JEUS Applications & Deployment Guide

JEUS 9

TMAXSOFT

owners.

The names of companies, systems, and products mentioned in this manual may not necessarily be indicated with a trademark symbol (^{TM}, ®).

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses: APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

Detailed Information related to the license can be found in the following directory: ${INSTALL_PATH}/license/oss_licenses

Document History

Product Version	Guide Version	Date	Remarks
JEUS 9	3.1.2	2025-03-24	-
JEUS 9	3.1.1	2024-12-24	-

Contents

Glossary

Console

A command-line interface (terminal window), which is a contrary concept to GUI interface.

DD

An abbreviation of deployment descriptor.

jeus-application-dd.xml

JEUS Application DD file

domain.xml

JEUS WAS configuration file

1. Application Management in a Domain Environment

This chapter describes how to manage and deploy applications in a domain environment. It also explains how to deploy applications on a managed server (MS).

1.1. Application Management

The Master Server is a server that manages domains. Since applications need to be managed at the domain-level, the Master Server manages all applications existing in the domain.

Executing deploy and search for applications can only be performed from the Master Server. When the Master Server starts up, a service that manages applications starts. Through this service, applications can be deployed to servers or clusters. Not only deploy, but also any other application-related commands can be executed only from the Master Server. If a MS goes into the INDEPENDENT state due to an error in the Master Server, the deploy command cannot be executed. But you can use the console tool (jeusadmin) to connect to the MS and get the application information.

> Viewing the application information by directly connecting to a MS is only possible when the MS is in the INDEPENDENT state. If a MS is in the INDEPENDENT state, it cannot access to the Master Server or the latter cannot manage the MS. In this case therefore, you need to view the application information by connecting directly to the MS.

To deploy and run an application, the application must be installed on a domain. Installing an application means installing application files on the Master Server, and deploying an application means making the application available for use.

- **install**

  Application files are uploaded to the Master Server. Only the applications installed on the Master Server can be deployed.

  Installing an application involves the following:

  - Uploading the application file to APPLICATION_INSTALL_HOME in the domain by using a JEUS-provided tool. APPLICATION_INSTALL_HOME is the repository of applications installed in the domain. This repository cannot be modified nor accessed manually. Any attempts to dynamically delete an installed application or add a new application to the repository will not succeed. Such changes to the repository can only be applied after the Master Server is restarted, but this is strongly discouraged.

  - Placing an application manually in a user-specified directory. The user can add this directory as an application repository to manually upload application files. You can add or delete an application repository in the domain by using the console tool commands **add-application-**

repository and **remove-application-repository**.

- **deploy**

  Applications are deployed to a server and ready to run.

  When the user sends the command to deploy an application to the Master Server, the Master Server deploys the application to target servers or clusters using the two-phase deployment protocol. For more information about the two-phase deployment, refer to 2 Phase Deployment.

1.1.1. 2 Phase Deployment

Application files are distributed to each target server, and preparation tasks are performed. After the jobs are completed successfully, the application is ready to start. These jobs are performed in two steps on the Master Server. All deployment-related commands are executed on the Master Server, and the Master Server sends the commands to each target server. When a user sends a deploy request, the deploy command is considered successful only when all target servers have successfully completed the deployment.

The request is handled in two steps, **Distribute** and **Start**. If the first step (distribution) fails, the deployment request will fail, and the application will be undeployed from all the servers.

- If the first step is successful, deployment is potentially successful, since the application is successfully distributed and verified. However, the application cannot be started yet.
- If the second step is successful, the application is ready to start. When there is a failed server, this step is retried on that server to guarantee that the application can run on it.

You can execute these two steps individually using the 'distribute' and 'start' commands, instead of using a single 'deploy' command.

Step 1 (Distribution Step)

In the application **distribution step** in JEUS, an application is distributed to target servers or clusters, and then verified. In this step, application files are distributed to each server, and preparation and verification tasks are executed.

When all jobs are completed successfully, an application is ready to run. If the application is an EJB module, a service port is opened, and if it is a web module, its context gets connected to the web listener's context. However, the application is in the DISTRIBUTED state, where the application cannot be started.

If an application in the DISTRIBUTED state receives a request to run, an exception or an error will occur. If the application is an EJB module, the exception stating that the Bean does not exist will occur, and if the application is a web module, the "503 Service Unavailable" error will occur.

A Client Exception for When an EJB Module in DISTRIBUTED State Receives a Request

```
[2016.08.10 15:28:35][1] [t-1] [JNDI.Context-0073] exception occurred during JNDI operation
<<__Exception__>>
```

```
javax.naming.NamingException: jakarta.ejb.EJBException: java.rmi.RemoteException: EJB object is not
 read at
jeus.ejb.client.BusinessObjectFactory.getObjectInstance(BusinessObjectFactory.java:89)
   at javax.naming.spi.NamingManager.getObjectInstance(NamingManager.java:304)
   at jeus.jndi.JNSContext.lookupInternal(JNSContext.java:594)
   at jeus.jndi.JNSContext.lookup(JNSContext.java:549)
   at jeus.jndi.JNSContext.lookup(JNSContext.java:538)
   at jeus.jndi.JEUSFailoverContext.lookup(JEUSFailoverContext.java:314)
   at javax.naming.InitialContext.lookup(InitialContext.java:392)
   at test.HelloTest.testHelloBean(HelloTest.java:29)
   ......
Caused by: java.rmi.RemoteException: EJB object is not ready at
jeus.ejb.container.RemoteInvocationManagerImpl.beforeInvoke(RemoteInvocationManagerImpl.java:72)
   at jeus.ejb.baseimpl.RemoteInvokerServer.preInvoke(RemoteInvokerServer.java:118)
   at jeus.ejb.baseimpl.RemoteInvokerServer.invoke(RemoteInvokerServer.java:97)
   at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
   at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
   at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
   at java.lang.reflect.Method.invoke(Method.java:597)
   at sun.rmi.server.UnicastServerRef.dispatch(UnicastServerRef.java:305)
   at sun.rmi.transport.Transport$1.run(Transport.java:159)
   at java.security.AccessController.doPrivileged(Native Method)
   at sun.rmi.transport.Transport.serviceCall(Transport.java:155)
   at sun.rmi.transport.tcp.TCPTransport.handleMessages(TCPTransport.java:535)
   at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run0(TCPTransport.java:790)
   at sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run(TCPTransport.java:649)
   at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886)
   at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)
   at java.lang.Thread.run(Thread.java:662)
```

HTTP Status : 503 - 503 Service Unavailable

Description Worker (http1-2): An error occurred while processing the request.

An Error Which Occurs When a Web Module in DISTRIBUTED Receives a Request

When an application is successfully distributed to all target servers or clusters, this step is completed successfully. If the distribution job fails even on a single server, the user's **deploy** command will fail. In such a case, the **undeploy** command will be sent to all successful servers, and all jobs in this step will be rolled back. If some of target servers are in a failed or shutdown state, this step will also fail. At this time, the user needs to run the deployment command again on all other successful target servers.

When the distribution target is a cluster, even if some of the servers in the cluster is in a failed or shutdown state, the distribution step can be successful. If all other servers, besides the failed servers, are successful, this step is also considered successful.

Step 2 (Start Step)

An application, which is already distributed on the target servers or clusters, is started in this step, which is referred as the application **start step** in JEUS. In this step, the application becomes available for service.

Unlike the distribution step, the start step is considered successful when even just a single server out of all target servers, or in an entire cluster, is successful. This is because, with the successful completion of the distribution step, the application is already verified and ready to run.

If the start step fails, it indicates that a server is in a failed state temporarily. Once the server's status returns to normal, the start step has the potential to succeed and become available for service. In the event of a server failure during this step, the Master Server retries the operation on the failed server up to 10 times, with a 5-second interval between attempts, using different threads.

If all retries fail, the server must be manually restored, because the server will remain in the failed state. If this step is retried for an application after the server is restored, it will execute successfully on the server.

1.1.2. Application ID

From JEUS 7 onwards, an ID is assigned to each application to allow a domain to identify and manage the application. An application ID must be unique within a domain, because it is used to identify the application in the domain. It is recommended to use alphanumeric characters for application IDs.

You can specify the ID when installing an application. If no ID is specified during installation, the name of the application file will be used to create an ID. For example, if you did not specify the ID for an application file named 'examples.ear' during the installation, 'examples_ear' will become the application ID.

The ID of an application is needed when executing commands to deploy/undeploy or find the application. If the application is located within a separate repository in the domain, the application file name is used as the ID.

When the Master Server receives an application file, it creates a directory using the application's ID in the application's INSTALL_HOME directory, which is located under the domain directory. It then locates the file in the created directory.

When an application is installed, it is placed in the following directory. For detailed information about the INSTALL_HOME directory, see INSTALL_HOME Directory in Domain.

```
INSTALL_HOME/<APPLICATION_ID>/<APPLICATION_FILE>
```

When a server receives or decompresses an application file, it locates the file in the APPLICATION_ID directory, under the DEPLOYED subdirectory.

From Java EE 6, an application name can be defined in the standard DD. A name is

required to start an application. Note that the name is different from the application ID.

An application name must be unique on a server, where the application is running, but an application ID must be unique within a domain. If an application name is not specified, the application's file name without the extension becomes the application name. For more information about how to set an application name, refer to "Jakarta EE 9 Platform Specification".

1.1.3. Application Status

Because a domain controls commands over applications, the application status shows the statuses of all the target servers. Each server shows statuses of applications, which are running on it.

Status of Applications in a Domain

The following illustrates the life cycle of an application in a domain. In other words, it is the status progress of an application in the Master Server.

```
INSTALLED → DISTRIBUTING → DISTRIBUTED → STARTING → RUNNING
```
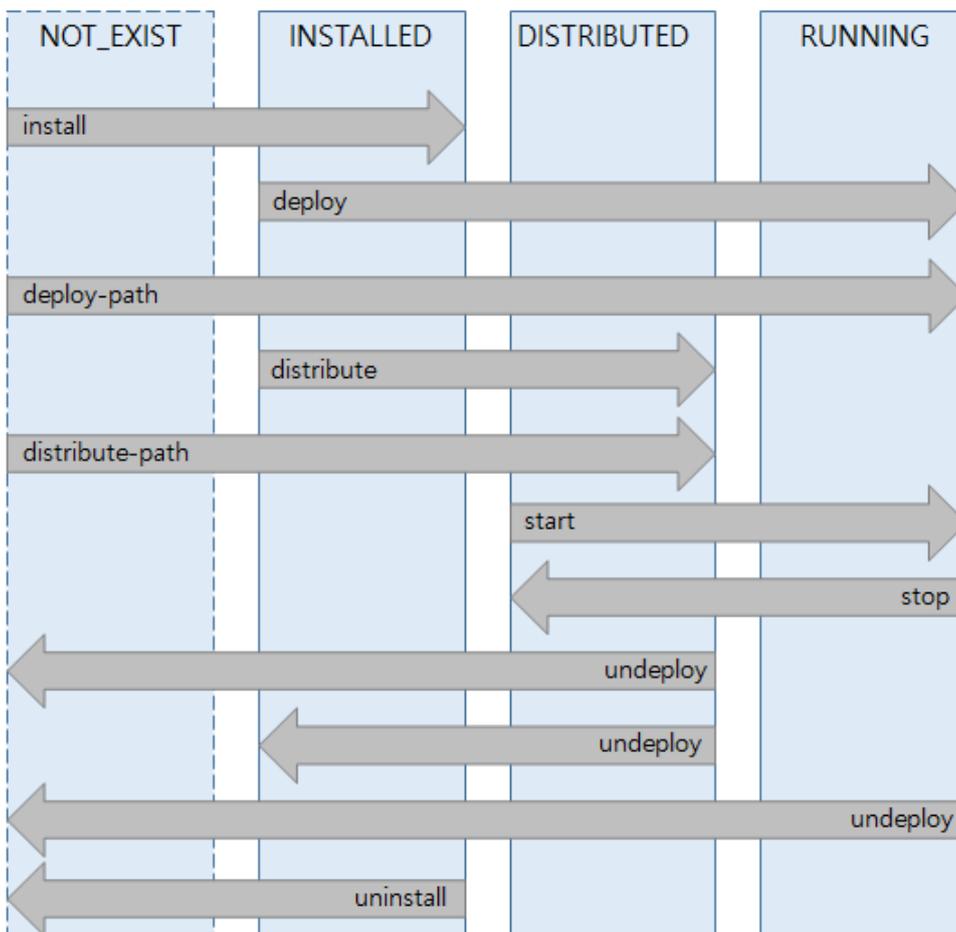

The Life Cycle of an Application in a Domain

The following describes each status.

Status	Description
INSTALLED	The application file is uploaded to a domain. In this status, a command to **deploy** or **distribute** the application can be run by specifying the targets.
DISTRIBUTED	The application is successfully distributed. At this point, even if all the target servers shut down, the application's status is displayed as DISTRIBUTED.
RUNNING	The application is running. If the application is running on even just a single server out of all target servers, the application's status is RUNNING.

> In a domain, another status titled **DEPLOYED** exists other than those three states described above.
>
> If all the target deployment servers of an application are not in the RUNNING state, the status of the application is displayed as DEPLOYED. This means that the application is not running on any server.

