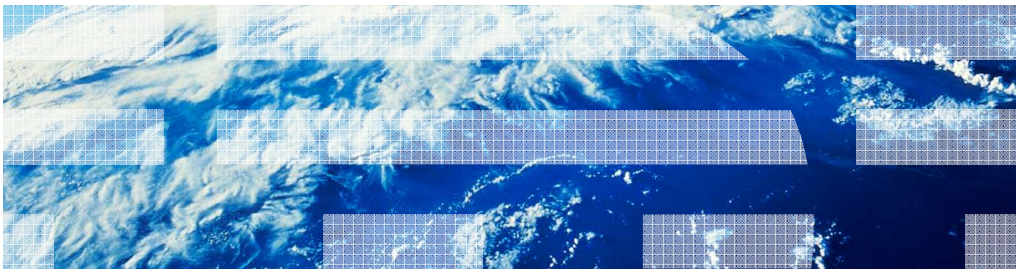
IBM WebSphere Application Server V8.5

Application manager



WebSphere software

This presentation describes support for the application manager included in IBM WebSphere Application Server V8.5 Liberty Profile

Overview

This section contains an overview of the Application Manager.

What is the application manager?

- WebSphere Application Server V8.5 Liberty profile only determines when to trigger actions on applications:
  - Deploy
  - Update
  - Remove
- Log messages will start with the "CWWKZ" prefix

The application manager is the component in WebSphere Application Server Liberty profile responsible for initiating application installs, updates and uninstalls from the server. It does not process the installation of the applications themselves – that is the role of the application handlers. The role of the application manager is purely as a portal which is used to initiate the process.

The application manager can be looked at under three different scenarios – deploying an application, updating an application and removing a deployed application. Deploying and removing applications have two different paths depending on if you use the dropin monitor or server.xml. The updating of applications is identical no matter how it is deployed.

When updating applications the application manager removes the application, and then re-deploys the application in one action to ensure the application is in the correct state. When removing applications the application manager tells the necessary application handler to remove the application and then removes the application handler.

When looking in the logs for actions performed by the application manager the message key prefix used is “CWWKZ” followed by four numbers and a letter – the CWWKZ is the important identifier to look for as those messages originate from the application manager.

Usage scenarios

The Application Manager is used in the following scenarios.

Deploying an application

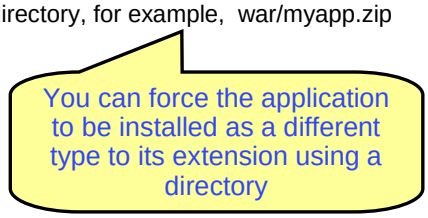
- There are two ways to deploy an application to a Liberty profile server.
- Via dropins folder:
  - copy application to dropins folder to trigger deployment
  - naming conventions:
    - <appname>.<type>, for example, myapp.war, or mayapp.ear
    - <type> directory, for example, war/myapp.zip

There are two different ways to deploy an application to the application server.

The quickest and easiest way to deploy an application to the server is to use the dropins folder. This folder by default is inside the server directory on the file system. You can copy an application inside the dropins folder and the server will immediately attempt to install the application (assuming the server is running – if it is not running the applications are installed on server startup). The dropin monitor works with applications which are inside containers such as zips, wars, ears and so on and applications which are within a directory structure. To make sure that the dropin monitor can understand enough about the application to install it you must follow some basic rules in regards to the file name and location within the dropin monitor.

Deploying an application

- There are two ways to deploy an application to a Liberty profile server.
- Via dropins folder:
  - copy application to dropins folder to trigger deployment
  - naming conventions:
    - <appname>.<type>, for example, myapp.war, or mayapp.ear
    - <type> directory, for example, war/myapp.zip



The rules for using the dropins folder are as follows:

- Application names are the names of the file or directory for the application is used as the application name and ID when deployed to the server (ignoring any file types).

- Location within dropins are all applications within the dropin directory that must either be deployed in the root of the dropin directory, unless the file has no type or you want to force the application to be deployed as a type different to the file type it has. If you want to force an application to be installed as a specific type you can put a directory underneath the dropin folder whose name is the type you want to force the application to be installed as. You can then place all applications you want to be forced to be that type on the server underneath that directory. A quick example is if you have an application which is in a .zip container and you want to deploy it as a “war” application then you can put it under “dropin/war/application.zip” and the dropin monitor will install it as a war. You can put as many applications as you like under these sub-directories.

- File extensions are any application installed directly under the dropin folder (and not under a directory which forces a type) need to have a file extension. This extension is used to define the type of the application so that the server knows what to try and deploy it as. If an application does not have an extension then it is treated as a directory which defines a type and will not work. This applies to applications installed as a single file or as a directory – both will need an extension (eg “.war”).

