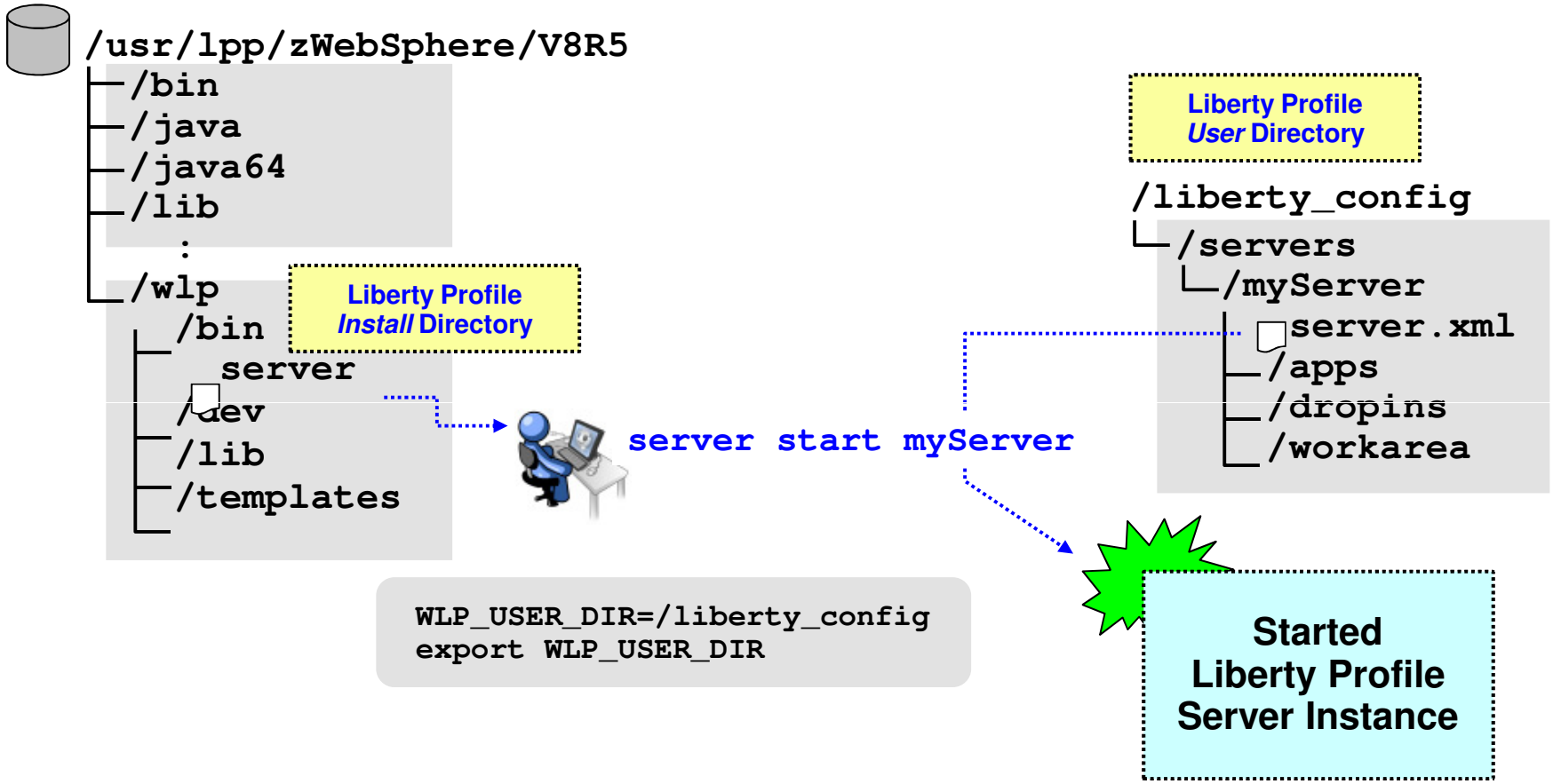
This is done in UNIX shell ... Telnet or OMVS

UNIX environment variable WLP_USER_DIR tells script where to put files

In this example "create" is the verb and "myServer" names the server instance

Starting The Server

The "server" shell script is used to start the server as well:



Verb here is "start" with the server instance to be started named

To stop a server the verb is "stop" with the server instance name provided

Shell script has other verbs as well

First Server Somewhat Unusable at First

Two things make your initial server unusable at first:

The default `server.xml` has `host=` defaulting to "localhost," which works fine on platform where a browser may run locally, but not well on z/OS where browsers are all remote.