Status of Applications in a Server

The following illustrates the life cycle of an application in a server. In other words, it is the status progress of an application within a server.

```
DISTRIBUTING → DISTRIBUTED → STARTING → RUNNING
```
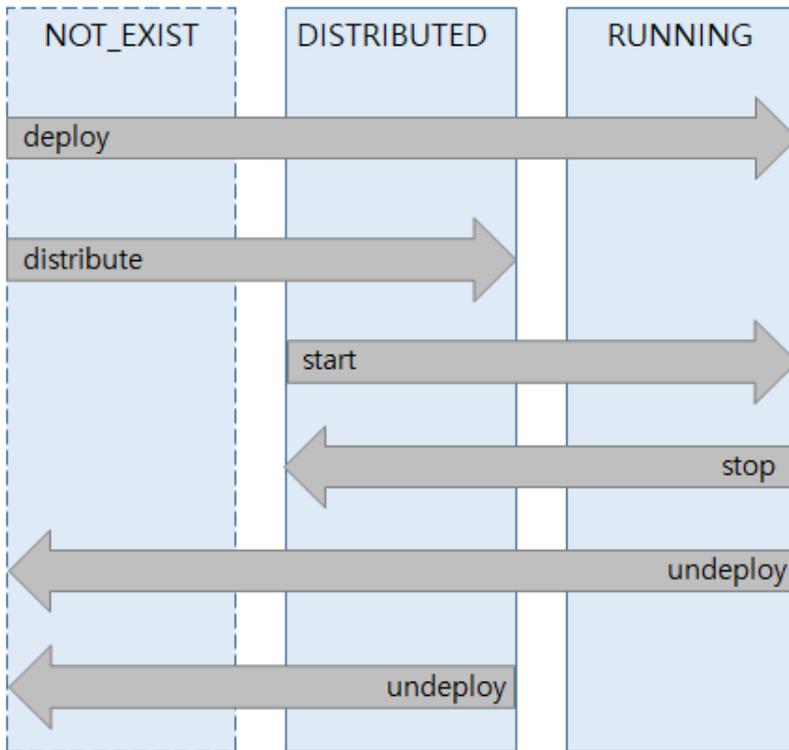
The Life Cycle of an Application in a Server

The following describes each status.

Status	Description
DISTRIBUTED	The **distribute** command from the Master Server is executed successfully.
	The application is ready to run, and can now be started with the **start** command.
RUNNING	The application is running on the server.

1.1.4. Directory Structure to Manage Applications

The directory structure used to deploy an application is as follows:

INSTALL_HOME Directory in Domain

When an application is installed on a domain, it is placed in the APPLICATION_INSTALL_HOME directory.

```
APPLICATION_INSTALL_HOME
    |--application_id
        |--[J]application file(.jar, .war, .ear, .rar)

* Legend
- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
```

```
    - [V] : java source file
    - [DD] : deployment descriptor
```

APPLICATION_INSTALL_HOME

> This directory refers to the DOMAIN_HOME/.applications directory and is called **INSTALL_HOME**, in short. It is only created on a machine where the Master Server resides. A directory is created under this directory using the application ID, assigned when the application was installed, as its name. Actual application files are placed in the newly created directory.

Application Repository Managed by a Domain

The following shows an application repository that is added to a domain.

```
application repository
    |--application file (exploded module)
    |--[J]application file(.jar, .war, .ear, .rar)

* Legend
- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor
```

An application repository includes both applications in the form of a directory and an archive. When applications are included in an application repository by a user, an application ID cannot be specified by the user, and is always created using the file name. For information about how to add and delete an application repository, see Adding, Deleting, and Searching Application Repositories.

Image Directory of an Application Deployed on a Server

A server receives an application from the Master Server and places it in the following path to deploy it.

```
SERVER_HOME/.workspace/deployed
```

The path is called **DEPLOYED_HOME**. A directory is created under the DEPLOYED_HOME directory using the application ID as its name. Received application files are placed in the new directory.

When deployed, an archived application is decompressed in a directory called the application's **deployment image directory**. The directory is named with the application file's name. For example, the name of the image directory where myApp.ear is decompressed is 'myApp_ear___'.

If the extension of an application file is ear, the file's modules also need to be decompressed under the ear image directory. The directory where the modules are decompressed is named in the same

way as an image directory. For example, the name of the directory where ejb.jar is decompressed is
'ejb_jar'.

The following illustrates the image directory structure of an ear application deployed on a server.

```
DEPLOYED_HOME
    |--_generated_
    |--myApp
        |--myApp_ear__
            |--appclient_jar__
            |--ejb_jar__
            |--lib
            |--META-INF
            |--web_war__
            |--[J]appclient.jar
            |--[J]ejb.jar
            |--[J]web.war

* Legend
- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor
```

gen Directory of an Application

In some cases, when an application is deployed on a server, the necessary files for the application are
created.

For EJB or web applications, those files need to be created in a directory under the '_generated_'
directory, in the DEPLOYED_HOME directory. The directory is created using an application ID, so that
each application has its own directory. The directory is called an application's **gen directory**.

The following files are created in the gen directory of each application.

- For EJB applications, stub and skel files are created when EJB 2.x type Bean is used with the stub
  method, and not with the dynamic proxy method.

- For web applications, Java source is generated for JSP, and stub files for embeddable EJB. An EJB
  module embedded in a web application is called an embeddable EJB in Java EE6.

- The file '_appdat.ser' is created. It is a serializable file containing the information about the time
  when the applications are deployed. When a server is shut down and restarted, if the application
  has not been changed, this file prevents recreation of deployment image files and already
  generated files for faster booting.

The following illustrates the gen directory structure for an ear application deployed on a server.

```
DEPLOYED_HOME
```

```
    |--_generated_
    |--myApp
        |--ejb_jar
        |     |--classes
        |--web_war
            |--__embedded_ejb
            |       |--classes
            |--__jspwork
            |--__ws
            |--_appdat.ser

 * Legend
 - [01]: binary or executable file
 - [X] : XML document
 - [J] : JAR file
 - [T] : Text file
 - [C] : Class file
 - [V] : java source file
 - [DD] : deployment descriptor
```

1.1.5. Application Deployment Targets

To deploy an application, specify the target servers where the application will run on. The targets can be one or more servers and/or clusters. You can also add a virtual host to deploy and start the web module on a specific host.

Server

Since a server is the minimum unit for running an application, it can be a target where the application is deployed on. When the deploy command is sent to multiple servers, if the deployment fails on one or more servers or one or more servers are shut down, the distribution step will fail. If the distribution step fails, take an appropriate action for the servers and run the deploy command again, or exclude the failed servers and run the deploy command again.

When a server is in the STANDBY or SUSPENDED state, it can't start an application, because this means that the server is not running normally.

When a server is in the RUNNING state after the server has restarted or resumed, it can start an application. For detailed information about server statuses, see "Lifecycle of Managed Server" in JEUS Server Guide.

If a user runs only the distribute command, not the deploy command, an application will not start. Even after the server restarts, the status needs to be maintained. Therefore, the Master Server records the status of the application, in the '.deployInfo.ser' file under the INSTALL_HOME directory. When the server restarts, a status notification is sent to the user in order to continue processing the previously started jobs.

If a user manually modifies a configuration file, the status of an application may

not be maintained. Therefore, it is recommended to use JEUS-provided management tools (e.g. jeusadmin) when modifying a configuration file.

Cluster

If an application's target is a cluster, the application can only be deployed on the cluster, and not on a particular server within the cluster. If an application fails to be distributed on even a single server in a cluster, the application will fail to be deployed on the cluster. However, unlike using a list of servers as a target for deployment, an application can be successfully deployed on the cluster even if some servers in a cluster are not alive. If the application is successfully deployed on all live servers in the cluster, it is successfully deployed on the cluster.

A cluster represents a group to operate a service, not merely a list of individual servers. If an application is successfully deployed on all live servers in a cluster, there is high probability that the application can be successfully deployed on servers which are not alive. The application must be deployed on the servers which are not alive, when the servers become live, restarted, restored, or dynamically added to the cluster.

An application cannot be deployed on a specific server that is included in a cluster. If attempted, the following warning message will be displayed for the user to run the undeploy or the remove-application-target command:

A Log Message Displayed When a Target is a Server Included in a Cluster

```
WARNING: The server[server1] is part of the cluster[cluster1], so this application[examples] can only
be deployed to the cluster.
Deploy again this application to the cluster target.
```

Since it is not possible to execute the deploy command while targeting an individual server within a cluster, the previous warning will not occur when using the deploy command. **It will occur when the user manually modifies the configuration file**.

If an independently running server is dynamically added to a cluster, applications currently running on the server will be undeployed, and the server will be removed from the target list. Also, applications which need to be deployed on the cluster will be dynamically deployed, since the user probably intends to use the applications in the cluster.

However, it is not recommended to dynamically add an existing server to a cluster, even if it is intended by the user. This may cause a problem for services already running on the server. Therefore, to expand a cluster, add the new server, and then add the server as a member of the cluster.

Virtual Host

Web engine allows a web context to run on a specific host registered as a virtual host. A virtual host cannot be used alone, but can be used along with server or cluster targets. If there is no virtual host, an application is deployed on the default virtual host.

For more information about virtual hosts, see "Virtual Hosts" For more information about the deploy command, see Deploying Applications.

1.2. Deployment to Managed Servers

This section describes how to deploy an application or a module on a managed server (MS).

1.2.1. Run-time Deploy

Run-time deployment is deploying an application or a module on a running server. It can be executed by accessing the Master Server using JEUS-provided tools.

In JEUS 7 or later versions, runtime deployment indicates **permanent deployment**. Permanent deployment means that if an application is successfully distributed with the deploy or distribute command, the details are recorded in the domain.xml file to save the application information. When the Master Server or MS reboots, it reads information about the application, and the application is deployed via boot-time deployment. The domain.xml is the configuration file for JEUS domains.

Because deployment can be executed only by using the Master Server, only the Master Server can write information about the applications in domain.xml. MS receives application settings from the Master Server when it boots, downloads application files registered in the settings, and executes boot-time deployment according to the domain.xml file. For information about how to execute runtime deployment, see Deploying Applications.

1.2.2. Boot-time Deploy

Boot-time deployment is deploying an application registered in domain.xml when MS boots. Because permanent deployment is the default setting for JEUS 7 or later versions, applications, which were deployed once already, are deployed again when MS boots.

Boot-time Deployment in the DEPENDENT State

When MS boots, it receives the file, domain.xml, from the Master Server. If there are applications registered in the file, MS checks to see if the applications need to be deployed on itself. The applications are not deployed individually on the MS. All applications are distributed first, and if the distribution is successful, the start step is executed.

> If an individual server in a cluster is a target to deploy an application, boot-time deployment will not be executed for the application. An application can be deployed on a server in a cluster only at the entire cluster level.

> This is guaranteed when deployment is executed with a JEUS tool, but not for when the domain.xml file is manually modified. Therefore, it is recommended to use JEUS tools for deployment.

The execution of the distribution step during boot-time is similar as during run-time. First, the server receives application files from the Master Server. If the application files already exist on the server, these files and the application files installed on the Master Server are compared to see if the application files need to be sent to the server. Then, the server decompresses the application files, loads the relevant classes, and executes the distribution step to configure the environment for the applications to run in.

If one or more applications fail to be distributed when a server boots, the server goes into the STANDBY state and stops the booting process.

When the applications are successfully distributed during boot-time deployment, MS checks with the Master Server to see if the application can be started.

If the applications are in the DEPLOYED or RUNNING states in the Master Server, they are ready to start. If MS fails to get the status of the application, it does not execute the start step. At this point, the MS stops booting and goes into the STANDBY state.

To start an application, regardless of the status of the application in the Master Server, run the start-server command with the -force option to force the server into the RUNNING state.

> If an application is in the DISTRIBUTED state in the Master Server, only the distribution step for the application can be executed on the server. To maintain the status of the application when the Master Server restarts, the Master Server records the status in a separate file.

If an application stops and starts when a server is in the SUSPENDED state, the application's status needs to be maintained for when the server resumes again and goes into the RUNNING state. At this point, the server does not retrieve the status information from the Master Server again. Even if the server is in the SUSPENDED state, the server can get the start and stop commands from the Master Server. The server remembers the commands and is able to maintain the application status when it resumes, even if the application cannot execute the commands.

> If a server which is in the SUSPENDED state receives the stop command for an application, ApplicationAlreadyStoppedException occurs, because all the applications in the server have already stopped running. the Master Server does not treat this exception as a failure of stopping an application.
>
> If a server which is in the SUSPENDED state receives the start command for an application, it cannot execute the command. If the application is in the DISTRIBUTED state when the server goes into the SUSPENDED state, it needs to restart when the server resumes.

Boot-time Deployment in the INDEPENDENT State

When a server boots in the INDEPENDENT state, it is not managed by the Master Server. Because the server cannot access the Master Server, it cannot receive application files from the Master Server. Therefore, if the previously received application files do not exist on the server or are damaged, boot-time deployment cannot be executed.

When a server boots in the INDEPENDENT state, deployment can be executed only when the server is able to access application files. A server can access the application files in the following cases:

- The server can access the application files installed on the Master Server, because the server and Master Server exist on the same machine.
- The server can access the application files, because the files exist on a network-attached server (NAS).
- There remains a decompressed deployed image of the files.

If none of these conditions are met, the application deployment fails since no files for the application to deploy can be found. At this time, the server goes into the STANDBY state and stops booting.

When a server boots in the INDENDENT state, it cannot retrieve a list of applications, which only needs to be distributed, from the Master Server. Therefore, the server starts all applications which have been successfully distributed.

If a server booting in the INDEPENDENT state goes into the DEPENDENT state by being connected to the Master Server, it tries to redistribute all applications that failed to distribute according to the specified options. If all the applications are successfully distributed, the server will get the list of applications, which need to be started from the Master Server, and starts the applications in the same way as during the boot-time deployment.

If there is a running application that only needs to be distributed, the server stops the application. However, if a server goes into the DEPENDENT state during booting because the Master Server is restored, the server does not handle these tasks.

1.2.3. Application Synchronization

One of the major roles of the Master Server is synchronizing the application files. The application files in the Master Server and MS need to be synchronized when:

- Runtime deployment is executed.

  MS receives the application files to be deployed from the Master Server.

- Boot-time deployment is executed.

  If the application files to be deployed in MS are modified, MS will receive the updated application files from the Master Server.

- The status of MS is changed from INDEPENDENT to DEPENDENT.

If the status of MS is changed from INDEPENDENT to DEPENDENT, MS is managed by the Master Server. MS synchronizes the applications only when the **'enable-to-resynchronize-applications'** is set to true in the domain configuration. By default, 'enable-to-resynchronize-applications' is set to false, and applications are not synchronized. If the option is set to true, the application files that failed to be deployed are sent to the MS. Any successfully deployed application files managed by the Master Server that have been modified are also sent to the MS.

