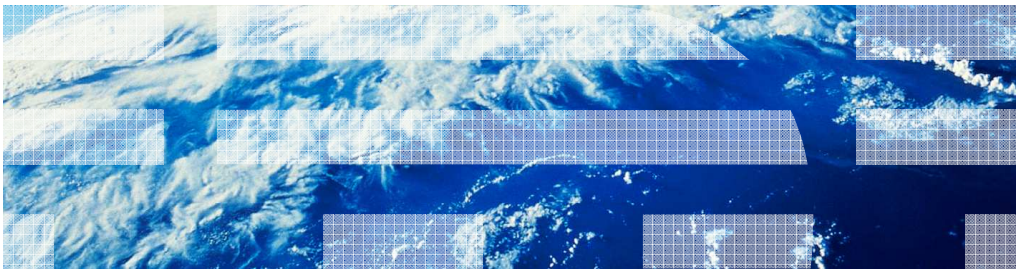
IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0

OSGi administration



WebSphere software

This module covers OSGi administration for IBM WebSphere® Application Server Feature Pack for OSGi Applications and Java™ Persistence API 2.0 OSGi Application

Table of contents

This module is an overview of the administration capabilities (and restriction) of the OSGi feature pack.

First the WebSphere Application Server 7 BLA framework is introduced.

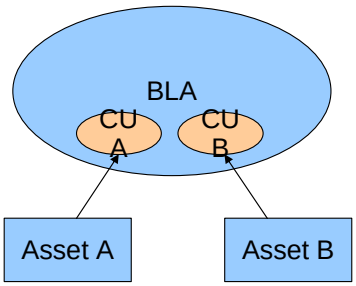
Then individual administrative capabilities are exposed starting from the base scenario up to the most complex, the update scenario.

Finally, there are some notes on how to troubleshoot problems in the administration layer.

This section will present an overview of OSGi administration.

BLA and OSGi

- Business level applications
  - Introduced in WebSphere Application Server Version 7
  - Mechanism of aggregating (JEE) assets and applications into logical units
  - Extensible asset types
    - Used in OSGi, SCA, ...



- OSGi application
  - .eba application file → BLA asset
  - Constraints
    - .eba asset can only be part of one BLA
    - For every application symbolic only one asset can be present
    - Some BLA operations like update are not supported for .eba assets

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/trun_app_bla.html

The majority of the Aries administration story is built around the Business level application (BLA) framework that is new in WebSphere Application Server version 7.

BLA provides two facilities; the first is the aggregation of assets and applications. For example one can have an OnlineShop BLA that comprises several JEE EARs that together implement all the services around an online shop. These applications can then be administered (in particular started and stopped) as a single logical unit. This aspect of BLA is primarily ignored by the OSGi application feature pack.

Second is the extensible asset system. Even though in base WebSphere Application Server BLA only supports JEE type assets (WARs and EARs) BLA itself is built ground up to allow custom extensions and in particular custom types of assets (and deployment units). This aspect makes it an ideal candidate for building the administrative capabilities of feature packs that introduce new programming models such as SCA and OSGi.

In BLA terminology there are three key concepts as depicted in the picture on the right: assets, composition units (CUs) and business-level applications (BLAs). Assets correspond to just plain binaries such as jar files, ears, wars, ebas that can be installed into WebSphere Application Server. An asset by itself is just a binary known to the application server. It only becomes useful when it gets added to a business level application, which can contain a combination of assets and BLAs. When an asset gets added to a BLA a composition unit is created that uniquely maps the asset to a BLA. The composition unit is the actual deployable element, which is started by the server runtime and can be mapped to different deployment targets. For more information see the information center link at the bottom.

