WebSphere Liberty z/OS

Applications and Application Deployment

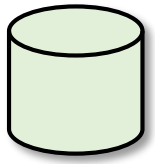
Objective of this Presentation

Provide an understanding of the application types supported by Liberty

Provide a general understanding of the API model of Liberty, particularly as it relates to the API model of WAS traditional

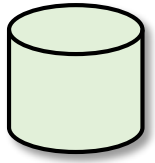
Provide an understanding of the deployment model of Liberty

Application Types Supported by Liberty

WAR 

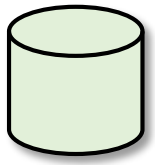
Web Application

- JSPs and Servlets
- Accessed over the network using a client (browser, or a REST client)
- Packaged in a Web ARchive (WAR) files

EAR 

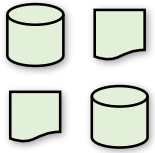
Enterprise Application

- EJBs
- Accessed using network protocols (RMI/IIOP) or message queueing (JMS)
- Packaged in Enterprise ARchive (EAR) files

EBA
ESA 

OSGi Application

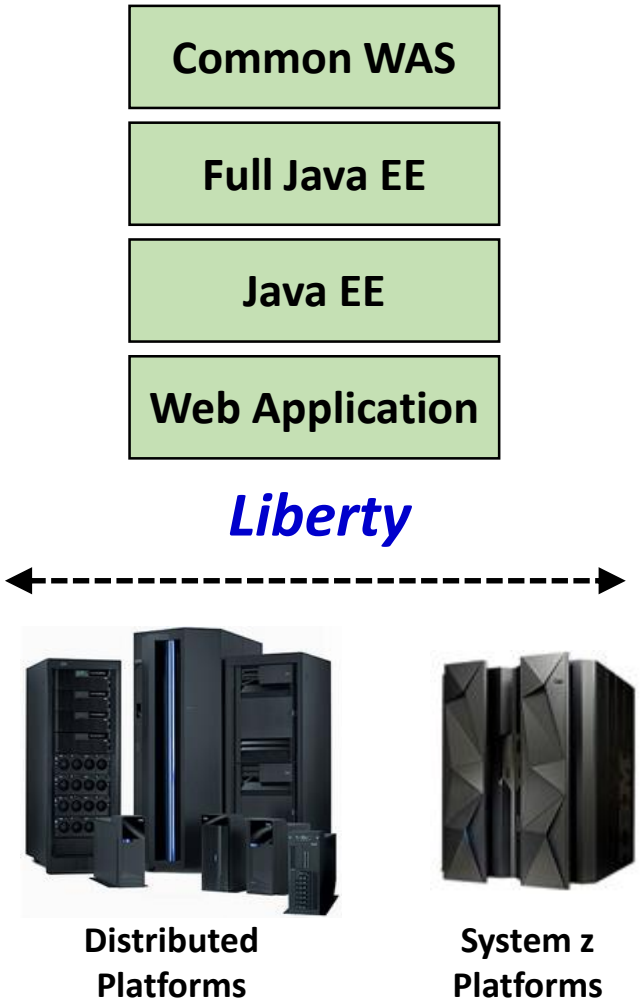
- Java-based architecture that implements a dynamic component model
- Can be installed, started, stopped, updated, and uninstalled without requiring a restart
- Packaged in either Enteprise Bundle Archive (EBA) or Enterprise Subsystem Archive (ESA)

Archives
Files
Directories 

"Loose" Application

- Not a different kind of application (it's a different way to provide application components to server)
- This allows your application components to be located in different places and run in Liberty
- Particularly good for development: run applications without export and deploy

Liberty APIs Across the Platforms



Liberty is supported across many different OS platforms:

- Windows, AIX, HP-UX, Solaris, IBMi, Linux, Linux for System z, z/OS

Same programming APIs across the platforms:

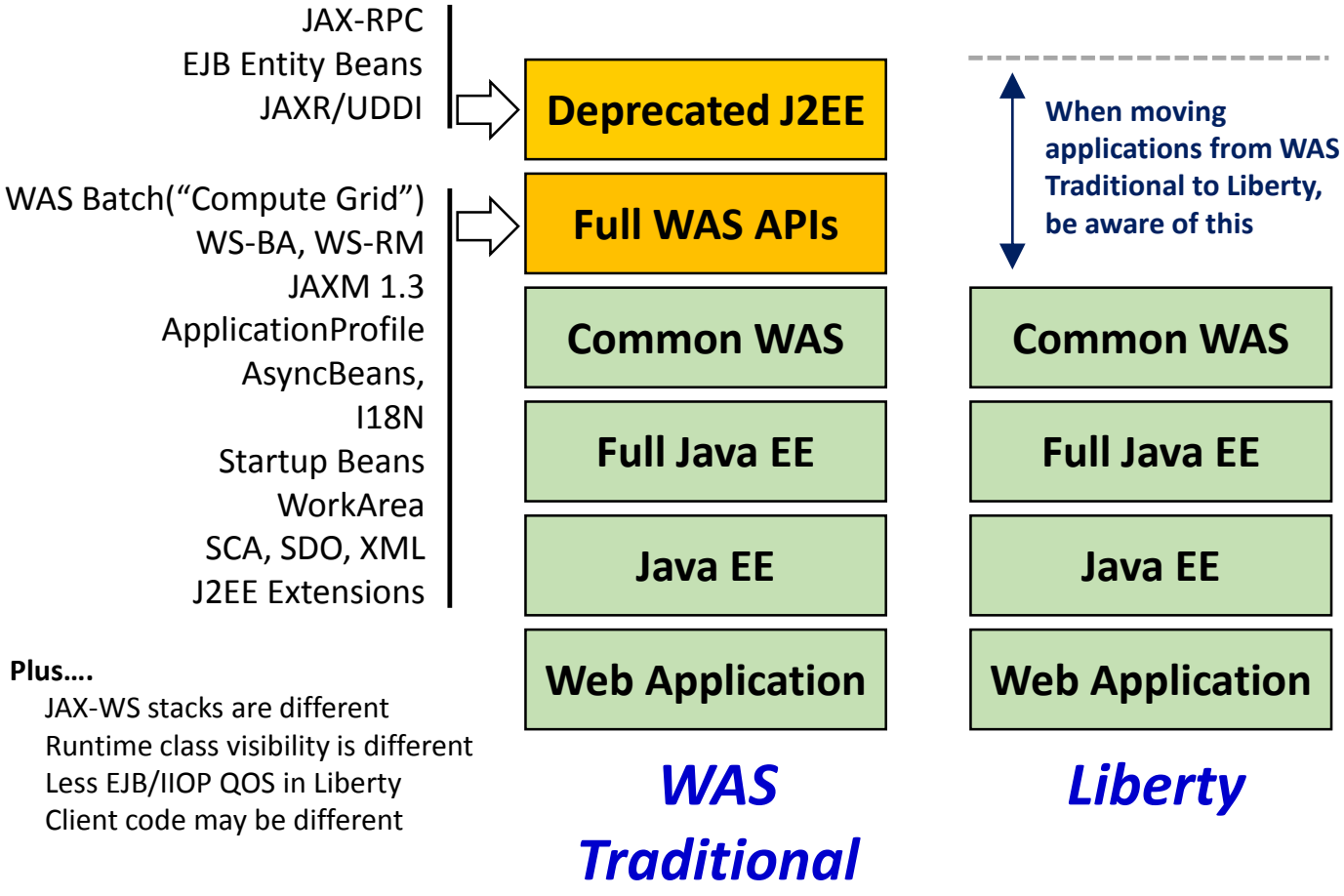
- When comparing the "Network Deployment" level of Liberty
- Distribute platforms have other "editions" which have different subsets of the full Liberty features (Base, Core, Express, Developers)

All else equal, applications can move across platforms

- "All else equal" -- same version of Liberty, Liberty features configured the same, data connectivity definitions are in place, security requirements are the same, etc.