The Master Server synchronizes not only the application files, but also the statuses of applications in the domain.

When boot-time deployment is executed, if an application is in the DISTRIBUTED status in the Master Server, the application will be in the DISTRIBUTED status in MS. To start this application, the user needs to run the **start-application** command. If MS goes into the DEPENDENT state from the INDEPENDENT state, application statuses as well as the application files are synchronized in MS. The statuses of applications that have become DEPENDENT in MS are changed according to the statuses of applications managed by the Master Server.

The statuses of all the applications in MS, in the INDEPENDENT state, need to be in the RUNNING state for the applications to run. If the MS goes into the DEPENDENT state, the statuses of the applications in MS will be changed according to those in the Master Server.

If the status of an application is DISTRIBUTED in the Master Server, the application in MS will be stopped and changed to the DISTRIBUTED state. If MS has an application which is already undeployed in the Master Server, the application will also be undeployed in MS.

1.3. Boot-time Auto Deployment

When booting a server, a specified path is explored to automatically deploy the application files located in that path. Unlike manually deployed applications, automatically deployed applications feature the following characteristics:

- The application ID is automatically generated from the file name, excluding the file extension. If the ID is already in use by another application, an error occurs during server boot-up.

- The deployed application information is not stored in domain.xml. Therefore, all behaviors of automatically deployed applications are effective only during runtime.

- Application files are not centrally managed by the Master Server. Therefore, no synchronization occurs.

- Automatically deployed applications always target individual servers, not clusters. Consequently, even if a target server belongs to a cluster, the application will still target that specific server. This is because application files are not centrally managed, and, as a result, it is difficult to perform synchronization between different servers in the cluster.

- Unlike manually deployed applications, automatically deployed applications are considered undeployed when all target servers are shut down, leading to the deletion of their associated information.

You can configure the system properties on the JVM Options of individual servers to specify whether

to use the auto-deployment feature and the location of applications.

- Use jeus.server.enable.auto-deploy to specify whether to use the auto-deployment feature. The default value is true (enable).

- Use jeus.server.auto-deploy.dir to specify the location of applications to be deployed automatically. The default value is DOMAIN_HOME/auto-deploy.

1.4. Auto Redeploy

Up to JEUS 6, specified directories or applications have been checked periodically. When there are changes, corresponding applications are automatically deployed. Applications do not need to be manually deployed with this function. This is beneficial for developing and servicing frequently updated applications. **JEUS 7 or later versions no longer support the automatic deployment feature provided by JEUS 6 or earlier versions. Instead, it provides a redeployment function that automatically redeploys applications by detecting changes made to the applications in the domain.**

Since the Master Server centrally manages applications, the automatic deployment function is no longer supported. The entire domain cannot be targeted to deploy applications, and exploded modules cannot be transferred to a separate machine, which is why automatic deployment is no longer provided. Instead, applications can be automatically redeployed, if you can manually set a cycle in place that checks for the application target and automatic redeployment, when the application is deployed for the first time. In this way, changes to application files can be detected for automatic redeployment.

Application files with automatic redeployment enabled are checked for any changes at the specified interval, using the following methods:

- For archive modules, changes are detected using the last modified time of the archive file.

- For exploded modules, changes are detected using the last modified time of the standard deployment descriptor (DD) file.

The Master Server sends the redeploy command to the target servers. Automatic redeployment and general redeployment are executed in the same way. When redeployment is successful on all servers, the Master Server regards the redeployment as a success.

When application files are changed, redeployment may not be executed immediately, because the changes are checked periodically. The default interval for the check is 10 seconds.

1.5. Deployment for Directory Type Applications

An application must be packaged into a file of an appropriate type according to the Jakarta EE specification. However, a non-packaged application can also be deployed to streamline the

development process. In JEUS, this directory type application is called an **exploded module**.

An exploded module can be deployed on the servers or clusters that reside on the same machine as the Master Server or on a different machine that is in an accessible location like Network Attached Storage (NAS). A module on a different machine is not installed under the INSTALL_HOME directory in the Master Server through the **install-application** command. It must be deployed by specifying the parent directory of the application as **'Application Repository'**, or by using the deploy command with the -path option. When deploying an application of this type with the deploy command, the application ID can also be given as an option. Otherwise, the application file name becomes the application ID.

> JEUS does not perform the synchronization or management of applications that were deployed in directory mode. Those directories must be managed by a user.

1.6. Application Repositories

Since the Master Server manages all applications running on a domain, the application files need to be installed on the Master Server before the applications are deployed. However, applications can be deployed without installing the application files when an application repository is added.

1.6.1. Adding, Deleting, and Searching Application Repositories

The applications installed on the Master Server are gathered and managed in a directory. The applications cannot be manually located in this directory. Multiple directories can be set as repositories, which are regarded as application repositories. To add, delete, and find application repositories, use the JEUS tools.

Jakarta EE applications, both archive modules and exploded modules, can be placed in a directory specified as an application repository in a domain.

The application files cannot be added to or deleted from an application repository, registered in a domain, with the install-application and uninstall-application commands, respectively. If the files are manually added or deleted, this can be automatically detected in the domain. The manually added applications are in the INSTALLED status, and the manually deleted applications are no longer managed by the domain.

Because an application ID is the application filename, the name is specified by using the -id option of the deploy command. To update the application files, manually update the files, which are located in a repository. If a cycle for automatic redeployment is set when an application is deployed, the application can be automatically redeployed by automatically detecting changes in the files.

If a filename in an application repository is the same as an application ID in the domain, the application cannot be recognized in the domain. In this case, the following log message is displayed:

```
[2016.08.08 12:37:40.625][1] [adminServer-68] [Deploy-0005] WARNING:
```

```
The application[myApp] already exists on /JEUS_HOME/.applications/myApp.ear.
Change the file name of [/home/user1/apps/myApp].
```

When a duplicated ID is used, if the application information is requested or the Master Server starts, the same log message will be generated.

If a valid application does not exist in an application repository, where you are trying to add to, the following log message is displayed:

```
[2016.08.08 12:37:40.625][1] [adminServer-68] [Deploy-0062] WARNING:
The repository(/home/user1/apps) does not contain any valid applications,
but new applications can be added.
```

If the application files, distributed or deployed on a domain, exist in an application repository to be deleted, the application continues to be managed by the domain, even after the application repository is deleted. In this case, the following log message is displayed:

```
[2016.08.08 12:44:04.328][1] [adminServer-93] [Deploy-0124] WARNING:
The application[myApp] is RUNNING in repository[/home/user1/apps].
To remove this application from the domain, undeploy it.
```

> JEUS cannot determine whether to delete running applications from the domain when deleting an application repository. To delete a running application, you need to undeploy it before deleting the repository.

Using the Console Tool

An application repository can be added to a domain with the **add-application-repository** command, and can be deleted from the domain with the **remove-application-repository** command. A subdirectory is created under the directory, where the applications installed on the Master Server reside, with an application ID that is assigned to it during the application installation. The application's archive files are placed in this new directory.

The following example adds, deletes, and searches for application repositories using the console tool.

```
[MASTER]domain1.adminServer>add-application-repository /apps/test
Successfully performed the ADD operation for An application repository.
Check the results using "add-application-repository or list-application-repositories"

[MASTER]domain1.adminServer>list-application-repositories
Application Repositories
========================================================================
+----------------------------------------------------------------------+
|                    Path to Application Repository                     |
+----------------------------------------------------------------------+
```

```
| /apps/test                                                                 |
+----------------------------------------------------------------------------+
============================================================================


[MASTER]domain1.adminServer>remove-application-repository /apps/test
Successfully performed the REMOVE operation for An application repository.
Check the results using "remove-application-repository or list-application-repositories"


[MASTER]domain1.adminServer>list-application-repositories
Application Repositories
============================================================================
+----------------------------------------------------------------------------+
|                        Path to Application Repository                       |
+----------------------------------------------------------------------------+
(No data available)
============================================================================
```

For detailed information on commands for adding, deleting, and querying application repositories using the console tools, see "add-application-repository", "remove-application-repository", and "list-application-repositories" in JEUS Reference Guide.

1.6.2. Deploying Applications in an Application Repository

After adding an application repository, you can deploy applications that exist in the repository. For more information, see Deploying Applications in an Application Repository.

1.7. Application Deployment by Specifying a Path

There are two ways to manage application files on a domain.

To configure the domain to manage the application files, it is recommended to install the applications in the INSTALL_HOME directory. If users want to directly manage the application files, it is recommended to set up application repositories. Even if the application files are directly managed by the user, in a production environment, it is recommended to group the application files by tasks and distribute them to multiple directories in the domain. In the development environment, applications do not need to be placed in the same path, but each in different paths. In such a case, applications in each directory can be deployed using the **-path** option.

You cannot use the -id option when deploying applications using the -path option. You can deploy both archive and exploded modules. If the -path option is used, applications are registered with the Master Server and then deployed, without being copied to the application's INSTALL_HOME directory. In such cases, an application ID becomes the name of the application file.

For more information on how to deploy by specifying a path, see Deploying Applications by Specifying the Path.

1.8. Staging Mode Deploy

Exploded modules can only be deployed to MSes, which reside on the same machine as the Master Server. Since exploded modules are used mostly during the development stage, exploded modules can be deployed and run on a single machine without having to use multiple machines. During the testing stage, however, since you can construct a domain on multiple machines while maintaining application in the exploded module format, there is no need to deploy the exploded module to another machine.

If the **-staging** option is used to deploy an application, the exploded module can be compressed and sent to a MS on a different machine. The application, that is deployed using the -staging option, is regarded by the MS as an archive module. The MS decompresses the module and deploys it. When the MS is restarted, the application files are synchronized with those on the Master Server.

> In staging deployment mode, files to synchronize between MS and the Master Server are not the files in the source application directory but the compressed file that the Master Server generates by compressing the directory. Therefore, modifications in files in the directory are not synchronized when MS restarts.

If there are changes to an application, the updates can be applied to the files on the MS using the redeploy command. Like the existing method for deploying archive modules, the redeploy command can be used without the -staging option. You cannot use the -path option in this case. When MS receives the redeploy command, it synchronizes the application files, undeploys the existing application, and then deploys the new application.

For more information about deploying an application in Staging mode, see Staging Mode Deploy.

1.9. Application Undeployment

Undeploying applications by explicitly using a JEUS management tool is different from undeploying applications when the server shuts down.

The following shows their differences.

	Undeployment using a management tool command	Undeployment during server shut down
File Deletion	The files created internally on the server will be deleted. For example, the results of JSP and EJB compilation, and the EJB 3.0 schedule information will be deleted from the database.	The files created internally on the server will not be deleted.
Timeout Mode	Graceful **Undeploy** Timeout	Graceful **Shutdown** Timeout (Undeploy Timeout is not applied.)

2. Graceful Undeployment and Redeployment

This chapter describes graceful undeployment and redeployment, which undeploys and redeploys an application after processing all current requests.

2.1. Graceful Undeployment

Graceful Undeployment undeploys an application after all user requests being processed are completed.

EAR, EJB, and web applications can process graceful undeployment. For EJB applications, only SessionBeans can process graceful undeployment.

If there are incomplete user requests when an application receives the undeploy command, the application waits for the requests to complete. The -gracefultimeout option can be used with the undeploy command. An application can be undeployed before all requests are completed, depending on the option value. When graceful timeout expires and there are incomplete requests, the application is undeployed by sending an interrupt signal to threads that are currently handling the requests.

When an application receives the undeploy command, it goes into the DISTRIBUTED state, where it cannot be executed. The application waits until the requests, which are being processed, are completed or the graceful timeout is reached. It is recommended for the user to set an appropriate -gracefultimeout option value. If the user does not specify the gracefultimeout option value, the default value of 5 minutes is used. This only guarantees the completion of requests in progress for 5 minutes.

Applications in the DISTRIBUTED state cannot handle external requests. EAR applications can handle internal requests, even after the process to undeploy the applications starts. If multiple Beans exist, standalone EJB applications can also handle requests from other Beans, even after the undeployment starts.

The following figure illustrates the execution of graceful undeployment.
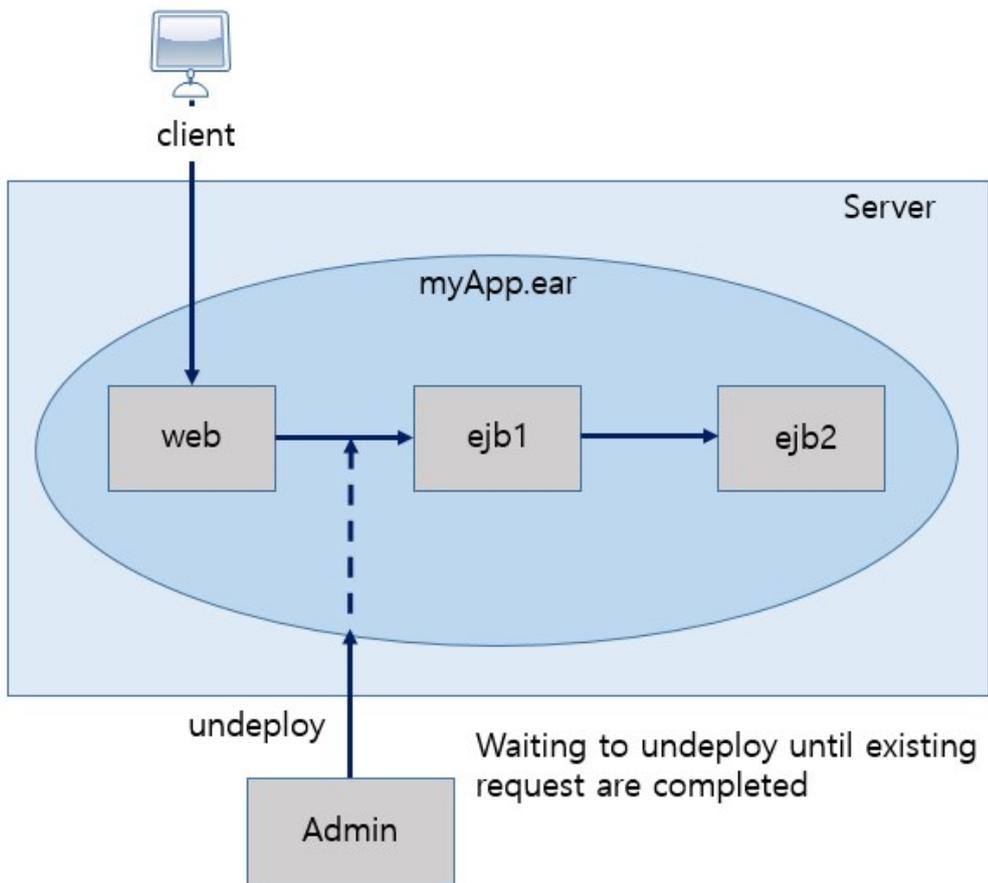
Graceful Undeployment of an EAR Application

In the previous figure, when web calls ejb1, the request can be handled, because myApp.ear is in the RUNNING state. When ejb1 calls ejb2, the request might be rejected, because the application is in the DISTRIBUTED state. However, because the request was sent to the application internally, it must be guaranteed to be handled, even if the application is in the DISTRIBUTED state. If an external application calls ejb1 in myApp.ear, the request may be rejected depending on the state of myApp.ear.