This is a security design ... not opened to outside unless you tell it to

Simple update -- edit `server.xml` and change `host="localhost"` to `host="*"`

Liberty Profile will detect change and dynamically implement

By the way, default HTTP port is 9080 so you may see port conflicts if that port is already in use on your system. That value may be changed dynamically as well.

By default no applications are deployed initially

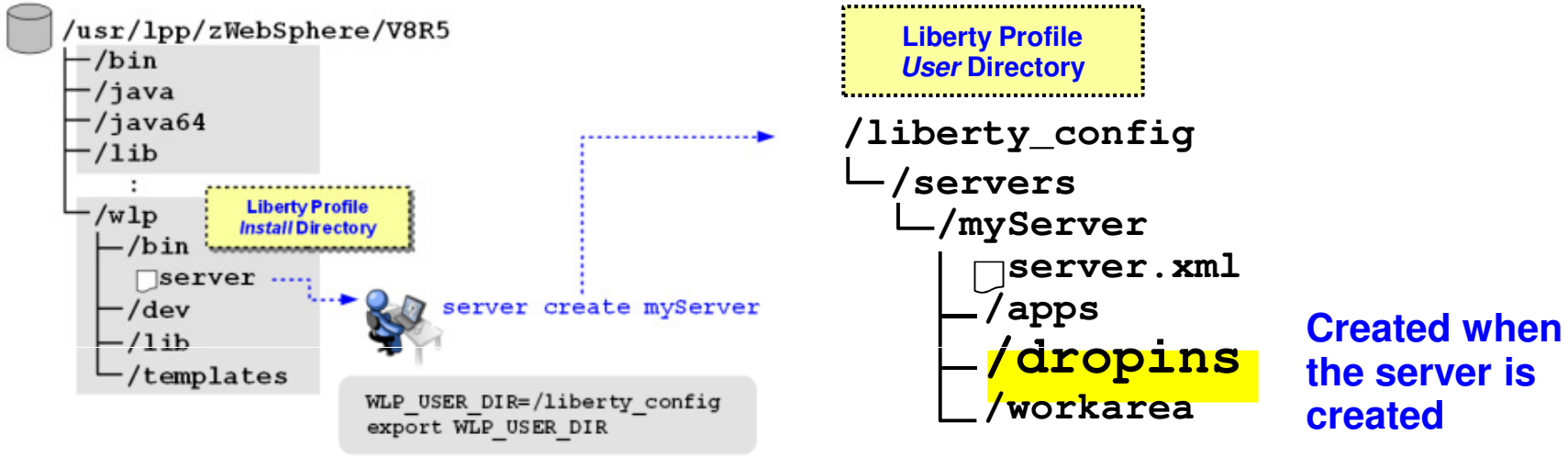
Simple update -- Liberty Profile allows application files to be "dropped" into a directory where it will be detected and auto-loaded and auto-started.

We'll look at deploying applications in the next section.