What about WAS traditional vs. Liberty?

WAS traditional APIs Compared to Liberty APIs



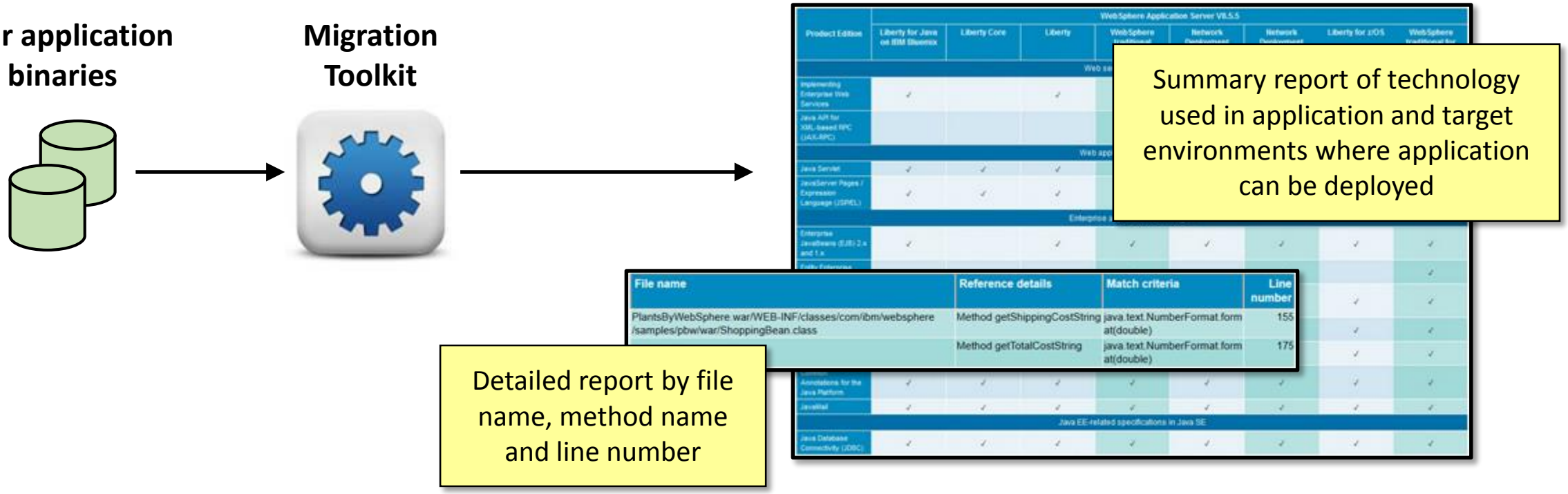
The API set is very similar, but they are not exactly the same

Application mobility:

- **WAS traditional to Liberty -- be aware of deprecated APIs and any use of "Full WAS" APIs in applications**
- **Liberty to WAS traditional -- fewer concerns about API differences**

An toolkit to evaluate applications exists ...

Reference: Application Migration Toolkit



Main wasDev page:

https://developer.ibm.com/wasdev/downloads/#asset/tools-Migration_Toolkit_for_Application_Binaries

Technical Overview:

https://developer.ibm.com/wasdev/docs/migration-toolkit-application-binaries-tech/

Updates page:

https://developer.ibm.com/wasdev/blog/2015/03/13/announcing-websphere-liberty-migration-tools-updates/

The WP102110 Techdoc*

Located at the bottom of the Techdoc page:

Liberty or WAS Traditional ... How to Decide?

Additional function has been added to Liberty since it was first introduced. It is now Java EE 7 compliant and supports the Java 8 SDK. So the question arises: "What should I consider when deciding between using WAS traditional and Liberty z/OS?"