Deploying an application

- There are two ways to deploy an application to a Liberty profile server.
- Via dropins folder:
  - copy application to dropins folder to trigger deployment
  - naming conventions:
    - <appname>.<type>, for example, myapp.war, or mayapp.ear
    - <type> directory, for example, war/myapp.zip
- via <application> element in server.xml
  - <application name=" myApplication" type="war" location=" myAppZip.zip" />

There is also an optional “autoStart” element

```
<application name=“myApplication” type=“war”
location=“myAppZip.zip” />
```


The other way to deploy an application is to add an application entry into the server.xml for the server. You can see an example server.xml entry at the bottom of the slide with four attributes:

The “name” needs to be unique and depending on the application it can also be used as the context-root of the application. For a war application which is not inside a “.ear” it is used as the context-root, but for war applications inside .ear applications then it is not used. eba applications define the context roots themselves so they will not use the name as the context-root. The name is also what the server uses when referring to the application in the logs.

The “type” is how you tell the server the type of the application which you are defining. You use this instead of the file extension so, as is in the example, you can deploy a .zip file as a .war application without needing to change the extension type. This needs to match an application type which is supported by the server (so giving it "zip" as a type will not work, as that is not an application type).

The “location” is what the server uses to find the application you are trying to install. This can be an absolute file path - such as "C:/myApplications/anApplication.war", a URL which you download the application from or it can also be just the file name of your application (including the file extension, if it has one). If you provide just the file name in the location field then the code looks for it in two locations - first it look inside the "apps" directory under your server's directory (which is "<libertyInstallLocation>/usr/servers/<yourServerName>/apps") - if the code does not find an exact file name match in that location it looks in "<libertyInstallLocation>/usr/shared/apps". If the code cannot find the file name in that location as well then the server will not install the application. Note that if you deploy an application from a URL the server will not detect updates made to the application at the URL.

There is also an optional “autoStart” element you can add to the application tag – this is not required and is purely optional. If you set autoStart to “false” then the application will not start on the server until an mbean on the server is told to start the application.

Deploying an application by way of the server.xml allows you more customization – you can define different class loading changes to an application in the server.xml which is not possible when deploying an application in the dropin directory.

Application updates

- Application updates are looked for by an update monitor
- There is one update monitor per application
- The update monitor monitors all files and directories within an application for any changes (addition of files or directories, deletion and changes)
- Not all file changes require an update
- When an application is updated it is restarted

This slide covers the default behavior of the application update monitor . During application deployment an application update monitor is started to monitor for any changes to the application. The relationship between an application update monitor and an application is 1:1 where each application will have its own update monitor.

The job of the update monitor is to monitor all the directories and files of an application for changes. It checks the files and folders every 500 milliseconds by default. If it detects that a file or directory within the application has been added, modified or removed it then checks to make sure that it is not a static file (static files are not “executed” as code and are only pointed to by an application, so changes to a static file is reflected within the application instantly without the need to update it). Once the update monitor has confirmed that a non-static file change has occurred it checks again to make sure that more file changes are not occurring (if copying a new file into an application, it can take a while for the copy to complete, so the monitor waits until no file changes occur between two “sweeps” of the files). Once the update monitor is happy that no more file changes are taking place it updates the application.

To update an application it is restarted on the server.

Removing a deployed application

- To remove an application deployed in the dropin folder remove it from the folder.
- To remove an application installed from the server.xml
  - Delete or comment out the entry in the server.xml. Once the application is stopped you can then remove the application from wherever it was located on your file system.
  - You can delete the application files and **not** remove the <application/> entry from the server.xml. This will put the application in a “stopped” state.
- An application in the stopped state:
  - Is not running
  - Cannot be accessed
  - Is still monitored by application update monitor, so will start if files are put back
  - Unique name is still reserved on the server

There are several ways to remove an application from a server – these depend on how the application was deployed.

For applications deployed in the dropin folder remove the application from the folder – the dropin monitor will detect that the application has been removed and will tell the state machine to stop the application.

For applications deployed by way of the server.xml comment out or delete the <application> entry in the server.xml. The state machine will then remove the application from the server. Once you have seen that the application has been successfully removed in the server logs you can then delete the file from wherever it was deployed (if you want to).

Once an application is removed then all references to it in the server are cleaned up completely and the update monitor is also stopped for that application.

You can also delete the application while it is still defined in the server.xml – this is not recommended but it possible. This does not fully remove the application from the server but puts it into a “stopped” state. When an application is in a stopped state it cannot be accessed and it is not running on the server, however its name is still reserved in the state machine and the application update monitor is still monitoring it’s location to see if the application is re-deployed there. To start an application which is in a stopped state. Copy it to where it is defined in the server.xml (or change the location entry for the application in the xml) and it is started again as normal.