Deploying Applications

"Dropins" Directory for Application Files

The first way to deploy an application is to simply drop the file into the folder:



The `/dropins` directory is monitored by the server instance

Application WAR files placed in this directory are read in and app started

Remove the WAR file and the app is stopped and removed

Coding Applications in server.xml

The other way is to explicitly code the application in the XML:

```
<server description="server1">

  <!-- Enable features -->

  <featureManager>
    <feature>jsp-2.2</feature>
  </featureManager>

  <application location="/path/ATS_servlet.war" />

  <httpEndpoint id="defaultHttpEndpoint"
    host="*"
    httpPort="9080"
    httpsPort="9443" />

</server>
```

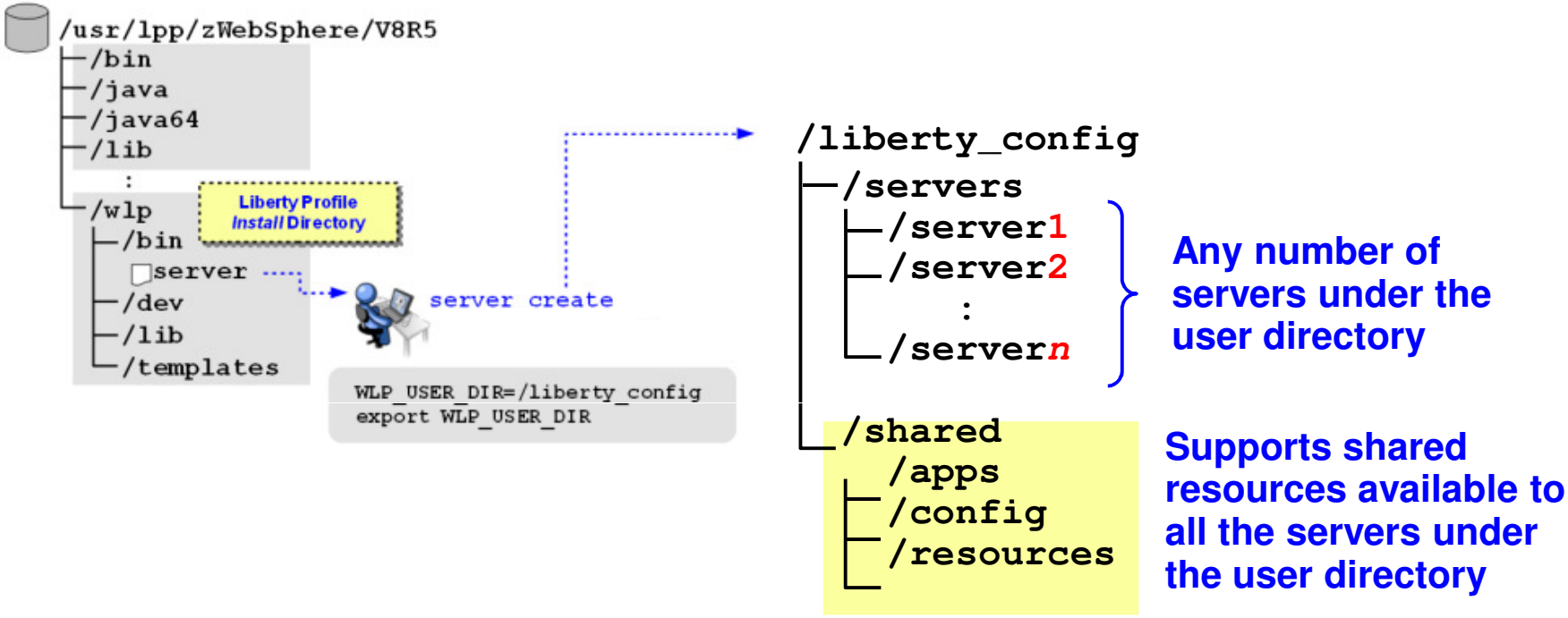
Any location provided server instance ID has READ to the location

There is a way to code a substitution variable to have multiple server instances share the same application files

Multiple Servers

Multiple Servers under User Directory

You can use `server create` to create multiple server instances:



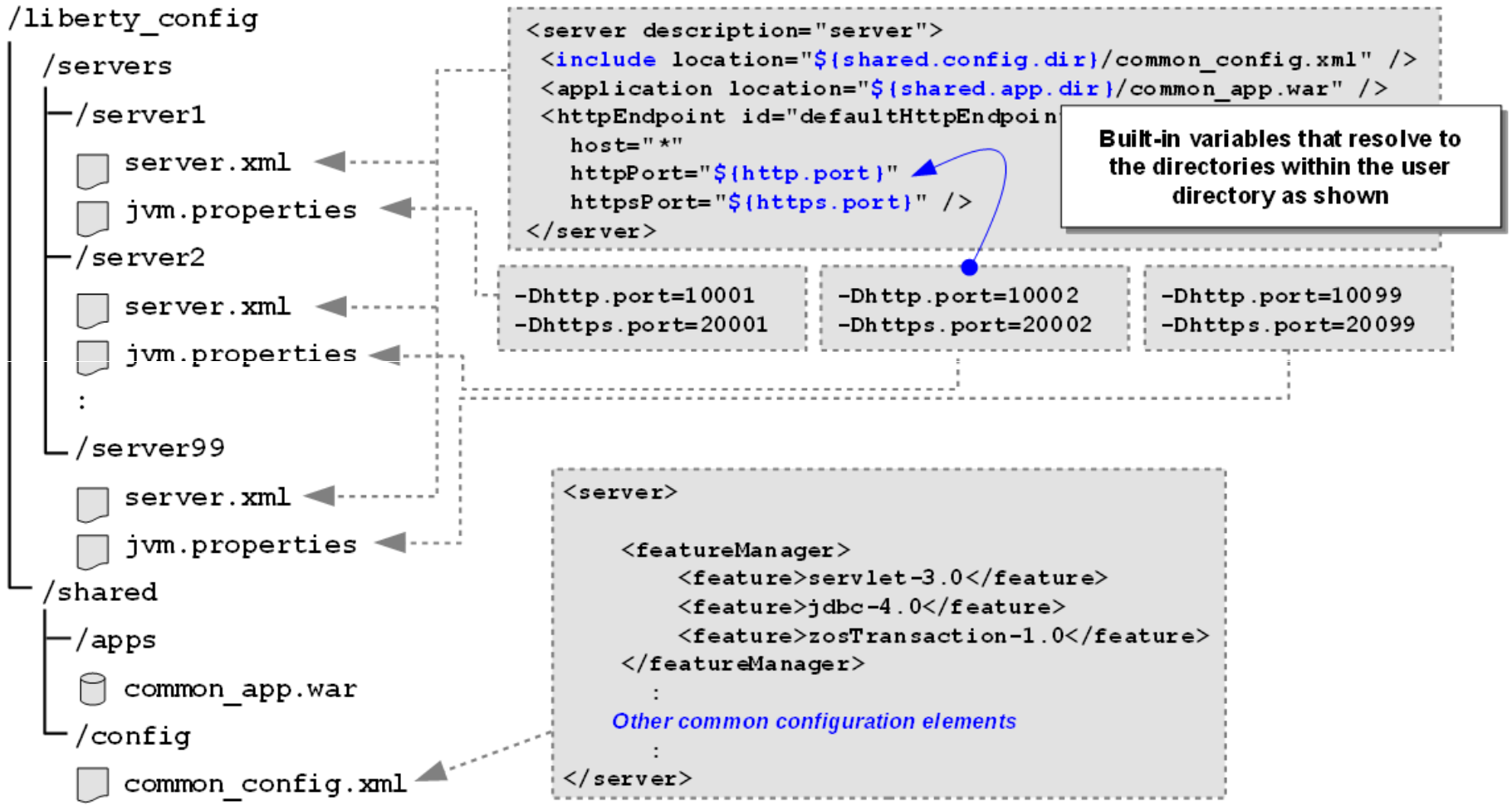
Any given WLP_USER_DIR may have multiple server instances within it

(In addition you may have multiple user directories)

Liberty Profile provides built-in substitution variables to access the shared resource directories. This allows several servers to have common configuration, each pointing to the shared resource directories.