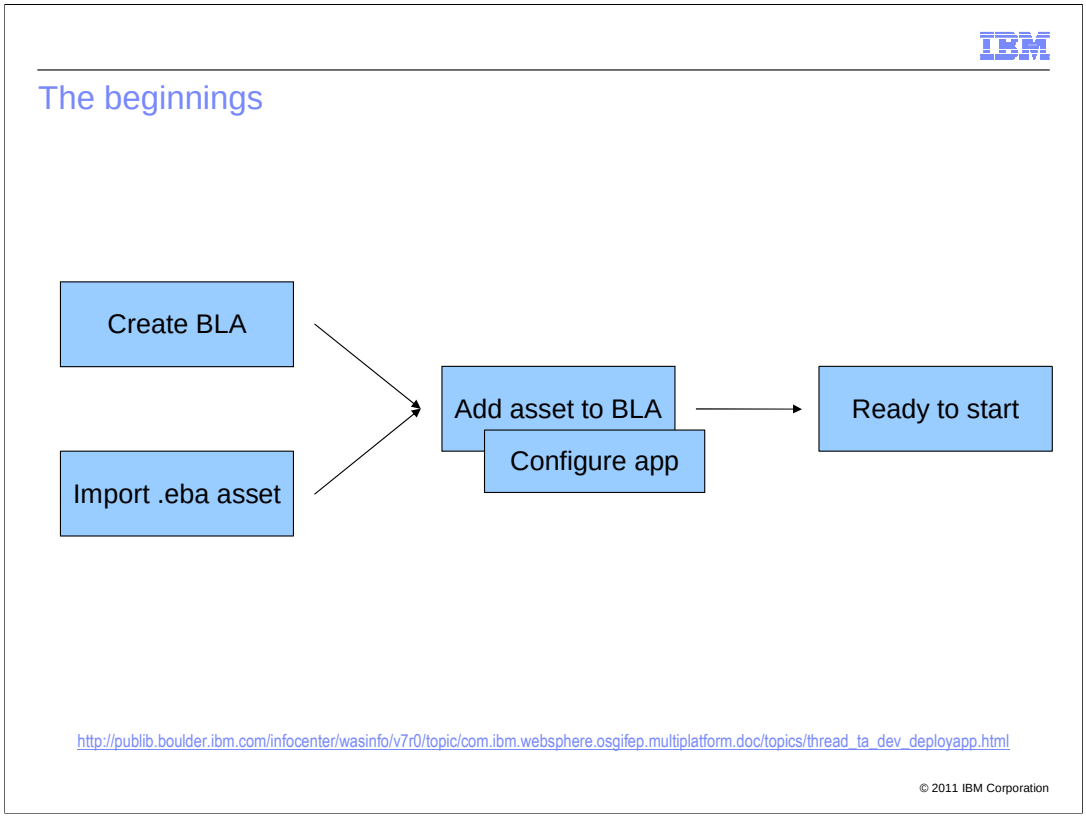
The OSGi applications feature introduces a new type of asset, .eba files, which are recognized and tagged by WebSphere Application Server administrative plug-ins. Furthermore, composition units for .eba assets also get custom extensions from the OSGi feature, in particular start handler that knows how to actually start an OSGi application.

There are several restrictions around the OSGi feature's use of BLA, which are noted here. These are also revisited in the troubleshooting section later on.

Section

A brief history of OSGi administration

This section presents a brief history of OSGi administration.



As noted before the OSGi applications feature uses BLA for most of its administration capabilities. One slight downside of this is that the flow of installing an application is constrained by what is possible in BLA and in particular there is no one-wizard install. Instead there are four steps

Step one is to create a business-level application. This is essentially just an empty hull with a name at that stage.

Step two is to import a .eba asset. During this step the asset is tagged as an eba application, unconverted wars will be converted and finally the application will be resolved. If the application cannot be successfully resolved, then this step will fail.

Step three, after completing the first two steps, the asset can be added to the BLA from the BLA detail panel. This includes going through several wizard steps, in which various configuration options can be set such as deployment targets, web module mappings, context path and virtual host, resource reference mappings and role mappings.

Step 4, when this is complete the application is installed and ready to be started, which is done by starting the BLA.

The information center link at the bottom points to the topic that explains deployment of EBAs in all detail.

Configuration options (1 of 2)

- No post-deployment configuration in version 1
- Configuration steps
  - Composition unit options (BLA)
  - Deployment targets (BLA)
    - Constrained by feature pack requirements (JPA2, SCA)
  - Context roots
  - Virtual hosts
  - Web module resource references
  - Web security role to user/group mapping
  - Blueprint resource references

The top bullet on this slide is very important to note. In version one of the OSGi feature there is no post-deployment configuration what-so-ever. When you add your asset as a composition unit to the BLA that is your one and only chance to configure it. After that point it is a matter of deleting the composition unit and adding it again to reconfigure, which of course means you will incur down time.

Here is a bit more in detail at the flow described in the previous slide. First are the configuration options currently supported for an EBA.

Every .eba asset will at least have two configuration steps, which are inherited from BLA.