The call stack which calls ejb1 from the web, and then calls ejb2 from ejb1 is actually a single request. That is, from the point of myApp.ear, calling ejb2 from ejb1 is not considered a new request, but an existing request, triggered from the web. Even when myApp.ear is in the DISTRIBUTED state, the internal request that calls ejb2 must not fail. The request must be successfully handled, and the application must be undeployed, after the request is complete. Because it may take a long time to handle the request, the application can be undeployed before the request is complete, depending on the timeout value assigned with the undeploy command. When timeout is reached, the application sends an interrupt signal to the thread that handles the request to stop the job.

> Even if a thread receives an interrupt signal, not all jobs, handled by the thread, are stopped. Some jobs are stopped, while for some jobs, the interrupt signal is ignored. JEUS does not take any special actions for stopped jobs. The corresponding application is responsible for any post-processing jobs. For more information about thread interruptions, refer to "Thread Control" in JEUS Server Guide.

Graceful Undeployment for internal requests within the application applies to a standalone EJB application as well as an EAR application. If there are multiple Beans in an EJB application, requests between Beans are regarded as internal requests in the same application. Thus, the requests that are currently being processed must be completed, even if an undeploy command is executed.

The undeploy command has the default timeout value of 5 minutes, which means that application request processing is guaranteed for five minutes. The guaranteed time can be changed using the -gracefultimeout option, which is the same as the -to option. If needed, a large timeout value may be specified for graceful undeployment.

To force an immediate undeployment without waiting for the completion of the requests in progress, you can use the -force (-f) option. In JEUS 6, this is the default behavior of undeployment. In this case, the completion of the requests in progress is not guaranteed.

2.2. Graceful Redeployment

Graceful Redeployment is a mechanism that provides application services uninterruptedly by redeploying a running application. The behavior of graceful redeployment varies with different application types.

This section first covers the general information, and then further explains the different behaviors of graceful redeployment for each application type.

2.2.1. Requirements for Graceful Redeployment

To use Graceful Redeployment, applications need to satisfy the following requirements.

- Applications must be packaged into a .war or .jar file.

- For a directory type application, the old and new application directories must be different. Since a directory type application uses the source directory as the service image, the graceful redeployment requirements cannot be satisfied if the old and new applications are in the same directory. However, for Staging Mode Redeployment, the old and new applications can reside in the same directory because the Staging Mode creates a service image for each application.

- If an application is located in an application repository, it is impossible to redeploy by specifying a new application. Since an application in the repository is installed on the domain, it cannot be replaced with a new application through redeployment.

- For application packaging, the following field must be added to the META-INF/MANIFEST.MF file.

  A directory type application must also add the META-INF/MANIFEST.MF file and the following field.

  ```
  Jeus-Use-Graceful-Redeploy: true
  ```

  > According to the MANIFEST.MF file rules, a new line (\n) must exist at the end

of the file.

- An application to redeploy must also add the aforementioned field. Otherwise, general redeployment is executed for the application.

- For a directory type application, application-name and module-name must be the same in the standard DD of the old and new applications.

- Last modified time of each packaged file must be unique.

- For an EJB, graceful redeployment is supported only for session beans (remote interface).

- In the SHARED mode, graceful redeployment of an application is not supported, and it is recommended to package the application into an EAR.

2.2.2. Considerations for Graceful Redeployment

The following must be considered to execute graceful redeployment.

- Verify if the graceful redeployment function needs to be used before a service starts. To use the function, ensure that the requirements listed above are met.

- An existing application is not immediately undeployed, and thus two applications exist on JVM simultaneously for some time. Thus, the size of the JVM permgen area should be calculated in advance because an OutOfMemoryError may occur if the size is not big enough.

- If the JNI library is used, JVM cannot concurrently load the same library using System.loadLibrary(). Therefore, graceful redeployment cannot be used if JNI library is used.

2.2.3. Graceful Redeployment Execution

Graceful redeployment can be executed with the redeploy command. The following example shows how to execute graceful redeployment:

Assume that there are the following packaged war files.

- old application : myservice.war (08/09/2016 1:00 pm)

- new application : myservice.war (08/25/2016 11:00 am)

- install id = myservice_war, context path = /myservice

A service provider discovered an issue with myservice.war, which had been packaged and deployed on August 9, 2016, at 1:00 p.m. The service provider fixed the issue, and newly packaged the file on August 25, 2016, at 11:00 a.m. The service provider decides to execute graceful redeployment instead of stopping JEUS, because there are users who are already using the service.

Under such a circumstance, the service provider runs the following command by using the console tool.

```
[MASTER]domain1.adminServer>redeploy myservice_war -path /path_to_new_application/myservice.war
```

To execute graceful redeployment, the redeploy command does not require a special option. JEUS determines whether to execute graceful redeployment using the precondition field in the META-INF/MANIFEST.MF file and the packaging time. If the field does not exist in the META-INF/MANIFEST.MF file of the application, packaged on August 9, 2016, at 1:00 p.m., or the field is set to false, normal redeployment is executed.

> If handling client request takes a long time, set an appropriate value for the undeploy timeout option so that the previous application request can be processed within the timeout period before the redeployment.

How to forcibly Undeploy a Previous Version of an Application

When graceful redeployment is successfully completed, new users start to use the new version of myservice.war. Existing users who have been using the previous version continue to use the previous version. New users and existing users are classified based on (socket) connections. For web applications, HTTP sessions are also classified this way.

The main purpose of graceful redeployment is for the users to be able to use the services, uninterrupted. Therefore, the previous version is not undeployed, until all connections to it have been terminated and the HTTP sessions created through the connections have been deleted. If the previous version remains undeployed because the HTTP session timeout is too long, a service administrator can forcibly undeploy by using the following **undeploy** command as follows:

```
[MASTER]domain1.adminServer>undeploy -old myservice_war
```

How to Automatically Undeploy a Previous Version of an Application After a Timeout

To automatically undeploy the previous version of the application, myservice.war, after a certain time period, use the Timeout option (in seconds), with the **redeploy** command as follows:

```
[MASTER]domain1.adminServer>redeploy -path /path_to_new_application/myservice.war -to 5
```

How to Roll Back to a Previous Version When the Previous Version and a New Version Exist Simultaneously

If a service provider detects that the new version of myservice.war has a problem while the previous version has not been undeployed yet, the previous version should be rolled back. To roll back the previous version, undeploy the new version with the **undeploy** command as follows:

Run the **undeploy** command as follows:

```
[MASTER]domain1.adminServer>undeploy -new myservice_war
```

Assuming that the new version of the application myservice.war contains a critical issue, it is required to immediately undeploy it. In this case, users of the new version experience interruption of service. To avoid such a situation, undeploy the previous version and then redeploy the new version.

How to Only Distribute a New Application

To provide a new version of myservice.war to users, after checking if the application will run normally, only issue the distribute command using the **redeploy** command as follows:

```
[MASTER]domain1.adminServer>redeploy myservice_war -path /path_to_new_application/myservice.war
-distonly
```

The method to check if a new application will run normally varies depending on the application type. For information about how to check if a new web application will run normally, see How to Check if a New Web Application will Run Normally.

To start the new application after checking the behavior, run the **start** command as follows:

```
[MASTER]domain1.adminServer>start -new myservice_war
```

To undeploy the new application because of an issue, run the **undeploy** command along with the -new option as follows:

```
[MASTER]domain1.adminServer>undeploy -new myservice_war
```

How to Undeploy Both Previous and New Versions of an Application

When a previous version and a new version exist simultaneously, you need to explicitly specify the version to be undeployed using the -old or -new option. To undeploy both applications, use the -all option with the **undeploy** command as follows:

```
[MASTER]domain1.adminServer>undeploy -all myservice_war
```

How to Forcibly Replace a Previous Version of an Application

To force normal redeployment instead of using graceful redeployment, run the **redeploy** command as follows:

```
[MASTER]domain1.adminServer>redeploy myservice_war -path /path_to_new_application/myservice.war
-force
```

How to Verify if the Current Application is Set to Use Graceful Redeployment

To confirm if a currently running application is utilizing the graceful redeployment feature, view the details about the application. If the application uses graceful redeployment, you will find an item labeled 'Packaged Time' among the application details. This will allow you to determine whether the feature is enabled as well as when the application was packaged.

```
[MASTER]domain1.adminServer>appinfo -id myservice_war -server server1
Application information for the server [server1] in the domain [domain1].
===========================================================================
+-------+--------+--------+-----+-------+--------+-----------------+--------+
| Appli | Applic | Applic | Sta |Target | Target |  Packaged Time  | Target |
|cation | ation  | ation  | te  |Servers| Cluste |                 |VirtualH|
|  ID   | Name   | Type   |     |       | rs     |                 |  ost   |
+-------+--------+--------+-----+-------+--------+-----------------+--------+
| myser | myserv | war    | RUN | serve |        | Tue Nov 14      |        |
|vice_w |ice     |        |NING |r1     |        |14:06:59 KST 2017|        |
|ar     |        |        |     |       |        |                 |        |
+-------+--------+--------+-----+-------+--------+-----------------+--------+
```

2.2.4. Graceful Redeployment of Web Applications

A new version of a web application can be redeployed uninterruptedly, without stopping the current application service. Because general redeployment undeploys an existing web application, users may receive error messages in the web browser. However, if there are users who are using the application, graceful redeployment does not undeploy the existing version. It proceeds to deploy a new version of the web application, so that temporarily, the existing version and the new version exist simultaneously.

It can be determined if there are users, who are using an existing web application, by seeing if there are existing HTTP sessions or requests, that are currently being handled. The requests usually do not take a long time to be handled, if there is no DB delay problem or it is not being processed asynchronously, which takes a long time. However, HTTP sessions may remain for a long time until timeout is reached, which can prevent the existing web application from undeploying until timeout is reached. The user, who executes redeployment, should set an appropriate timeout to undeploy the existing version. The user should consider the characteristics of the requests and the HTTP sessions, or choose to forcibly undeploy the existing version.

When the new version is running successfully, new requests are no longer sent to the existing version. Users, who continues to use the existing version, are clients, who have connected before the successful completion of the graceful redeployment.

> If users, who are using an existing version, are using an HTTP session, they

cannot use the new version until the session expires. To prevent this, execute normal redeployment, or forcibly undeploy the existing application by executing graceful redeployment and setting the undeployment timeout. For detailed information, see Graceful Redeployment Execution.

How to Check if a New Web Application will Run Normally

You can confirm in advance if a new web application will run normally, after the application is distributed. For details about how to only distribute a new web application, see How to Only Distribute a New Application.

An HTTP request can be sent through the base port of the server, which distributed the new application. The following assumes that the application path of myservice.war is '/myservice,' the host name of the server that issued only the distribute command is host1, and the base port number of the server is 9736.

```
http://host1:9736/myservice/
```

2.2.5. Graceful Redeployment of EJB Applications

Graceful redeployment of EJB applications redeploys a new EJB file while guaranteeing the processing of existing EJB requests. Like graceful undeployment, graceful redeployment is supported only for session beans. Thus, in order to perform the graceful redeployment of a JAR application, the JAR file must only consist of session beans.

When graceful redeployment is performed, new clients use the new EJB by doing a lookup of the new EJB. Clients who are currently using the existing EJB are guaranteed the processing of requests for the Timeout period, specified by the user. If it is determined that there are no current requests for the existing EJB, the existing EJB will be undeployed, even if Timeout has not expired.

The request handling criteria varies depending on the EJB type. The request in progress is the criteria for stateless session beans, while the existence of a running session is the criteria for stateful session beans. In other words, for stateful session beans, undeployment will be performed when all sessions are destroyed, and it is determined that there will be no additional requests.

Currently, to improve performance, JEUS uses the client cache for Home Stub and EJB Object Stub. If a client is using the existing EJB from the cache, when graceful redeployment completes, the existing information in the cache must be deleted and new information about the new EJB must be obtained. EJB 3.x requires no additional configurations for normal operations, but for EJB 2.x, you must configure the <use-dynamic-proxy-for-ejb2> setting to true. The default value of <use-dynamic-proxy-for-ejb2> is true.

2.2.6. Graceful Redeployment of EAR Applications

The graceful redeployment of WEB and EJB modules in an EAR file is guaranteed for EAR applications. As in the graceful redeployment of standalone modules, the aforementioned services are guaranteed until the context's session expires. JEUS guarantees the processing of the current requests from stateless session beans, and requests that were received before the expiration of the stateful session beans.