The following guide provides some considerations to take into account when comparing the two.

WP102110 - Liberty or WAS Traditional - CHARTS.pdf

WP102110 - Liberty or WAS Traditional - NOTES.pdf

Framework of the presentation:

Hyperlink

Executive Overview
A one-chart summary of the usage considerations presented in the document

Setting Context
Establishing terminology and providing background on the evolution over time of each runtime models

Application Considerations
Exploring the application interface considerations of each runtime model

Operational Considerations
Exploring the runtime operational considerations of each runtime model

Performance Considerations
Exploring the performance profile of each runtime model

Other Information for Consideration
A collection of other information you may find useful when making this decision

At that Techdoc page there's a section devoted to going through the considerations in a systematic manner

No "formula" ... the key considerations are around:

- The application design and the APIs it uses
- The degree of reliance on the CR/SR structure of WAS traditional on z/OS
- The degree of reliance on automated scripting (WSADMIN)
- Memory and GP (Liberty tends to use less of both)

For rest of presentation we assume Liberty

* http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102110

Liberty Application Development Good Practices

Liberty is a Java EE application server, so there's nothing unique about it compared to other Java EE runtimes

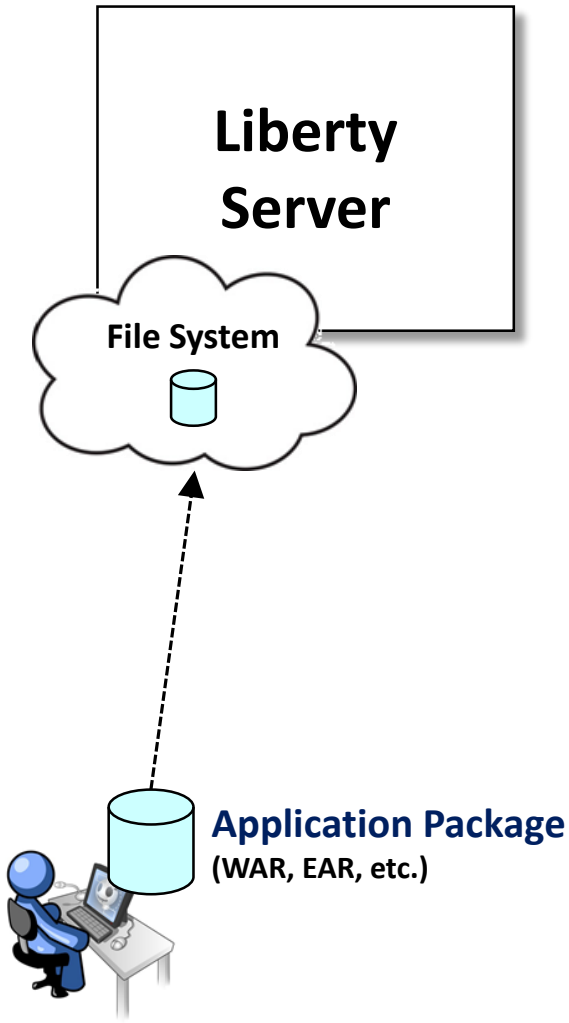
General good practices Java development ...

- Use good design practices
- Maintain good source control
- For best performance, make heavily used code as efficient as possible
- Profile the application prior to deployment into production
- Use proven change control processes

Application Deployment

Deploying applications into a Liberty z/OS runtime environment

The Liberty Deployment Model



Two choices:

1. Drop file in /dropins directory and let Liberty detect and dynamically load
2. Place file in another location and configure application into server.xml

Use whatever mechanism you wish to get application package file from your development / source control environment to a file system accessible by the server

Then the question is: Dynamic update? Or rely on server restart?