Configuring the application manager

- The application manager is highly customizable
- The configuration options available are:

```
<applicationMonitor
pollingRate="500ms"
dropins="dropins"
dropinsEnabled="true"
updateTrigger="polled"/>
```

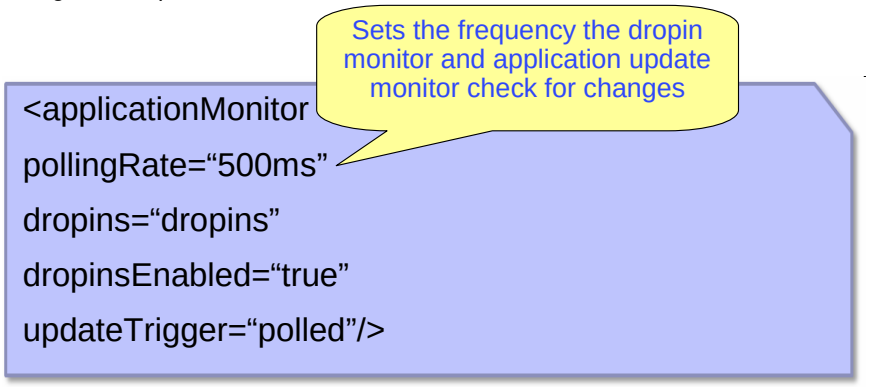

The application manager can be highly customized. All changes to the application manager occur under the "applicationMonitor" element in the server.xml.

The example xml entry on the slide contains the defaults which the server will use if you do not specify the attributes in the server.xml. You only need to specify the values you want to change in the server.xml – if a value is not specified then the default is used.

You can move, rename or disable the dropins folder and change the frequency which it checks for application addition or removal.

Configuring the application manager

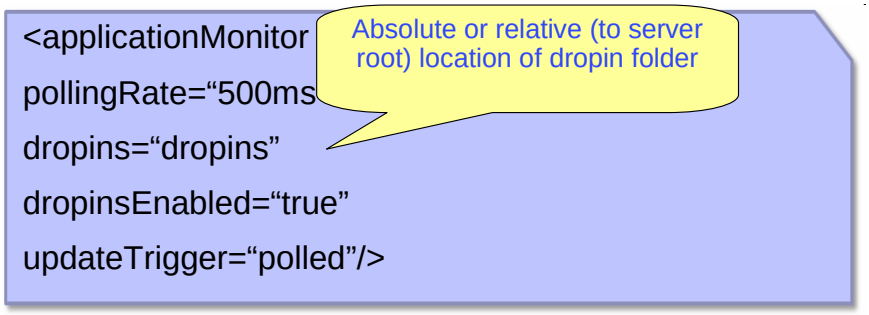
- The application manager is highly customizable
- The configuration options available are:



The "pollingRate" element sets the frequency between the dropin monitor looking for new applications or application removals, and how frequently the update monitor looks for changes to an application before updating it. The default value is "500ms" which is every half second.

Configuring the application manager

- The application manager is highly customizable
- The configuration options available are:



To move or rename the dropins folder use the dropins="" element where the content of the value for "dropins" is the location and name of the new dropins folder relative to the server's directory. If you change the dropins folders' location while the server is running any applications running from the old dropins folder is stopped.

Configuring the application manager

- The application manager is highly customizable
- The configuration options available are:

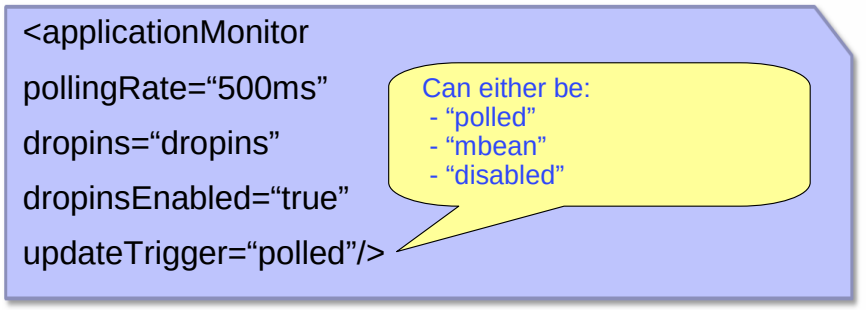
```
<applicationMonitor
pollingRate="500ms"
dropins="dropins"
dropinsEnabled="true"
updateTrigger="polled"/>
```

Enables or disables the dropin folder scanning

The "dropinsEnabled" element is a boolean to enable or disable adding and removing applications installed in the dropins directory. If you disable the dropins folder while the server is running any applications currently deployed from the dropins folder is removed from the server immediately.