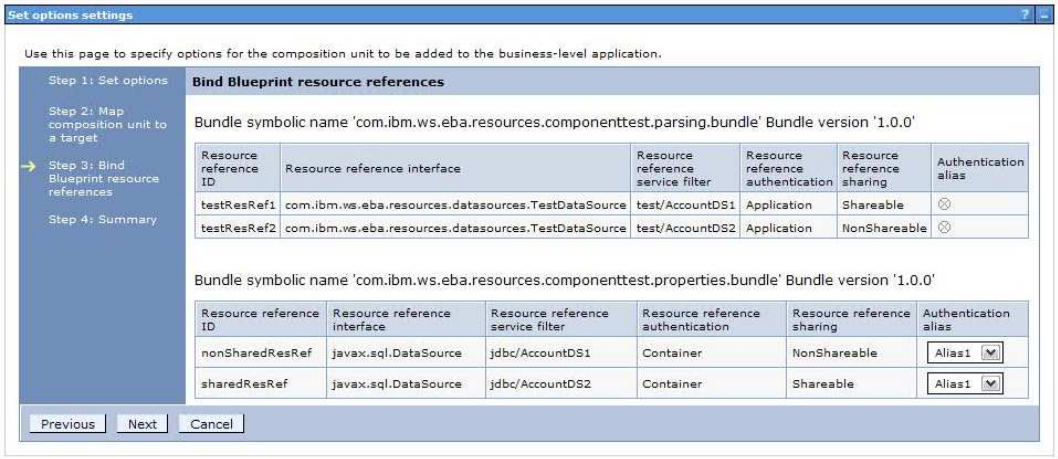
The composition unit options are various BLA options (like description, cu name, starting weight), which for the purpose of OSGi applications can safely be left to the defaults.

To actually run anywhere, a composition unit must be mapped to a server. Note that the selection of servers or clusters is constrained by the application environment requirements. For example, an application that contains bundles that pull in JPA 2 packages can only be deployed to servers or clusters that are augmented with the JPA 2 feature pack, similarly for SCA. Servers or clusters without the required feature packs will not be shown in the list of available deployment targets.

The majority of the OSGi applications configuration options deal with web applications. In essence they replicate the corresponding configurations for traditional JEE web application. The panel layout and options also are adapted from the corresponding JEE configuration panels. In particular, you can configure the context root and virtual host for every web application bundle. In addition for web bundles that have the required configuration web module resource references and web module security can also be configured.

Finally, blueprint resource references, which are very similar to web module resource reference but for blueprint bundles, are about the only really new configuration step.

Configuration options (2 of 2)



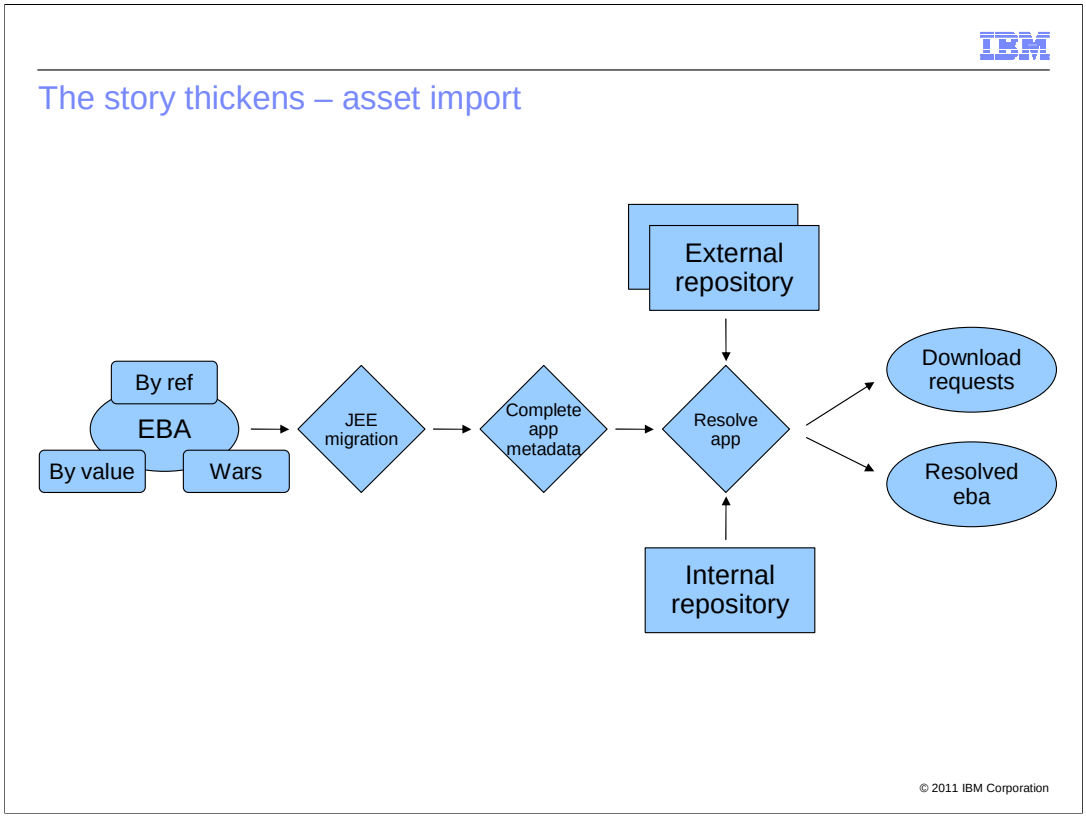
Just as an example on this slide shows the blueprint resource reference panel configuration panel from the import wizard. On this slide two bundles in the application define blueprint resources references, which are displayed.