Since an EAR application is a single application, when an EJB module in the EAR file completes a request, the EJB module cannot be undeployed if there are requests from other modules that are still being processed. Like in graceful undeployment, a module which is currently processing a request can send another request to the EJB module. Hence, you cannot guarantee the completion of all EAR application requests by undeploying only a single EJB or WEB module. All WEB or EJB module requests in an EAR file must be completed before undeploying the current EAR application.

	The name of the new application, as well as its module names, must be identical to those of the previous version of the application. For more information about configuring the application name, refer to "Jakarta EE 9 Platform Specification".

3. Applications

This chapter describes configuration of modules and applications, their components, and how to configure the components. It also explains JEUS functions for application deployment.

3.1. Modules and Applications

A Jakarta EE application consists of one or more modules. It contains one or more Jakarta EE components of the same type (EJBs, web applications, application clients, connectors) and deployment descriptors (DD).

DDs are divided into two types: Jakarta EE standard DD (application.xml) and JEUS DD (jeus-application-dd.xml). Most configurations in the standard DD can be replaced with annotations.

The following figure shows configurations of Jakarta EE modules and applications.
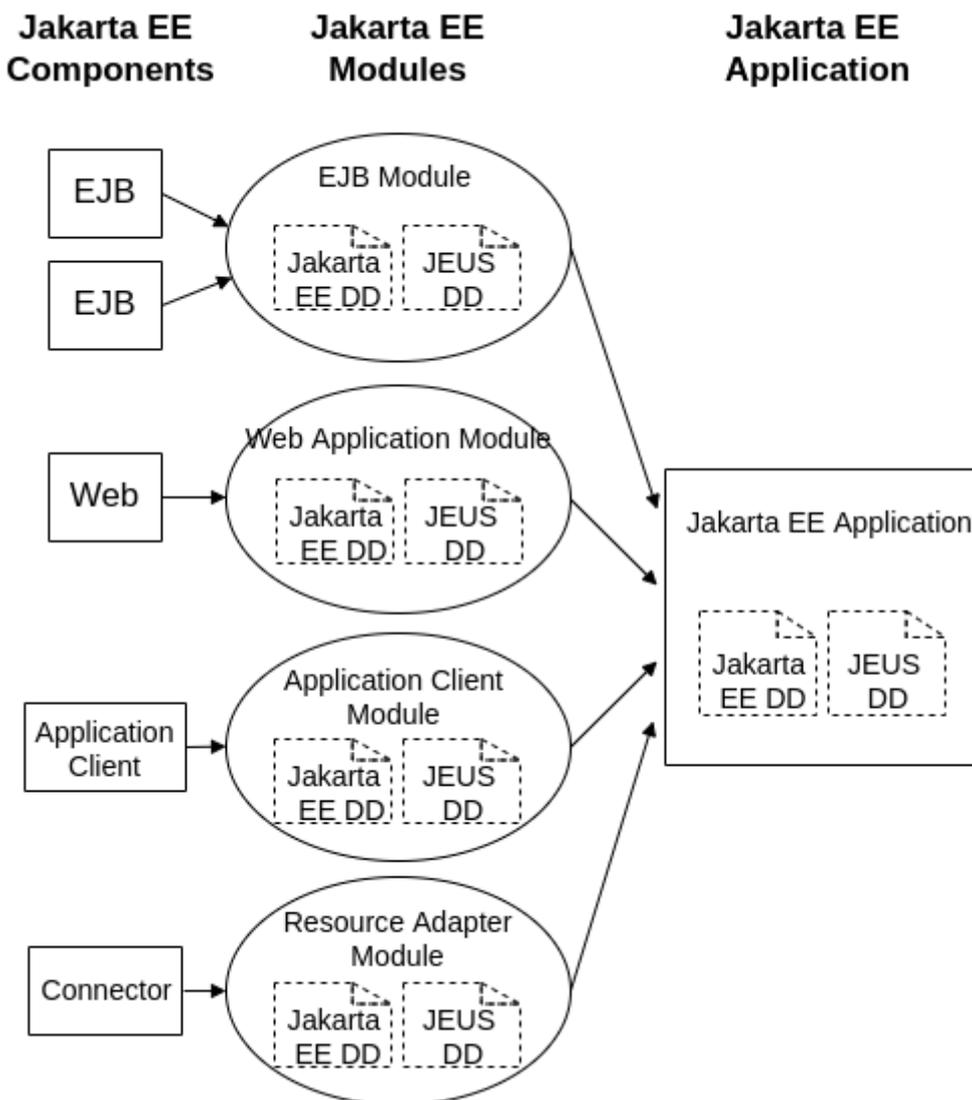
Jakarta EE Modules and Applications

3.1.1. Modules

Jakarta EE modules are divided into 4 types.

- **EJB module (.jar file)**

  EJB(Jakarta™ Enterprise Beans) is a standard server side component model, that implements business logic using transactions and security services. The role of an EJB module is to give access to and group the EJBs. In JEUS, EJB is the smallest unit that can be deployed to a JEUS EJB engine. Therefore, even when only one EJB is deployed, the EJB must be packaged as an EJB module (.jar file). For detailed information about EJB modules, refer to "JEUS EJB Guide".

- **Web application module (.war file)**

  A web module consists of static and dynamic contents of web-based services that are executed by a client request. Examples of web-based services are adding a product to a shopping cart, buying a product in the shopping cart through an online shopping site, or browsing products for purchase through a web-based auction site. For detailed information about web application modules, refer to "JEUS Web Engine Guide".

- **Application client module (.jar file)**

  An application client module is a client program executed in a JVM. This module can be executed by calling the main() method, and it ends when the virtual machine terminates.

  Like other Jakarta EE application components, an application client module operates in a client container, which provides system services. Client containers use a very small amount of system resources compared to other Jakarta EE containers. For detailed information about the application client modules, refer to "JEUS Application Client Guide".

- **Resource adapter module (.rar file)**

  A resource adapter is the main component of Jakarta EE connector architecture. It is developed for a specific enterprise information system (EIS) and provides the APIs to interact with EIS and system APIs to integrate with Jakarta EE application servers. To accomplish this, a JEUS administrator only needs to set and deploy a resource adapter. For detailed information about resource adapter modules, refer to "JEUS JCA Guide".

3.1.2. Applications

As shown in Jakarta EE Modules and Applications, a Jakarta EE application consists of one or more Jakarta EE modules and two deployment descriptors, Jakarta EE DD (application.xml) and JEUS DD (jeus-application-dd.xml).

A Jakarta EE application is a JAR archive file with the extension, '.ear'. The following figure shows a simple example of an EAR file. Each module's archive files, the lib directory, and the META-INF directory reside in the root directory.

The following illustrates the structure of an .ear archive file and each directory.

```
myApp.ear
    |--lib
    |    |--[J]myLib.jar
    |- META-INF
    |    |--[X]application.xml
    |    |--[X]jeus-application-dd.xml
    |--[J]appclient.jar
    |--[J]ejb.jar
    |--[J]web.war

* Legend
- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor
```

lib

The lib directory is a common library directory of EAR applications. Library files with the extension '.jar' are under this directory. The library files can be used, after being automatically added to the class path by a sub-module of an EAR application. This directory can be replaced by configuring a setting to use a different directory.

From Java EE 5 onwards, users can specify a common library directory with the <library-directory> tag in the application.xml file, which is the standard DD for EAR applications. If the application.xml file does not exist, or the <library-directory> tag is not set in the application.xml file, the `lib` directory will be used by default.

> The APP-INF directory, which is supported starting with JEUS 5, provides functions which are similar to those of the WEB-IF directory for web application modules. Similar to the WEB_INF directory, the APP-INF directory includes the subdirectories, named classes and lib, and they contain classes and '.jar' files, respectively.
>
> The class and '.jar' files can be used as a library by the corresponding application. However, starting with JEUS 6, it is recommended to use a library directory instead of the APP-INF directory.

META-INF

Includes two deployment descriptors: application.xml, Jakarta EE DD, and jeus-application-dd.xml, JEUS DD. These files are optional. Other than the two DDs, this directory contains the MANIFEST.MF file, which can be used to specify information and classpath of the archived EAR file.

3.2. Deployment Descriptor (DD)

A Jakarta EE application is deployed under the format of EAR(Enterprise ARchive .ear file).

An EAR file includes an EJB module (.jar), a web application module (.war), an application client module (.jar), a resource adapter module (.rar), and other necessary Java classes. A standalone module, that consists of a single module, is also considered as a type of Jakarta EE application. To start an application for service, the process of uploading the application module files to JEUS and controlling them is called **Deployment**.

To create a module or an application, to be distributed to an application server, the deployment descriptors, Jakarta EE standard DD and JEUS DD, are required.

> Since all settings in standard DD can now be expressed using annotations, standard DD and JEUS DD for EJB modules and EAR files, can be omitted, starting with JEUS 6, which complies with Java EE 5. Standard DD and JEUS DD for the Web application modules can be omitted in JEUS 7, which complies with Java EE 6.

The following shows deployment descriptors necessary for each module and application.

	Jakarta EE Standard DD	JEUS DD
Application	application.xml	jeus-application-dd.xml
EJB module	ejb-jar.xml	jeus-ejb-dd.xml
Web application module	web.xml	jeus-web-dd.xml
Application client	application-client.xml	jeus-client-dd.xml
Resource adapter module	ra.xml	jeus-connector-dd.xml
Web service	webservices.xml	jeus-webservices-dd.xml

As a module has a separate JEUS DD as well as Jakarta EE standard DD, there exists the jeus-application-dd.xml file for each application descriptor. This file is placed in the META-INF directory of the EAR, and is used to configure additional application settings. If this file does not exist in the EAR or Standalone modules, the application will be deployed using the default settings.

> For detailed information about Jakarta EE standard DD, refer to Jakarta EE 9 specifications. For detailed information about JEUS DD, refer to relevant JEUS guides.

When an application is deployed, information in jeus-application-dd.xml file is used to configure the environment, where the application will run in.

The following describes the <application> tag in the jeus-application-dd.xml file.

- Security settings

The security-related settings that will be applied to the application are as follows. For a detailed description of the items, refer to "JEUS Security Guide".

Element	Description
<role-permission>	A Principal-role mapping used by an application. This setting is applied to all modules of an application.
<java-security-permission>	Permissions used by an application, if the J2SE security manager is used.

* Library settings

  Using a shared library in an application requires the following setting. For detailed information about each element, see Shared Library.

Element	Description
<library-ref>	Shared library used by an application.

* Jakarta EE Namespace Settings

  Information about namespaces to be used by an application can be specified in the application.xml file or in the annotations. Mapping information about the namespaces can be specified in the jeus-application-dd.xml file. The information can be replaced by annotations. For detailed information about the naming convention, refer to Jakarta EE 9 specifications. For detailed information about namespace related settings, refer to "Configuring jeus-application-dd.xml" in *JEUS XML Reference*.

* Classloading Settings

  The sequence of loading the libraries included in an EAR application can be configured. By default, classes are loaded from the parent class loader in the Java class loader. However, there may be a situation that the libraries included in an application need to be loaded before the parent library depending on the application setting. In such case, the sequence of loading classes can be changed.

Element	Description
<classloading>	Method of loading the libraries included in an EAR application. If it is set to true, the libraries included in EAR are first loaded. If there is not library, the classes are loaded from the parent class loader.

3.3. Application Libraries

This section describes how to add and use the application library (lib/application), and a shared library. It also describes library deployment, which is a new feature added to JEUS.

3.3.1. lib/application Directory

The application library, lib/application, is shared among applications. It is distinguished from the system library, JEUS_HOME/lib/system that is added to JEUS system.

The system library includes libraries used by the system. The application library contains libraries, used in classes defined by applications or users. The libraries contain a function to manage the server life cycle, a user logger, logger filter, logger formatter, etc. The classes do not reference libraries in the application library. The libraries are shared among applications by placing them in the lib/application directory.

Because libraries in lib/application can be shared by applications, a user does not need to include the libraries when packaging applications. Libraries in lib/application are usually used by all the applications in a domain or a server. A library can be a .jar file or .class file.

> If the same libraries with different versions are installed in lib/application, an application may not be able to use the desired library. To prevent this, use shared libraries for which a version to be used by an application can be specified. For more information about shared libraries, see Shared Library.

The lib/application directory can be placed in both the DOMAIN_HOME and SERVER_HOME directories.

Libraries shared by an entire domain are located in the DOMAIN_HOME/lib/application directory. To transfer an application library installed on a domain to a server on a separate machine, create a domain on a separate machine with the pack-domain command to copy the existing DOMAIN_HOME/lib/application directory to the machine. If the libraries are added or modified after the domain is created, they should be manually synchronized by the user.

Libraries that are used only by a specific server are located in the SERVER_HOME/lib/application directory. The libraries should be manually copied to this directory.

> If needed, you can manually create the SERVER_HOME/lib/application directory to place the libraries, since the directory is not created when installing JEUS.

Class Loading

Classes in the application library, lib/application, are loaded by the JEUS root class loader. The classes can be used by all the applications deployed on the server.

Classes in the system library, JEUS_HOME/lib/system, are also loaded by the JEUS root class loader, but the system library resides in a directory that a user cannot access. However, since the application library is a directory meant to be referenced by the user, it can be accessed by the user.

> For a detailed explanation of the server's class loading method, refer to "Class

The lib/application directory can be placed in both the DOMAIN_HOME and SERVER_HOME directories.

The SERVER_HOME/lib/application directory is first added to the classpath of the class loader, and then the DOMAIN_HOME/lib/application directory is added. If a library already exists in both the SERVER_HOME and DOMAIN_HOME directories, the library in the SERVER_HOME directory will be loaded, and the library in the DOMAIN_HOME directory will be ignored.

If a library with a name, that is already used by another library in the class loader, is added, the following warning message will be displayed when the server boots:

```
Warning [same classpath-name] : [JEUS_HOME/domains/domain1/servers/adminServer
/lib/application/applib.jar] is registered earlier than
[JEUS_HOME/domains/domain1/lib/application/applib.jar] in JEUSClassLoader.
```

> The previous log message is displayed when lib/application is added to the JEUS root class loader, using a classpath, in the class loader configuration step. This step is the first step in booting a server and is processed before a server logger is configured. Therefore, the message should be checked with the launcher logs.