Configuring the application manager

- The application manager is highly customizable
- The configuration options available are:



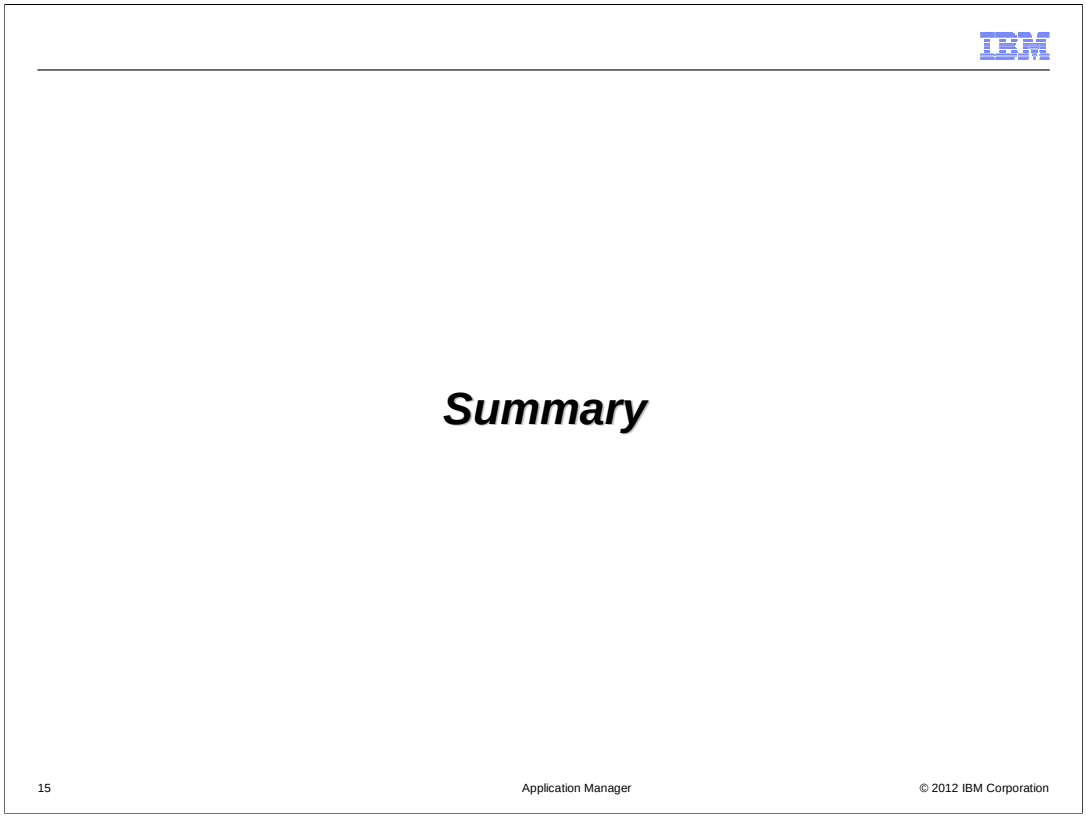
The "updateTrigger" element allows you to set how the server will detect updates to the applications. There are three options:

- "polled" is the default and will scan the applications by a frequency set by the pollingRate for any changes – once a change is detected the application is updated.

- "mbean" disables the application scanning and will activate an mbean which will only update the applications when told to by an external program to the server – the Liberty tools use mbeans to tell the server when to update applications by default.

- "disabled" completely disables application updates – the server will not update applications at all. Note that static content does not require an application to be updated by the server so any static file changes will still occur on the applications if updates are disabled.

You can have any combination of the configuration options displayed here– if you do not set a new value for any of the configuration options the default values are used even if you override any of the other attributes.



This section contains a summary of this presentation.

Summary

- Application manager: deploy, update and remove applications
- Two ways to deploy applications:
  - The dropins is the easiest way to deploy an application
  - The server.xml entry for an application allows more customization
- The correct way to remove an application differs depending on how it was deployed
- You can configure the application manager in the server.xml

In summary, this presentation covers these topics regarding WebSphere Application Server V8.5 application manager. The application manager is in charge of deploying, updating and removing applications from the server. There are two different ways to deploy applications, either by way of the dropin directory or by way of the server.xml. The dropins is very quick and easy to use for simple application deployments, although the configuration options available are more limited than deploying with a server xml (example, you cannot configure class loading for applications in the dropins folder). The server.xml entry for an application allows more customization and lets you add in extra configuration for the application (such as custom class loading). The process for removing an application depends on how it was deployed (dropins or server.xml). You can configure the application manager's dropin monitor and application update monitor to suit your needs

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WAS_V85_LP_Application_Manager.ppt

This module is also available in PDF format at: ../WAS_V85_LP_Application_Manager.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION 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 ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.