In traditional JEE applications the configuration would be keyed off the module which has the configuration associated with it.

Note that all configuration of an OSGi application is keyed off the isolated bundle, which will always be bundle symbolic name and version.

You cannot configure provision bundles because you do not know if they will be deployed at run time and it doesn't make sense to put resource references or the like into one of those bundles.

Configuration includes for bundles directly included by value in the .eba file and bundles provisioned dynamically.



In the previous discussion the step of importing an asset might have seemed a relatively trivial operation, but actually there is a fair amount of works going on, especially when the .eba contains by reference bundle, so here is a bit more detail what happens.

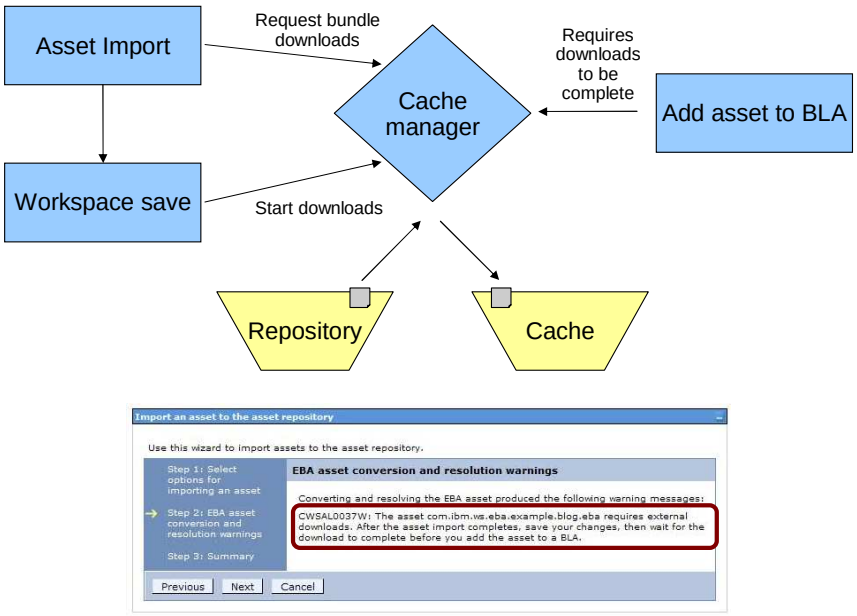
Initially, the contents of the eba are scanned for bundles and some JEE artifacts as described in a previous presentations will be converted. In particular, war files will be converted to web application bundles (WABs) and a was.policy will be converted to permissions.perm file (the equivalent of a was.policy for the OSGi feature pack).

After that the bundles inside the .eba zip are known and if necessary an application manifest can be created, or completed if either none exists or is incomplete.

Next, with a valid application manifest and the bundle content inside the eba, now starts the most important step: resolving the application. In this step the specified bundle content (Application-Content and Use-Bundle headers of the application manifest) is resolved and there transitive dependency closure. It is at this step that bundles that are logically part of the .eba but not physically included in the archive are located in an internal or external bundle repository. Furthermore, unresolved package, service, or bundle dependencies are also resolved from the repositories along with any further dependencies.