Application libraries can be used in applications, as well as in all cases when a special class can be configured in the server, including server life cycle call function, user logger, log message filter class, log message formatter class, etc. Since the special classes do not have reference to the libraries in the application library, they must be placed in the lib/application directory.

3.3.2. Shared Library

Shared libraries are shared among the applications, and are distinguished from the JEUS system library, JEUS_HOME/lib/system, and the application libraries, DOMAIN_HOME/lib/application and SERVER_HOME/lib/application.

Shared libraries do not affect the entire JEUS system. Each application can specify which shared libraries it will use. Shared libraries can be dynamically added, without restarting JEUS, and can be used selectively when different versions of the libraries are installed.

Shared libraries have the following characteristics.

- Because a shared library can be shared among applications, the user does not need to consider the library when packaging applications.

- A shared library can be dynamically added and removed even when a server is running.

- A shared library can be upgraded by adding a new library and redeploying the applications.

- Multiple versions of implementation classes, in the shared library, can be registered, and during deployment, one can be selected for use.

Because there can be two versions of a library, a specification version and an implementation version, the user can install multiple versions of the same library. A version, highest, minimum, or exact, required by an application, can be selected dynamically during deployment.

To support the multiple versions of a library, it is recommended to specify a version for use. If a version is not specified, the default version, 0, will be internally used. A version may not be used at all, for simple use cases.

Installing and Setting Libraries

A library consists of multiple JAR files, which are usually located in the shared library directory, JEUS_HOME/lib/shared. The JAR files are registered in the configuration file, JEUS_HOME/lib/shared/libraries.xml, as follows:

In the following example, myLibrary is registered as the implementation class 2.1, which implements the myLibrary 2.0 specification. This library consists of multiple JAR files: commons-logging.jar, commons-util.jar, and all JAR files under the myLib -2.1 directory.

Registering Shared Libraries: <libraries.xml>

```
<library>
    <library-name>myLibrary</library-name>
    <specification-version>2.0</specification-version>
    <implementation-version>2.1</implementation-version>
    <files dir=".">
        <include name="commons-logging.jar"/>
        <include name="commons-util.jar"/>
    </files>
    <files dir="myLib-2.1"/>
</library>
```

Descriptions for each tag are as follows:

- <library-name>, <specification-version>, <implementation-version>
    - Used when an application references a specific library.
    - <*-version> fields can be used, when multiple versions of the same library are used.
- <files>

    Sets a classpath of an actual library in multiple ways as follows:

    The <files> Tag for Shared Libraries

```
<files dir=".">
    <include name="a.jar"/>
    <include name="b.jar"/>
</files>

<files dir="testa"/>
```

```
<files dir="/home/works/lib/testc" />

<files dir="/home/works/lib/testd" mode="classes"/>
```

- dir is a directory, which can be the JAR files directory or the classes directory. A relative or an absolute path can be used. If dir is a relative path, the base directory is JEUS_HOME/lib/shared.

- The <include> child tag specifies the JAR files to be included. If this tag is not set, all JAR files in the corresponding directory will be included. JAR files, in the directory, will be searched for during deployment. Therefore, JAR files can be added to this directory without making any changes to the settings. To specify the classes directory, not JAR files, set the mode to 'classes'.

- The libraries can be dynamically added while JEUS is running, because if the settings are changed while a new application is deployed, the settings will be read again. Therefore, a new library or an updated library can be added without rebooting JEUS.

> The settings described above must be used by directly modifying the corresponding .xml file.

Referencing Libraries from Applications

Jakarta EE applications and standalone modules can use the registered shared libraries, with entries in jeus-application-dd.xml, jeus-web-dd.xml, and jeus-ejb-dd.xml files.

The following example references the shared library, 'myLibrary'.

```
<library-ref>
    <library-name>myLibrary</library-name>
</library-ref>
```

In the previous example, when an application is deployed, it searches for the library 'myLibrary' and adds the corresponding classpath to the application classpath. If there are multiple versions of 'myLibrary', the latest version will be selected. If a version of a library to be referenced is not specified, the latest version will be used always according to Version Ordering Rule.

The following example requires the earliest version.

```
<library-ref>
    <library-name>myLibrary</library-name>
    <specification-version>
        <value>2.0</value>
    </specification-version>
    <implementation-version>
        <value>2.0</value>
    </implementation-version>
```

```
    </library-ref>
```

The following example requires an exact version.

```
<library-ref>
    <library-name>myLibrary</library-name>
        <specification-version>
                <value>2.0</value>
                <exact-match>true</exact-match>
        </specification-version>
        <implementation-version>
                <value>2.0</value>
                <exact-match>true</exact-match>
        </implementation-version>
</library-ref>
```

To only specify an exact specification version, set as follows. In this case, the application searches for the latest implementation version of the corresponding specification.

```
<library-ref>
    <library-name>myLibrary</library-name>
        <specification-version>
                <value>2.0</value>
                <exact-match>true</exact-match>
        </specification-version>
</library-ref>
```

If a referenced library is not found during deployment, a WARNING log message will be displayed, but the deployment will continue to be processed. To discontinue the deployment and make it fail, set the <failon-error> attribute as follows.

```
<library-ref>
    <library-name>myLibrary</library-name>
    <failon-error>true</failon-error>
</library-ref>
```

Class Loading

Shared library class is loaded by an application class loader or a module class loader, depending on where the library reference is defined. For example, if 'lib1' is defined in jeus-application-dd.xml as a reference, it will be loaded by an application class loader. If it is defined in jeus-web-dd.xml as a reference, it will be loaded by a web-level class loader. A class or a library loaded by an application class loader is isolated in the corresponding application. That is, the class instance is not shared with other applications.

> For a detailed explanation of the server's class loading method, refer to "Class

Version Ordering Rule

A version is made up of a fraction part and a non-fraction (string) part. For example, the version "6.2.3-b12" consists of <fraction_part> (6.2.3) and <string(non-fraction)_part> (-b12).

```
Version ::= <fraction_part> | <string_part> | <fraction_part> <string_part>
fraction_part ::= <integer> | <integer> "." <fraction_part>
string_part ::= <non-numeric> <character>*
```

The ordering rule of a version is as follows:

- Numbers in <fraction_part> are compared numerically, in the order of major and minor.

- If numbers in <fraction_part> are identical, strings in <string_part> will be compared.

The following shows the versions that are ordered according to the ordering rule.

```
6.0 < 6.2.3 < 6.2.3-b12 < 6.2.3-beta < 6.2.4
```

3.3.3. Library Deployment

From JEUS 8 onwards, the library deployment feature is provided so that users can manage libraries through the Master Server. Users can deploy a library to a domain by using a JEUS console tool and then configure an application to reference the library. This deployment and reference function minimizes class conflicts that can occur when multiple applications reference different versions of one library.

The advantages of library deployment are as follows.

- Users do not need to have a library included in each application for packaging. This resolves inconvenience that users must perform packaging for all applications that use the same library. It also resolves resource usage problems that are caused when the same class is loaded by multiple applications.

- Users can specify a library to use when deploying an application. To deploy the same library to multiple applications, the library can be set with different versions for each application.

Key Features

Installation and deployment are required to use a library in applications. The features used to deploy a library are as follows:

1. **Library Installation**

Install a library to deploy in the domain directory of the server in which the Master Server is installed. Installation is performed through a console command. The information required for installation are as follows.

Item	Description
Library Identifier (ID)	Name of a library to deploy, delete, or reference. This must be unique in a domain. If this option is not set, installation is not proceeded.
Library Version	Version of library to install. If not set, the version is deemed to be 1.0.
Library Path	Path to the library to install. If not set, installation is not proceeded.

After the installation completes, the library files are located under DOMAIN_HOME/.libraries/LIBRARY_ID/VERSION.

> The information about the installed libraries are not stored in the configuration file. The Master Server will search for the directory of DOMAIN_HOME/.libraries if library installation status needs to be identified. Therefore, changing this directory is not recommended.

2. **Library Deployment**

   An installed library can be deployed using console commands. Specify a server or cluster to use the library. All servers can be specified as well. When deployment completes successfully, the Master Server saves the changes in the configuration to maintain library installation status when it is restarted.

3. **Library Undeployment**

   A library can be undeployed using console commands when it is no longer used. When undeployment completes, the Master Server removes the information about the undeployed library from the configuration. The library file itself is maintained.

4. **Library Uninstallation**

   A library file can be deleted using console commands.

3.3.3.1. Installing and deleting a library

This section describes how to install and delete a library.

Using the Console Tool

The following describes how to install and delete a library using a console tool jeusadmin.

- **Checking the information about installed libraries**

View the list of installed and deployed libraries using the **library-info** command.

```
[MASTER]domain1.adminServer>library-info
Library information
================================================================================
+-----------+--------+-------+--------------+-----------------+------------+
| Library ID| Version| State | Target Servers| Target Clusters | Applications|
+-----------+--------+-------+--------------+-----------------+------------+
(No data available)
================================================================================
```

- **Library Installation**

  Install a library using the **install-library** command. Library ID and path must be entered. Otherwise, installation is not proceeded.

```
[MASTER]domain1.adminServer>install-library log4j -path /usr/lib/apache-log4j-1.2.17/log4j-
1.2.17.jar -version 1.2.17
Successfully installed the library [log4j] version [1.2.17].
[MASTER]domain1.adminServer>library-info
Library information
================================================================================
+-----------+--------+----------+-------------+-----------------+-----------+
| Library ID| Version|   State  |    Target   | Target Clusters | Applicatio |
|           |        |          |    Servers  |                 |     ns    |
+-----------+--------+----------+-------------+-----------------+-----------+
| log4j     | 1.2.17 | INSTALLED|             |                 |           |
+-----------+--------+----------+-------------+-----------------+-----------+
================================================================================
```

- **Library Deployment**

  Deploy a library using the **deploy-library** command. Specify a server, cluster, or all servers to use the library for deployment.

```
[MASTER]domain1.adminServer>deploy-library log4j -servers adminServer
deploy the library [log4j] succeeded.
[MASTER]domain1.adminServer>library-info
Library information
================================================================================
+-----------+--------+--------+--------------+-----------------+------------+
| Library ID| Version|  State | Target Servers|  Target Clusters | Applications|
+-----------+--------+--------+--------------+-----------------+------------+
| log4j     | 1.2.17 | RUNNING| adminServer  |                 |            |
+-----------+--------+--------+--------------+-----------------+------------+
================================================================================
```

- **Library Undeployment**

  Undeploy a library using the **undeploy-library** command if it is no longer used. Undeployment does not cause any functional issue of the server. However, it may require unnecessary tasks

such as library synchronization when the server is restarted.

```
[MASTER]domain1.adminServer>undeploy-library log4j
undeploy the library [log4j] succeeded.
[MASTER]domain1.adminServer>library-info
Library information
================================================================================
+-----------+--------+----------+-------------+-----------------+-----------+
| Library ID| Version|   State  |    Target    | Target Clusters | Applicatio |
|           |        |          |   Servers    |                 |     ns     |
+-----------+--------+----------+-------------+-----------------+-----------+
| log4j     | 1.2.17 | INSTALLED|             |                 |           |
+-----------+--------+----------+-------------+-----------------+-----------+
================================================================================
```

- **Library Uninstallation**

  Remove a library from the domain using the **uninstall-library** command. A deployed library cannot be deleted. Undeployment must be performed first to delete a library.

```
[MASTER]domain1.adminServer>uninstall-library log4j
uninstall the library [log4j] succeeded. : Successfully deleted [log4j].
[MASTER]domain1.adminServer>library-info
Library information
================================================================================
+-----------+--------+-------+--------------+-----------------+------------+
| Library ID| Version| State | Target Servers|  Target Clusters | Applications|
+-----------+--------+-------+--------------+-----------------+------------+
(No data available)
================================================================================
```

3.3.3.2. Referencing Libraries from Applications

When deploying an application, a library to reference can be specified. Specify the ID and version of a library to deploy the application. If a version is not set, it is deemed to use the latest version of the library.

A deployer performs tasks that enable applications to reference a library based on the specified information. The information about the library referenced by applications is saved in domain.xml, same as other deployment information.

Using the Console Tool

Specify the identifier and version of a library to reference when executing the **deploy-application** command.

```
[MASTER]domain1.adminServer>deploy-application sample -lib log4j -version 1.2.17 -servers adminServer
deploy the application for the application [sample] succeeded.
[MASTER]domain1.adminServer>library-info
Library information
```

```
================================================================================
+-----------+--------+--------+--------------+------------------+-------------+
| Library ID| Version|  State | Target Servers| Target Clusters | Applications|
+-----------+--------+--------+--------------+------------------+-------------+
| log4j     | 1.2.17 | RUNNING| adminServer  |                  | sample      |
+-----------+--------+--------+--------------+------------------+-------------+
================================================================================
```

```
+-----------+--------+--------+--------------+------------------+-------------+
| Library ID| Version|  State | Target Servers| Target Clusters | Applications|
+-----------+--------+--------+--------------+------------------+-------------+
| log4j     | 1.2.17 | RUNNING| adminServer  |                  | sample      |
```

4. Application Implementation and Deployment

This chapter describes how to write Jakarta EE application files (EAR files) and deploy them on JEUS. It also explains how to deploy applications and handle related jobs by using tools provided by JEUS.

4.1. Application Implementation

This section describes how to write a Jakarta EE application, which includes the already created modules from the previous sections, using the 'jar' utility.

Before writing an EAR file, create modules that need to be included in the file. The modules can be EJB modules (JAR files), web application modules (war files), application client modules (JAR files), or resource adapter modules (rar files). For information about how to create each module, refer to their relevant guides.