Build-in Vars ... Multiple Servers, Common Artifacts

Illustration of 99 servers sharing common artifacts:

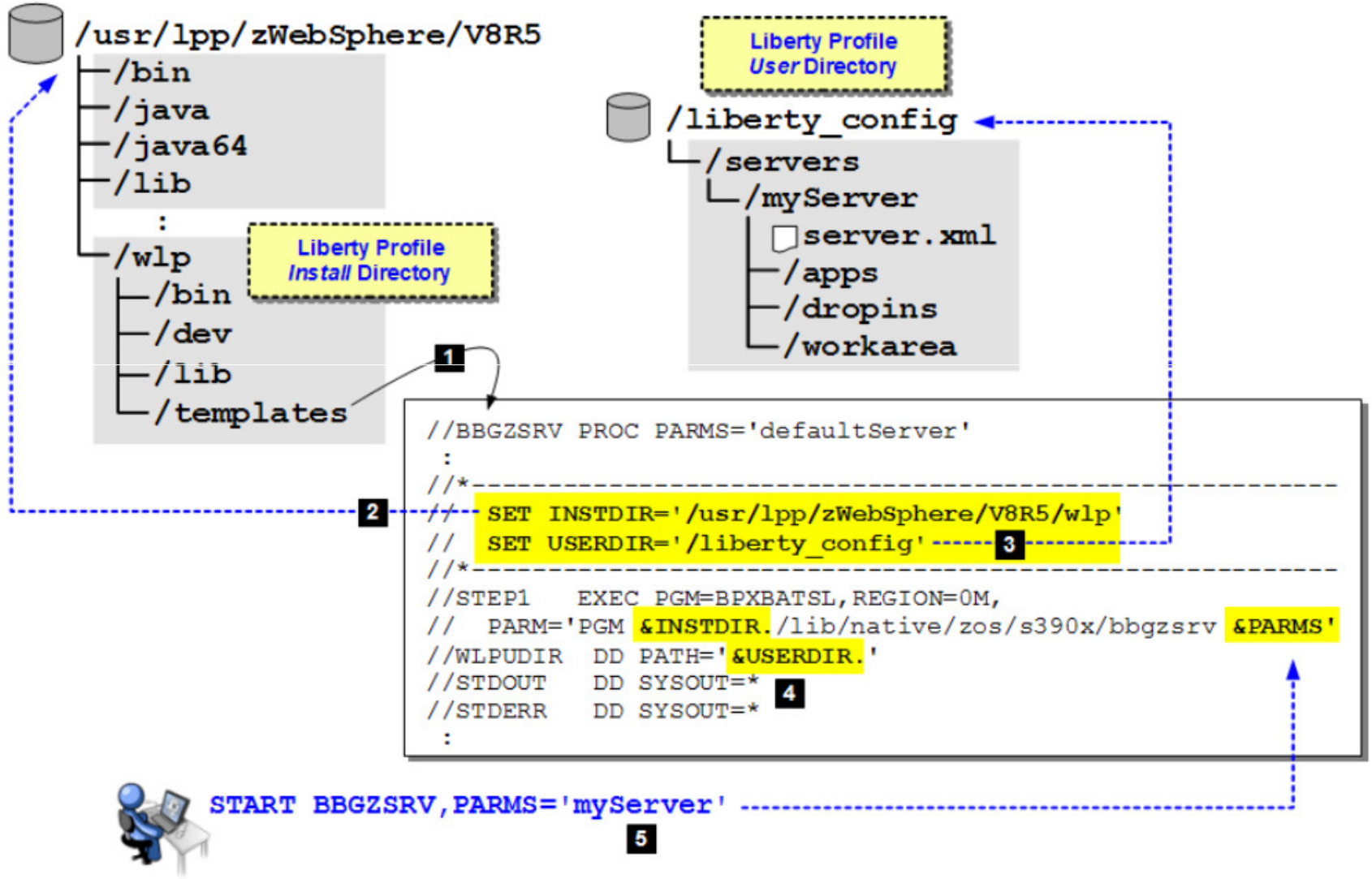


Changes to shared artifacts dynamically picked up by all server instances

z/OS Started Tasks

Run as z/OS Started Task with Supplied JCL

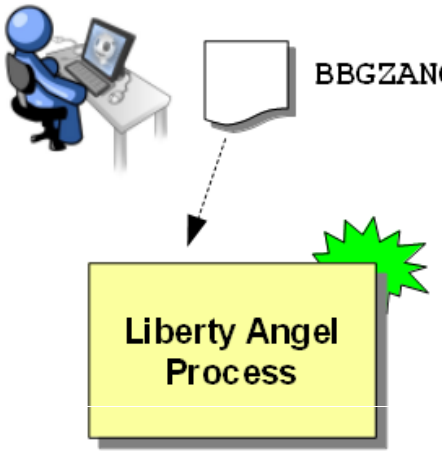
Copy out the sample JCL, simple updates, then start as z/OS started task



z/OS Extensions

Liberty "Angel" Process

The "Angel" process provides access to z/OS authorized services



- **Not strictly required**
  Only required if there's a Liberty server instance on the LPAR that requires access to z/OS authorized services