At the end of this process you now have a fully resolved eba. The actual deployment information is captured in the deployment manifest. Also, as part of the resolution process we have a list of bundles that need to be downloaded from external sources.

The bundle cache manager



This leads nicely to the discussion of the bundle cache manager. Although the cache manager is effectively hidden in the console, it is important to understand what it does and where it fits in to understand the various restrictions it imposes to the administrative tasks.

A list of bundles to be downloaded is generated during the eba resolution during asset import. These download requests are registered with the bundle cache manager. However, the downloads are not kicked off **until** the asset import changes are saved. If the asset import changes are discarded, then the bundle downloads request are also discarded. Hence, the downloads happen asynchronously and the administrator can continue to use the administration console for unrelated configuration tasks while bundles are downloading.

Furthermore, since downloaded bundles may need configuration during the step of adding an eba asset to a BLA. This operation cannot begin before all downloads related to that asset are complete. So for an installing an eba asset saving the import asset changes is crucial. To underline that the warning message shown below is displayed whenever an eba requires bundles to be downloaded.

Finally, there is the question of how does the bundle cache get to the nodes in a Network Deployment topology? The answer to that is that the bundle cache lives in the configuration tree and fully downloaded bundles are treated as configuration files. So the bundle cache will be synched to any nodes using the standard synchronization mechanisms. Note though that the bundle cache manager does not itself trigger node sync operations. These only happen on the administrator's explicit request or as part of regular auto-sync operations.

Bundle repositories

- Bundle repository
  - Place to contain reusable bundle along with metadata required for resolution
  - Based on Apache Felix OBR and the OSGi bundle repository draft specification (RFC 112)
- Two types
  - External
    - Specified by a URL to the repository descriptor and a name
    - Some open source tooling support (bindex)
  - Internal
    - Contains full jars on the deployment manager system
    - Convenience to get started
    - Extended support for bundle metadata
      - Service provisioning
      - Composite bundle archives

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.osgifep.multiplatform.doc/topics/ta_admin_obr.html

Bundle repositories, mentioned previously, contain reusable bundles along with extracted bundle metadata with at least the dependency information from the bundle manifests to allow resolution without downloading the bundles.

The format of bundle repositories is based around a draft specification (RFC 112) of the OSGi alliance, which most prominently is implemented by Apache Felix.

The OSGi feature supports two types of repositories: external repositories, which are defined in the console and accessed by their URL, and a single internal repository. The internal repository is essentially a convenience for customers to get started with repositories. Whereas for an external repository a customer would have to generate the repository descriptor either with an open source tool such as bindex or by hand, the internal repository is generated by the feature pack runtime from plain bundles. External repositories are indispensable though for sharing bundles between more than one Network Deployment cells since internal repositories are cell-scoped.

Furthermore, the internal repository will fully generate all the metadata required for the OSGi application resolution process. In particular, the open source bindex tool does not require the service provision metadata; neither does it support composite bundle archives.

The link at the bottom of this slide again points to the relevant information center topic.

Composite bundle idiosyncrasies

- All bundles in Composite-BundleContent must be in internal repository
- Bundles are added to the internal repository
- CBAs with by ref content cannot be added if content is not available
- Individual bundles cannot be deleted while CBA is in bundle repository

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.osgifep.multiplatform.doc/topics/ta_manage_cba.html

Composite bundles mentioned on the last slide have several additional restrictions around them.

The idea of a composite bundle is a selection of bundles that resolves together and can be presented at runtime as one coherent unit to the rest of the system. To guard against potential sources of insidious problems, the OSGi applications feature pack restricts composite bundles and their contents to only reside in the internal bundle repository. This allows an easy check whether a CBA is available in its entirety at any given point.

These checks happen on two operations: importing a CBA into the internal repository and deleting bundles from the internal repository.

In order for a CBA to successfully import, all the bundles that make up the CBA must be in the internal repository. Either they are contained, by value, in the CBA archive itself – in this case the bundles are added to the repository – or the bundles are only referenced in the CBA manifest – in this case the bundles must already be in the internal repository.

In order to ensure that a CBA is valid for as long as it is in the internal bundle repository, all the bundles that make up the CBA must be kept in the repository. Hence, any attempts to delete a bundle that is part of a CBA (regardless of whether it was installed by value or by reference) will fail.

Note a customer may choose to generate CBA metadata in an external repository. However, there is no tooling to do this and no claimed support for this scenario.