- In general, we see production environments *avoiding* dynamic updates.
- If dynamic updates are disabled, then application are loaded at server restart
- If dynamic updates, the trigger mechanism (polled, mBean) can be configured

Static Definition of Application

Application Location Pointer from `server.xml`

1. `<application location="<app_pkg_name>" />`

The server will search both the /apps directory and the /shared/apps directory for the application.

2. `<application location="${server.config.directory}/apps/<app_pkg_name>" />`

The variable resolves to the server's configuration directory. This definition points to the /apps directory under that.

3. `<application location="${shared.app.directory}/<app_pkg_name>" />`

The variable resolves to the /shared/apps directory under WLP_USER_DIR.

4. `<application location="/<full_path>/<app_pkg_name>" />`

You may provide an explicit pointer to a path and file and load application from any accessible location.

```
<config updateTrigger="polled" monitorInterval="500ms"/>
                       "mbean"
                       "disabled"
```

Controls whether the server.xml change is dynamically detected and acted upon.

Application Load from a "Dropins" Directory

Dynamic update from "dropins" directory

The default location is the `/dropins` directory under the server directory

```
<applicationMonitor dropins="${server.config.dir}/myApps" />
```

Sets "dropins" directory location to a directory you specify under the configuration directory

```
<applicationMonitor dropins="${shared.app.directory}/myApps" />
```

Sets "dropins" directory location to a directory you specify under the shared apps directory

```
<applicationMonitor dropins="/<path>/myApps" />
```

Sets "dropins" directory location to a location using an absolute path

```
<applicationMonitor dropinsEnabled="false"/>
```

Disables "dropins" monitoring and dynamic loading of applications

If Dynamic Update, then Controlling When Dynamic Update Takes Place

Note: dynamic update and "dropins" are related, but are not the same thing. You can have a statically-defined application and replace the package with a new file. Liberty can detect change and reload dynamically if you choose.

1. `polled` -- based on `monitorInterval` value
   `mbean` -- based on mbean invocation
   `disabled` -- dynamic configuration updates turned off

2. `ms` (milliseconds)
   `s` (seconds)
   `m` (minutes)
   `h` (hours)

```
<applicationMonitor updateTrigger="<value>" pollingRate="<value>"
                    dropins="<value>" dropinsEnabled="<value>"/>
```