- **If needed, then only one per LPAR**
  Whether one Liberty server instance or a thousand
- **Very lightweight**
  Very little memory, almost no CPU once started, no TCP ports, no configuration files
- **Access to authorized through `SERVER` profiles**
  Small handful of `SERVER` profiles to set up ... you grant `READ` to server ID
- **Services: SAF, WLM, RRS, z/OS DUMP**
  Of those, only RRS and z/OS DUMP *require* Angel process; SAF and WLM will work without but not as efficient as authority check then done for every call rather than once

z/OS Extensions for Liberty Profile

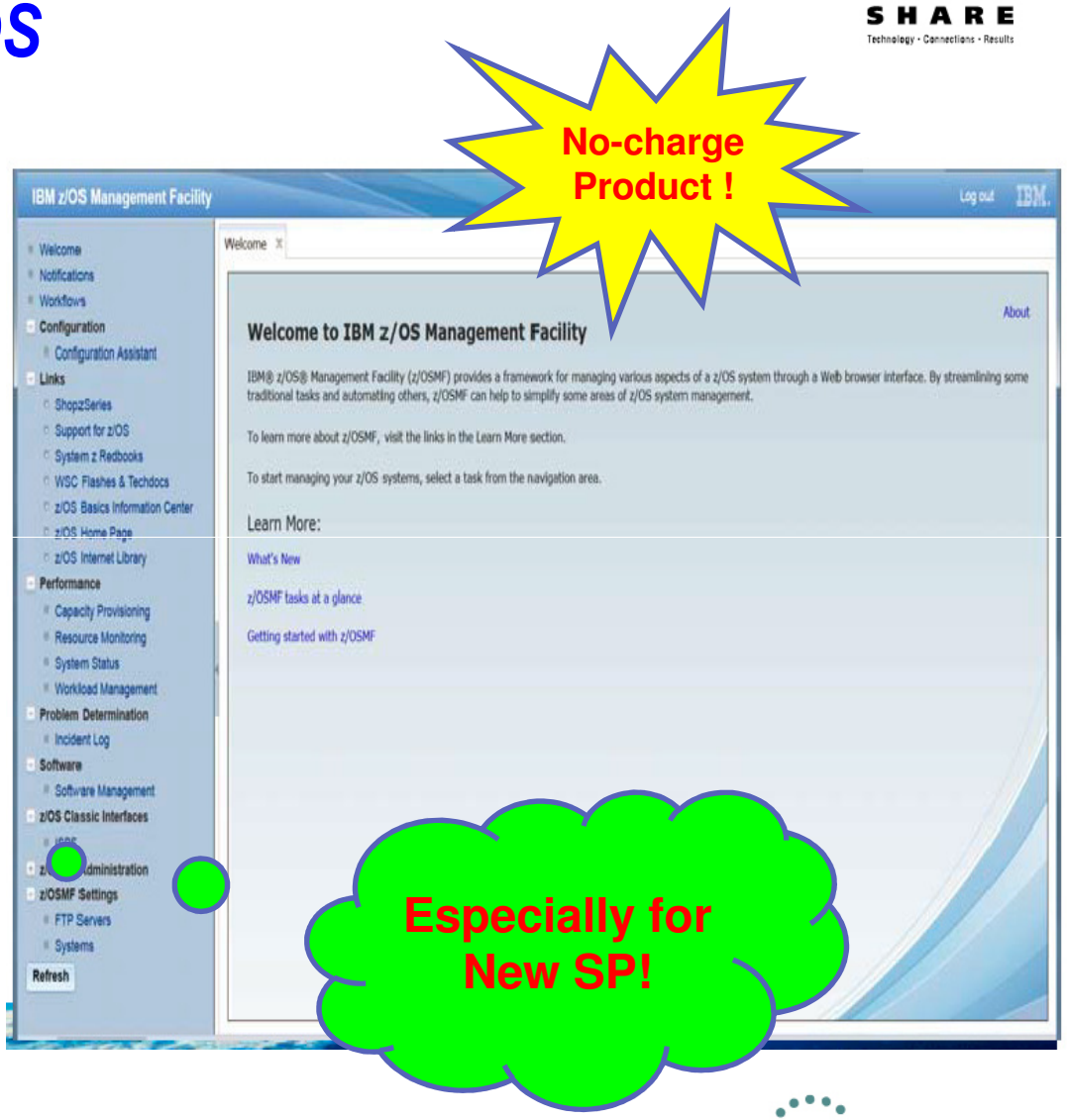
Four areas of platform exploitation:

SAF	• Use SAF for authentication repository (userid and passwords) • Use SAF for trust and key store (digital certificates) • If Angel, then SERVER profile: `BBG.AUTHMOD.BBGZSAFM.SAFCRED`
WLM	• Provide transaction classification (TC) to work requests • Elements in `server.xml` provide classification rules *(not separate XML file like trad. WAS z/OS)* • Common use-case: provide separate reporting classes for work • If Angel, then SERVER profile: `BBG.AUTHMOD.BBGZSAFM.ZOSWLM`
RRS	• Use for JDBC Type 2 with RRS for transaction management • Angel process required for this • SERVER profile: `BBG.AUTHMOD.BBGZSAFM.TXRRS`
DUMP	• Provides ability MODIFY request for SVCDUMP or Java Transaction (TDUMP) • Angel process required for this • SERVER profile: `BBG.AUTHMOD.BBGZSAFM.ZOSDUMP`

z/OS MF

z/OSMF is Web 2.0 application; no client install; manage z/OS from z/OS

- The IBM z/OS Management Facility provides a Web-browser based management console for z/OS.
- Helps system programmers to more easily manage and administer z/OS by simplifying day to day operations and administration.
- More than just a graphical user interface, the z/OS Management Facility provides real value
  - ►Automated tasks can help reduce the learning curve and improve productivity
  - ►Embedded active user assistance (such as wizards) guides you through tasks and helps provide simplified operations



One New Enhancement in z/OSMF V2.1

z/OSMF is designed to use the **Liberty profile** in IBM WebSphere Application Server for z/OS, V8.5, this is expected to improve and simply -

- **Packaging:** Provide a smaller and faster product package that requires fewer resources. WAS OEM is now removed from z/OSMF, WAS Liberty Profile is part of z/OSMF package, the new package footprint is 300+ MB
- **Installation and Configuration:** Setup no longer requires two separate configurations for runtime(WASOEM) and application(z/OSMF), reduced to one single stream configuration (one setup only), this change also results fewer overall prompts and variables.
- **Service:** Follow the normal z/OS model through normal SMPE receive/apply and restart z/OSMF to pick up new service, no longer required separate step for activation
- **Performance:** Faster startup (<15 seconds);

  Memory requirement(1GB+, previously required 2 GB)