Updating OSGi applications – fine grained updates (1 of 2)

- ≠ BLA update
- Upgrade / downgrade on the level of individual bundles
  - Within the constraints of the APPLICATION.MF
- Restrictions
  - All bundles must be located in bundle repositories
  - Bundles must be downloaded
    - Three steps:
      - Select updates, save changes
      - Wait for downloads to complete (check on asset detail panel)
      - Restart BLA (applicable updates shown on composition unit detail panel)
  - No new configuration is supported
- http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.osgifep.multiplatform.doc/topics/ta_admin_maint_bundle_ver_console.html

OSGi applications support a fine-grained update process where individual bundles that are part of an application can be updated within the version ranges specified in the application manifest.

Note that this notion of update is completely separate from the BLA update operation, which is **not** supported for OSGi applications. In particular, if a customer wants to change the application manifest itself, then he currently **must** reinstall the OSGi application.

Now updating an OSGi application similar to installing it is a somewhat complex operation. Initially, a user should go to the asset detail panel and configure the updates, as shown on the next slide. Updates can be chosen on a per bundle basis for every bundle in Application-Content or Use-Bundle. The default setting is to move to the latest available version; however individual versions can also be selected. After choosing an option, the runtime will check that the selected version resolve together. If that is not the case an error message will be produced and you will be asked to revisit his selections. Otherwise the new versions are shown in a preview page; these versions can then be committed.

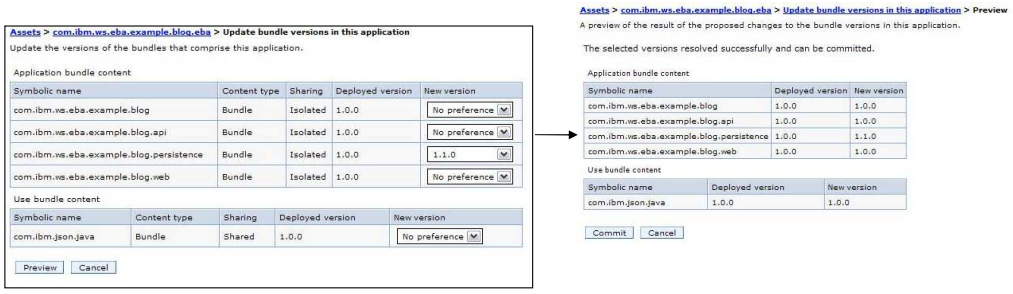
Saving the update changes will trigger the bundle downloads exactly like during the import asset operation. The state can be checked on the asset detail panel. Once all the downloads are complete, the updates will take effect with an application restart.

Note some of the restrictions around updates: the versions to which a bundle can be updated are retrieved from all the installed repositories and the application itself. So to update a bundle to a new version one must first add the newer version to a bundle repository. Also, in the whole update process there is no provision for reconfiguring bundles. So for example a web application bundle would have the same configuration (created during the initial installation) pre and post-update. In particular, this means a web application bundle should not use new resource references or new security roles.

Updating OSGi applications – fine grained updates (2 of 2)

Choose updates

Preview changes



This slide shows the two central panels of the OSGi update scenario.

The panel on the left shows an administrator's options for update.

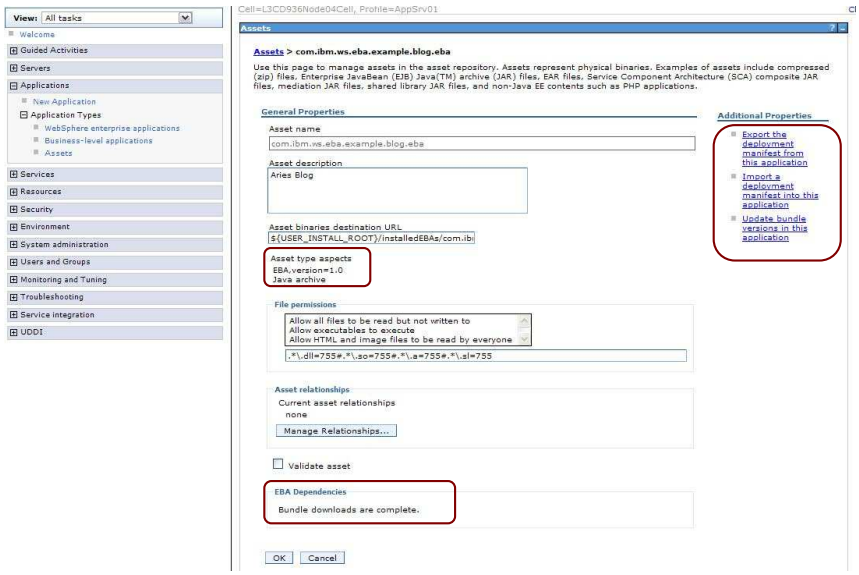
All the named bundles in the application, Application-Content and Use-Bundle but not provisioned bundles, are shown with their current versions. For each of them, a drop down box will be populated with all the available versions plus "No preference".