3. Location as discussed on the previous chart

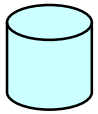
4. `true`
   `false`

Server restart
`updateTrigger="disabled"`

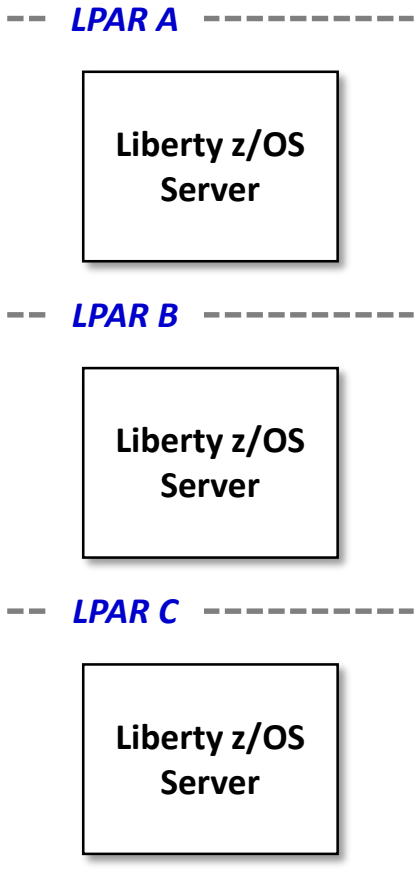
mBean Triggered
`updateTrigger="mbean"`

Timer-based
`updateTrigger="polled"`
`pollingRate="<value>"`

Rolling an Application Update Across LPARs



Application
In a shared location



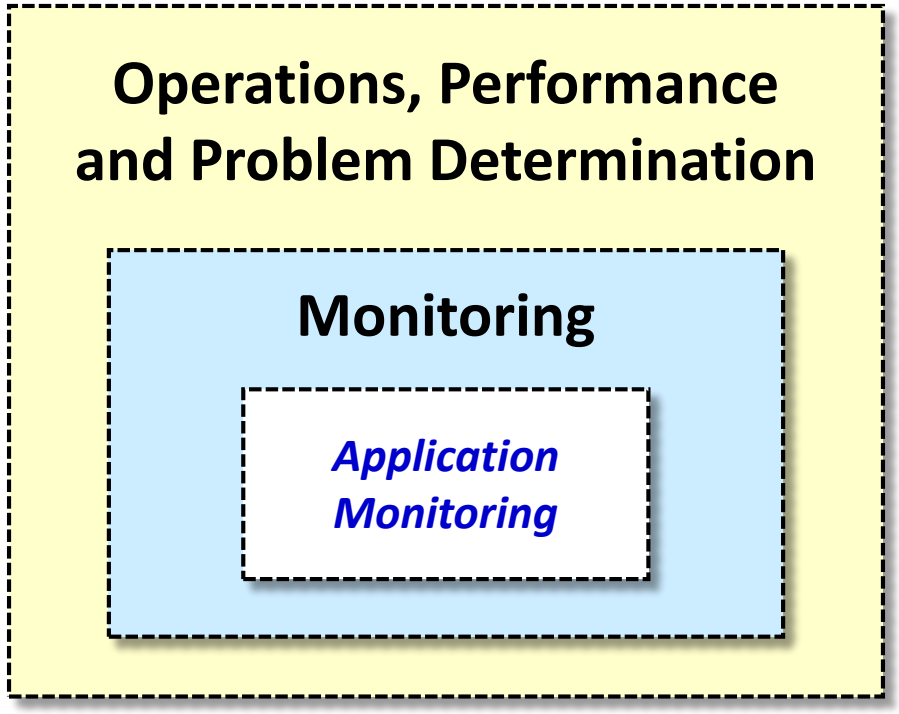
Framework of the approach:

- **Assume application is in a shared location accessible to all servers**
  If each server has its own unique copy of the application, then it's a matter of updating each unique copy.
- **Dynamic update is either disabled or mBean invoked (that is, not polled)**
- **Update shared copy of application**
- **Update in each server -- either server restart or mBean-invoked update**

Potential complicating factors:

- **When session affinity is required due to application design**
  Then you need to insure session persistence is enabled so affinity can be re-established
- **Stopping the flow of work to a server in which the application is to be updated**
  This is a function of the front-end routing mechanism you're using to route across LPARs
- **Cases where the application update implies simultaneous mixed-levels can't be tolerated; for example: a significant change to the backend data model**
  In this case you may need to schedule a Sysplex-wide update during a maintenance window

Application Monitoring?



Monitoring your applications is an important subject

It's part of the broader "monitoring" topic

Which is part of the even broader topic on operations, performance and PD

We have an entire unit dedicated to those topics.

Summary

Application deployment is fairly simple -- upload application package and make it available to the Liberty z/OS server. No "deploy through Admin Console" needed.

The key questions are:

- **Will the /dropins mechanism be used? Or <application> tag reference to file?**
- **How will updates be handled -- manually, mBean invoked, or timer-based polling?**

Reference

WebSphere Liberty 16.0.0.x Knowledge Center

http://www.ibm.com/support/knowledgecenter/en/SS7K4U_liberty/as_ditamaps/was900_welcome_liberty_zos.html

WebSphere Knowledge Center collection on the topic of migration

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg27008724

Migration Toolkit for Application Binaries

https://developer.ibm.com/wasdev/downloads/#asset/tools-Migration_Toolkit_for_Application_Binaries

https://developer.ibm.com/wasdev/docs/migration-toolkit-application-binaries-tech/

https://developer.ibm.com/wasdev/blog/2015/03/13/announcing-websphere-liberty-migration-tools-updates/