Descriptions of myApp.ear, the application shown in this section, are as follows:

- myApp.ear is an EAR file which includes the EJB module ejb.jar, the web application module web.war, and the application client module appclient.jar.

- myApp.ear must be installed on the Master Server adminServer, and then deployed on the server named 'server1'.

The following is the process of creating an application.

1. Write modules to be included in the EAR file.

2. Create the META-INF directory, where the directory, which includes JAR, WAR, and RAR files, resides.

3. Create application.xml file, which includes the modules of the EAR file and copy it to the META-INF directory.

   <application.xml>

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <application version="9"
       xmlns="https://jakarta.ee/xml/ns/jakartaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
           https://jakarta.ee/xml/ns/jakartaee/application_9.xsd">
       <description>Application description</description>
       <display-name>Sample application</display-name>
       <module>
           <ejb>ejb.jar</ejb>
       </module>
       <module>
           <web>
               <web-uri>web.war</web-uri>
               <context-root>hello</context-root>
           </web>
   ```

```
    </module>
    <module>
        <java>appclient.jar</java>
    </module>
</application>
```

4. Run the following command using the jar utility to create the myApp.ear file.

```
> jar cf myApp.ear ejb.jar web.war appclient.jar META-INF
```

> Use only upper cases for the name of the META-INF directory. If a lowercase is used, an error will occur.

4.2. Deployment Commands

Deployment tools provided in JEUS support the following jobs.

Item	Description
distribute	Copies application files to the target servers or clusters and prepares to run the application. If the distribution job fails in one or more target servers or clusters, the entire job will fail, and application files will be undeployed from all successful servers.
deploy	Copies application files to the target servers or clusters and prepares to run the application. If the distribution job fails in one or more target servers or clusters, the entire job will fail, and application files will be undeployed from all successful servers. Even if the start job fails in one or more servers, it is regarded as successful, if there are one or more servers where the application can run.
start	Starts applications that have been distributed on the target servers. Even if the start job fails in one or more servers, it is regarded as successful, if there are one or more servers where the application can run.
stop	Suspends the running applications deployed on the target servers. The applications are not removed from the target servers, and can be started or redeployed using the application name.
undeploy	Stops running the applications deployed on the target servers. The applications are removed from the target servers.
redeploy	Updates and redeploys the applications, when currently running applications are modified. If one or more applications fail to be redeployed, all applications will be stopped.

The following deployment jobs are available only in JEUS.

Task	Description
Add Application Target	Adds specific servers or clusters as the targets of deployed or distributed applications. This expands the targets of the applications that are currently running. Targets can be multiple servers or clusters. If this job fails on one or more new targets, the applications will be undeployed from all the new targets, but not on the existing targets.
Remove Application Target	Removes specific servers or clusters from targets of deployed or distributed applications. It reduces the number of targets of the applications. Targets can be multiple servers or clusters.

4.3. Controlling and Monitoring Applications

This chapter describes how to control and monitor applications.

4.3.1. Installing Applications on a Domain

The following is the process of installing an application on a domain.

Using the Console Tool

The **install-application** command installs an application into the domain and places it under INSTALL_HOME (DOMAIN_HOME/.applications). In this step, you need to specify the path to the application file to be installed. You can specify the application ID using the '-id' option.

If the application ID is not specified, JEUS automatically assigns an application ID. If the application file name is `myApp.ear`, `myApp_ear` becomes the application ID.

```
----------------------------------------------------------------
 using install-application command for application install
----------------------------------------------------------------
[MASTER]domain1.adminServer>install-application -id myApp /usr/apphome/myApp.ear
Successfully installed the application [myApp].

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+----------+-------+---------+----------+----------------------+
| Applicati|Application| State |  Server  |  Cluster |    Application Path   |
|  on ID   |   Type   |       | Targets  | Targets  |                       |
+----------+----------+-------+---------+----------+----------------------+
| myApp    |          | INSTAL|         |          | ${INSTALL_HOME}/myApp/m|
|          |          |LED    |         |          |yApp.ear               |
+----------+----------+-------+---------+----------+----------------------+
```

```
================================================================
```

For more information about the uninstall-application command, see "uninstall-application" in JEUS Reference Guide.

4.3.2. Uninstalling Applications from a Domain

An application can be removed from a domain, if the application is no longer being used. Applications, in the INSTALLED or DEPLOYED states, can be uninstalled.

The following is the process of uninstalling an application from a domain.

Using the Console Tool

You can uninstall an application using the **uninstall-application** command. When running the command, you need to specify the application ID to uninstall.

```
[MASTER]domain1.adminServer>uninstall-application myApp
uninstall the application for the application [myApp] succeeded. : Successfully deleted [myApp].

[MASTER]domain1.adminServer>application-info
No applications exist in this domain.
Application information for the domain [domain1].
================================================================================
+------------+---------------+-------+-----------+------------+--------------+
| Application| Application   | State |   Server  |   Cluster  | Application  |
|    ID      |     Type      |       |  Targets  |   Targets  |     Path     |
+------------+---------------+-------+-----------+------------+--------------+
(No data available)
================================================================================
```

4.3.3. Deploying Applications

An installed application can run using the deploy command. There are three ways to deploy applications.

- Deploying Applications Installed on a Domain

- Deploying Applications in an Application Repository

- Deploying Applications by Specifying the Path

Executing runtime deployment using a console tool assumes the following scenario.

- MyApp.ear described in Application Implementation is used.

- JEUS domain name is 'domain1,' and the name of the server where the application will be deployed is 'server1.'

For more information about the options available when executing the **deploy-application** command, see "deploy-application" in JEUS Reference Guide.

JEUS provides three methods for runtime deployment through the console tool, according to the location where an application resides. Each method deploys an application in this order: deploy, stop, start, redeploy, and undeploy commands. After executing each command, **applist** is executed to check the status of the application in each step.

JEUS console tool can be used to connect to DAS to manage servers and clusters and deploy applications.

The following are console tool commands related to application deployment.

Command	Description
distribute-application	Copies the application files to target servers or clusters, and prepares the application for service.
deploy-application	Distributes the application files to target servers or clusters, and starts the application. If an application is deployed successfully, it will be in the RUNNING state. If the application is distributed successfully, but cannot be started, it will remain in the DISTRIBUTED state.
start-application	Starts an application, that is in the DISTRIBUTED state. While this task is running, the application goes into the STARTING state. After the task has been completed successfully, the application goes into the RUNNING state.
stop-application	Stops an application in the RUNNING state. During this process, the application goes into the STOPPING state. When the task completes successfully, the application goes into the DISTRIBUTED state.
redeploy-application	Redeploys the application when a deployed application is modified. It executes each step in the same way as the deploy command. The status in each step also matches the status in each step of the deploy command.
undeploy-application	Stops running the applications deployed on the target servers. The applications are removed from the target servers.
application-info	Displays information about applications in a domain.
add-application-target	Adds a target for deployed applications. Servers and clusters can be the target.
remove-application-target	Removes a target for deployed applications. Servers and clusters can be the target.

For a description of the console tool and detailed information about each command, see "jeusadmin" in JEUS Reference Guide.

4.3.3.1. Deploying Applications Installed on a Domain

The following describes how to deploy an application installed on a domain.

Using the Console Tool

The **install-application** command installs an application into the domain and places it under the INSTALL_HOME (DOMAIN_HOME/.applications) directory. The application ID can be specified as an option.

When the ID is not specified, the application file name 'myApp_ear' becomes the application ID. After an application has been installed, it can be deployed on the target servers using the **deploy-application** command.

```
-----------------------------------------------------------------
 using deploy command for application with install application
-----------------------------------------------------------------
[MASTER]domain1.adminServer>install-application -id myApp /usr/apphome/myApp.ear
Successfully installed the application [myApp].

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+-------+----------+----------+----------------------+
| Applicati|Application| State |  Server  |  Cluster |   Application Path    |
|  on ID   |  Type     |       | Targets  |  Targets |                      |
+----------+-----------+-------+----------+----------+----------------------+
| myApp    |           | INSTAL|          |          | ${INSTALL_HOME}/myApp/m|
|          |           | |LED  |          |          | |yApp.ear            |
+----------+-----------+-------+----------+----------+----------------------+
================================================================================

[MASTER]domain1.adminServer>deploy myApp -servers server1
deploy the application for the application [myApp] succeeded.

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicati| Application|  State  |  Server  |  Cluster |   Application Path  |
|  on ID   |   Type     |         | Targets  |  Targets |                    |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR        | RUNNING | server1  |          | ${INSTALL_HOME}/myAp|
|          |            |         |          |          | |p/myApp.ear        |
+----------+-----------+---------+----------+----------+--------------------+
================================================================================

[MASTER]domain1.adminServer>stop-application myApp
stop the application for the application [myApp] succeeded.

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+--------+----------+----------+----------------------+
| Applicati|Application|  State |  Server  |  Cluster |   Application Path    |
|  on ID   |   Type    |        | Targets  |  Targets |                      |
```

```
+----------+-----------+--------+----------+----------+----------------------+
| myApp    | EAR       | DISTRIB| server1  |          | ${INSTALL_HOME}/myApp/m|
|          |           |UTED    |          |          |yApp.ear               |
+----------+-----------+--------+----------+----------+----------------------+

==============================================================================

[MASTER]domain1.adminServer>start-application myApp
start the application for the application [myApp] succeeded.

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
==============================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicati| Application|  State  |  Server  |  Cluster |  Application Path  |
|  on ID   |    Type   |         | Targets  | Targets  |                    |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |          |  ${INSTALL_HOME}/myAp|
|          |           |         |          |          |p/myApp.ear         |
+----------+-----------+---------+----------+----------+--------------------+

==============================================================================

[MASTER]domain1.adminServer>redeploy-application myApp
redeploy application on JEUS Master Server for the application [myApp] succeeded.

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
==============================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicati| Application|  State  |  Server  |  Cluster |  Application Path  |
|  on ID   |    Type   |         | Targets  | Targets  |                    |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |          |  ${INSTALL_HOME}/myAp|
|          |           |         |          |          |p/myApp.ear         |
+----------+-----------+---------+----------+----------+--------------------+

==============================================================================

[MASTER]domain1.adminServer>undeploy myApp
Undeploying [myApp] (This may take time due to graceful undeployment) ..........
undeploy the application for the application [myApp] succeeded.
successfully undeployed (elapsed = 415ms)

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
==============================================================================
+----------+-----------+-------+----------+----------+----------------------+
| Applicati|Application| State |  Server  |  Cluster |   Application Path    |
|  on ID   |   Type    |       | Targets  | Targets  |                       |
+----------+-----------+-------+----------+----------+----------------------+
| myApp    | EAR       | INSTAL|          |          | ${INSTALL_HOME}/myApp/m|
|          |           |LED    |          |          |yApp.ear               |
+----------+-----------+-------+----------+----------+----------------------+

==============================================================================

[MASTER]domain1.adminServer>uninstall-application myApp
uninstall the application for the application [myApp] succeeded. : Successfully deleted [myApp].

[MASTER]domain1.adminServer>application-info
No applications exist in this domain.
Application information for the domain [domain1].
```

```
================================================================================
+-----------+--------------+-------+-----------+-----------+--------------+
| Application|  Application | State |  Server   |  Cluster  | Application  |
|    ID     |     Type     |       |  Targets  |  Targets  |    Path      |
+-----------+--------------+-------+-----------+-----------+--------------+
(No data available)
================================================================================
```

4.3.3.2. Deploying Applications in an Application Repository

After adding an application repository, you can deploy the applications located in the repository. For information about how to add and delete application repositories, see Adding, Deleting, and Searching Application Repositories.

Using the Console Tool

The following is the process of deploying applications from an application repository using the console tool.

```
[MASTER]domain1.adminServer>add-application-repository /home/user1/apps
Successfully performed the ADD operation for An application repository.
Check the results using "add-application-repository or list-application-repositories"

[MASTER]domain1.suok>list-application-repositories
Application Repositories
================================================================================
+------------------------------------------------------------------------------+
|                        Path to Application Repository                         |
+------------------------------------------------------------------------------+
|  /home/user1/apps                                                            |
+------------------------------------------------------------------------------+
================================================================================

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+-------+----------+-----------+----------------------+
| Applicati|Application| State |  Server  |  Cluster  |    Application Path   |
|  on ID   |   Type    |       | Targets  |  Targets  |                       |
+----------+-----------+-------+----------+-----------+----------------------+
| exploded |           | INSTAL|          |           | /home/user1/apps/explod|
|          |           | |LED  |          |           | |ed                    |
+----------+-----------+-------+----------+-----------+----------------------+
================================================================================

[MASTER]domain1.adminServer>deploy exploded -servers server1
deploy the application for the application [exploded.war] succeeded.

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+--------+----------+----------+-------------------+
| Applicati| Application|  State |  Server  |  Cluster |   Application Path |
|  on ID   |    Type   |        | Targets  |  Targets |                    |
```

```
+----------+-----------+---------+----------+-----------+-------------------+
| exploded | WAR       | RUNNING | server1  |           | /home/user1/apps/exp|
|          |           |         |          |           |loded              |
+----------+-----------+---------+----------+-----------+-------------------+

====================================================================
```

4.3.3.3. Deploying Applications by Specifying the Path

When a parent directory cannot be added as an application repository, the application can be deployed by specifying the path. For information about how to deploy an application, in a machine where the Master Server exists, using the absolute path, see Application Deployment by Specifying a Path.

Using the Console Tool

The following is the process of deploying an application using the console.

```
[MASTER]domain1.adminServer>deploy -path /home/user1/apps/myApp.ear -servers server1
deploy the application for the application [/home/user1/apps/myApp.ear] succeeded.