No preference in this instance means go up to the highest version of the bundle that resolves together with the other selections. This will therefore not always result in the highest available version, especially if that is in conflict with other explicit selections.

The panel on the right shows the results of the selection on the left, which was valid. For every bundle the old and new version are listed.

Note that despite setting "No preference" for most bundles only one has actually changed. This is because in the example shown only one bundle had a newer version available.

EBA asset details



http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.osgifep.multiplatform.doc/topics/ta_depmf_export.html
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.osgifep.multiplatform.doc/topics/ta_depmf_import.html

The asset detail panel contains several important and useful pieces of functionality and information.

First of all the asset should always show the correct type “EBA,version=1.0”, which indicates that the OSGi application feature pack is installed and has correctly recognized the asset.

Secondly, the bundle status panel allows a user to check whether downloads are complete after an import or an update operation.

Finally, the side panel contains links to additional administration options like fine-grained updates as discussed before and the last remaining configuration option to be discussed: importing and exporting assets.

This is a mechanism that allows a customer to take an application in its completely resolved state from a test system to a production system. Hence, this path eliminates any worries of whether resolution on the production system might provision a different set of bundles than were previously tested.

The steps to achieve this are; export DEPLOYMENT.MF from the test system, import the .eba asset and import the DEPLOYMENT.MF into the .eba asset previously created.

Section

Troubleshooting

This section will provide some troubleshooting tips.

Known limitations

- Not supported
  - BLA asset update operation
  - Post deployment configuration
  - Multiple copies of an EBA
  - EBA asset in more than one BLA

Probably the most important aspect of early troubleshooting of the OSGi feature pack administrative is going to be around the limitations of the OSGi applications in the BLA framework.

As noted on the slide, OSGi applications support only a subset of all that BLA has to offer. There are some important limitations.

The update operation on an .eba asset (this is not the fine-grained update discussed above) but the standard “replace the binary” update that BLA provides is not supported.

Furthermore, none of the OSGi application configuration can be reconfigured post deployment. Hence, for changing configuration the EBA composition unit has to be deleted and the EBA asset re-added to the BLA.

Finally, OSGi applications can neither exist in multiple versions in a cell nor can a single version of an OSGi application be deployed into multiple BLAs.

Troubleshooting

- Trace
  - BLA=all: Aries.eba.bla=all: Aries.eba.admin=all: Aries.app.utils=all: Aries.eba.provisioning=all
- Simple checks
  - Is an .eba recognized as an OSGi application
    - Check for asset type aspect “EBA,version=1.0” on the asset detail panel
  - Are the bundle downloads working
    - Check on the asset detail panel for the bundle download status
    - Use the bundle cache manager mbean

Many of the administrative related problems previously (system test and development) encountered have been around the BLA framework and its use in the OSGi feature pack. Hence, it is crucial also to collect BLA trace as shown in the example trace string on this slide.

One of the simplest means to check the OSGi applications administration is by looking at these two items.

First, the type aspect for OSGi applications on the asset detail panel should clearly show that the feature pack is operating properly and has recognized the .eba asset. If this type aspect is not present, this points either to a configuration error of the feature pack or a severe problem during the import steps of an .eba asset.

Secondly, also on the asset detail panel is the bundle download status panel. Many bundle cache related problems, in particular failed downloads, should be highlighted on this panel (as failed downloads). For further delving into such a problem the bundle cache manager mbean can be used. There is another module on the capabilities and use of the bundle cache manager mbean.

Summary

- Overview
- Survey of administrative capabilities
- Troubleshooting

In Summary, this module has provided an overview of administering OSGi applications in the WebSphere Application Server administration console. It has provided illustrations and descriptions of the actions one can take in the console and has provided some troubleshooting hints.

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_wasosgijpafep_OSGi_admin.ppt

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

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. in the United States, other countries, or both.

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.