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
====================================================================
+----------+-----------+---------+----------+-----------+-------------------+
| Applicati |Application | State   | Target   | Target    | Application Path  |
|  on ID   | Type       |         | Servers  | Clusters  |                   |
+----------+-----------+---------+----------+-----------+-------------------+
| myApp.ear | EAR       | RUNNING | server1  |           | /home/user1/apps/ |
|          |           |         |          |           |myApp.ear          |
+----------+-----------+---------+----------+-----------+-------------------+

====================================================================
```

4.3.4. Redeploying Applications

If you have modified an application, you can use the redeploy command to redeploy the application with the modifications applied.

Using the Console Tool

The following is the process of redeploying an application by using the console tool.

1. After modifying an application that has been installed in a domain, it needs to be newly installed. As the modified version replaces the previous version, you need to use the -f option when executing the install-application command.

   When deploying the application by specifying the path, replace the previous application file in the specified path with the modified version.

```
[MASTER]domain1.adminServer>install-application -id myApp -f /home/user1/apps/myApp.ear
Successfully installed the application [myApp].
```

2. Redeploy the application using the **redeploy-application** command. When executing the command, you must enter the ID of the application to redeploy.

```
[MASTER]domain1.adminServer>redeploy-application myApp
redeploy application on JEUS Master Server for the application [myApp] succeeded.
```

> For more information about the options available when executing the **redeploy-application** command, see "redeploy-application" in JEUS Reference Guide.

4.3.5. Undeploying Applications

You can stop and undeploy an application in service by using the undeploy command. When undeploy is executed, the application is deleted from the domain.

Using the Console Tool

The following is the process of undeploying an application by using the console tool.

You can undeploy an application using the **undeploy-application** command. When entering the undeploy-application command, you must enter the application ID to undeploy.

```
[MASTER]domain1.adminServer>undeploy-application myApp
Undeploying [myApp] (This may take time due to graceful undeployment) ..........
undeploy the application for the application [myApp] succeeded.
successfully undeployed (elapsed = 82ms)
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+-------+---------+----------+----------------------+
| Applicat | Applicati | State | Target  |  Target  |   Application Path    |
|  ion ID  |  on Type  |       | Servers | Clusters |                       |
+----------+-----------+-------+---------+----------+----------------------+
| myApp    | EAR       | INSTA |         |          | ${INSTALL_HOME}/myApp/ |
|          |           |LLED   |         |          |myApp.ear             |
+----------+-----------+-------+---------+----------+----------------------+
================================================================================
[MASTER]domain1.adminServer>
```

> ◦ In case of undeploying an application that was not installed on the domain, it will be immdiately removed from the list without being displayed in the INSTALLED state.
>
> ◦ For more information about the options available when executing the

> **undeploy-application** command, see "undeploy-application" in JEUS Reference Guide.

4.3.6. Starting Applications

Start an application to run, which is in the DISTRIBUTED status. An application can be in the DISTRIBUTED state, when it is deployed with the **Only Distribute** option. A running application goes into the DISTRIBUTED state, after it's stopped by using the stop command.

Using the Console Tool

The following is the process to start an application in stopped state by using the console tool.

You can start an application by executing the **start-application** command. When executing the start-application command, you must enter the application ID to start.

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+--------+----------+----------+----------------------+
| Applicat | Applicati |  State | Target   | Target   |    Application Path   |
|  ion ID  |  on Type  |        | Servers  | Clusters |                      |
+----------+-----------+--------+----------+----------+----------------------+
| myApp    | EAR       | DISTRI | server1  |          | ${INSTALL_HOME}/myApp/ |
|          |           | BUTED  |          |          | myApp.ear            |
+----------+-----------+--------+----------+----------+----------------------+
================================================================================
[MASTER]domain1.adminServer>start-application myApp
start the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+-----------+--------------------+
| Applicat |Application |  State  | Target   | Target    |  Application Path   |
|  ion ID  |   Type     |         | Servers  | Clusters  |                    |
+----------+-----------+---------+----------+-----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |           | ${INSTALL_HOME}/myA |
|          |           |         |          |           | pp/myApp.ear       |
+----------+-----------+---------+----------+-----------+--------------------+
================================================================================
[MASTER]domain1.adminServer>
```

> ℹ️ For more information about the options available when executing the **start-application** command, see "start-application" in JEUS Reference Guide.

4.3.7. Suspending Applications

If you want to stop the service temporarily without undeploying the application, you can stop the application as follows.

Using the Console Tool

The following is the process of suspending an application service by using the console tool.

You can stop an application service using the **stop-application** command. When executing the stop-application command, you must enter the application ID to be stopped.

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicat |Application |  State  |  Target  |  Target  |  Application Path   |
|  ion ID  |   Type    |         | Servers  | Clusters |                    |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |          | ${INSTALL_HOME}/myA |
|          |           |         |          |          |pp/myApp.ear         |
+----------+-----------+---------+----------+----------+--------------------+
================================================================================
[MASTER]domain1.adminServer>stop-application myApp
stop the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+--------+----------+----------+----------------------+
| Applicat | Applicati |  State |  Target  |  Target  |   Application Path    |
|  ion ID  |  on Type  |        | Servers  | Clusters |                      |
+----------+-----------+--------+----------+----------+----------------------+
| myApp    | EAR       | DISTRI | server1  |          | ${INSTALL_HOME}/myApp/ |
|          |           |BUTED   |          |          |myApp.ear              |
+----------+-----------+--------+----------+----------+----------------------+
================================================================================
[MASTER]domain1.adminServer>
```

> For more information about the options available when executing the **stop-application** command, see "stop-application" in Reference Guide.

4.3.8. Adding a Server as a Target of Running Applications

A server or cluster can be added to scale up service of currently running applications. Because an application that already exists in a domain cannot be deployed on a specific server, JEUS supports the add-target command to add a server as a target. The command can be used when the application is in RUNNING or DISTRIBUTED state.

> To add a server as the application service target, an extra server is needed. For

instructions on how to add an extra server, see "Adding Servers" in JEUS Server Guide.

Using the Console Tool

The following is the process of adding a server as a target for a running application by using the console tool.

You can add a target for a running application using the **add-application-target** command. When running the add-application-target command, you must specify the application ID to add the target to and the list of servers (or clusters) to which you want to add the application.

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+-----------+--------------------+
| Applicat |Application |  State  |  Target  |  Target   |  Application Path   |
|  ion ID  |   Type    |         | Servers  | Clusters  |                    |
+----------+-----------+---------+----------+-----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |           | ${INSTALL_HOME}/myA |
|          |           |         |          |           |pp/myApp.ear        |
+----------+-----------+---------+----------+-----------+--------------------+
================================================================================
[MASTER]domain1.adminServer>add-application-target myApp -servers server2
add a target server or cluster to the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+-----------+--------------------+
| Applicat | Applicati |  State  |  Target  |  Target   |  Application Path   |
|  ion ID  |  on Type  |         | Servers  | Clusters  |                    |
+----------+-----------+---------+----------+-----------+--------------------+
| myApp    | EAR       | RUNNING | server1,s |          | ${INSTALL_HOME}/myA |
|          |           |         |erver2     |          |pp/myApp.ear        |
+----------+-----------+---------+----------+-----------+--------------------+
================================================================================
[MASTER]domain1.adminServer>
```

> For more information about the options available when executing the **add-application-target** command, see "add-application-target" in JEUS Reference Guide.

4.3.9. Removing a Target Server of Running Applications

A server or cluster can be removed from targets of currently running applications. Because an application, which already exists in a domain, cannot be undeployed on a specific server, JEUS supports the remove-target command to remove a server from the targets of the application. The command can be used when the application is in the RUNNING or the DISTRIBUTED state.

Using the Console Tool

The following is the process of removing a server from targets of running applications by using the console.

You can use the **remove-application-target** command to remove the target of a running application. When running the remove-application-target command, you must enter the application ID from which to remove the target and the list of target servers (or clusters) to remove.

```
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicat | Applicati |  State  |  Target  |  Target  |  Application Path   |
|  ion ID  |  on Type  |         | Servers  | Clusters |                    |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR       | RUNNING | server1,s|          | ${INSTALL_HOME}/myA |
|          |           |         |erver2    |          |pp/myApp.ear        |
+----------+-----------+---------+----------+----------+--------------------+
================================================================================
[MASTER]domain1.adminServer>remove-application-target myApp -servers server2
remove server or cluster target from the application for the application [myApp] succeeded.
[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicat |Application |  State  |  Target  |  Target  |  Application Path   |
|  ion ID  |   Type    |         | Servers  | Clusters |                    |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |          | ${INSTALL_HOME}/myA |
|          |           |         |          |          |pp/myApp.ear        |
+----------+-----------+---------+----------+----------+--------------------+
================================================================================
[MASTER]domain1.adminServer>
```

> For more information about the options available when executing the **remove-application-target** command, see "remove-application-target" in JEUS Reference Guide.

4.3.10. Checking Application Information

This section describes how to view application information.

4.3.10.1. Using the Console Tool

Information about myApp.ear application can be checked by using the **application-info** command of the console tool.

The following is an example of checking application information with the console tool. For detailed

explanations of each option, see "application-info" in JEUS Reference Guide.

```
[MASTER]domain1.adminServer> application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicati| Application|  State  |  Server  |  Cluster |  Application Path   |
|  on ID   |   Type    |         | Targets  |  Targets |                    |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |          | ${INSTALL_HOME}/myAp|
|          |           |         |          |          |p/myApp.ear         |
+----------+-----------+---------+----------+----------+--------------------+
================================================================================

[MASTER]domain1.adminServer>application-info -id myApp -server server1
Application information for the server [server1] in the domain [domain1].
================================================================================
+-----------+----------------+------------+------+-----------+-------------+
| Application| Application Name| Application | State|   Server  |   Cluster   |
|    ID     |                |    Type     |      |  Targets  |   Targets   |
+-----------+----------------+------------+------+-----------+-------------+
| myApp     | myApp          | ear         | RUNNI| server1   |             |
|           |                |             |NG    |           |             |
+-----------+----------------+------------+------+-----------+-------------+
================================================================================

[MASTER]domain1.adminServer>application-info -id myApp -server server1 -detail
Application name : myApp
Application [myApp]
==========================================================
+-------------+-------------------------------+-------------+
| Module Name |      Unique Module Name       | Module Type |
+-------------+-------------------------------+-------------+
| ejb         | myApp#ejb                     | EJB         |
| appclient   | myApp#appclient               | CAR         |
| web         | myApp#web                     | WAR         |
+-------------+-------------------------------+-------------+

To view detailed information about EJBs or web modules in an EAR, use the "-module" or "-type"
option.
==========================================================

[MASTER]domain1.adminServer>application-info -id myApp -server server1 -detail -module ejb
Application name : myApp
General information about the EJB module [ejb].
==========================================================
+-------------+---------------------------------------------+
| Module Name |              Unique Module Name             |
+-------------+---------------------------------------------+
| ejb         | myApp#ejb                                   |
+-------------+---------------------------------------------+
==========================================================

Beans
================================================================================
+-----------+----------------------+------------------+-------------------+
| Bean Name |        Type          | Local Export Name | Remote Export Name |
+-----------+----------------------+------------------+-------------------+
```

```
| HelloBean | StatelessSessionBean    |                 |                  |
+-----------+-----------------------+-----------------+------------------+
=================================================================

[MASTER]domain1.adminServer>application-info -id myApp -server server1 -detail -type war
Application name : myApp
There are no EJBs in this module.
General information about the web module [web].
=============================================================
+-------------+-------------------------------+-------------+
| Module Name |      Unique Module Name       | Context Path |
+-------------+-------------------------------+-------------+
| web         | myApp#web                     | /hello      |
+-------------+-------------------------------+-------------+
=============================================================

Servlets
================================================================================
+---------+----------------+-------+-------+-----------+---------+-----------+
|  Name   |     Class      | State | Count | Attribute | RegType |URLPatterns|
+---------+----------------+-------+-------+-----------+---------+-----------+
| HelloSer| dvt.deployment.se| READY |     0 | SYNC      | WEB_XML | /HelloServ|
|vlet     |rvlet.HelloServlet|       |       |           |         |let        |
+---------+----------------+-------+-------+-----------+---------+-----------+
================================================================================

Filters
================================================================================
+------+-------+-------------+----------+---------------------+-------------+
| Name | Class |  Attribute  | RegType  |     URLPatterns     |  Servlets   |
+------+-------+-------------+----------+---------------------+-------------+
(No data available)
================================================================================

Listeners
=============================================
+------+------+-----------------------------+
| Name | Type |           RegType           |
+------+------+-----------------------------+
(No data available)
=============================================

EJBs
================================================================================
+-----------+------+-----------------------+-------------------------------+
| Bean Name | Type |   Local Export Name   |      Remote Export Name       |
+-----------+------+-----------------------+-------------------------------+
(No data available)
================================================================================

[MASTER]domain1.suok>application-info -id myApp -server server1 -detail -module ejb -bean HelloBean
bean HelloBean
Application name : myApp
Module name : ejb
Bean name: HelloBean
================================================================================
+--------------+-----------+-----------------+-------------+---------------+
|     Name     |  (Count)  | WaterMark(High:Low| Bound(Upper:L| Time(Max:Min:T|
|              |           |        :Cur)    |     ower)    |     otal)     |
```

```
+---------------+----------+-----------------+-------------+--------------+
| create        | times(0) |                 |             |              |
+---------------+----------+-----------------+-------------+--------------+
| remove        | times(0) |                 |             |              |
+---------------+----------+-----------------+-------------+--------------+
| timed-rb      | transactio|                |             |              |
|               |n(0)       |                 |             |              |
+---------------+----------+-----------------+-------------+--------------+
| request       | request(0)|                |             |              |
+---------------+----------+-----------------+-------------+--------------+
| active-bean   |          | bean(0:0:0)     |             |              |
+---------------+----------+-----------------+-------------+--------------+
| rolledback    | transactio|                |             |              |
|               |n(0)       |                 |             |              |
+---------------+----------+-----------------+-------------+--------------+
| total-bean    |          | bean(0:0:0)     |             |              |
+---------------+----------+-----------------+-------------+--------------+
| comitted      | transactio|                |             |              |
|               |n(0)       |                 |             |              |
+---------------+----------+-----------------+-------------+--------------+
| MethodReadyCou|          | bean(0:0:0)     |             |              |
|nt             |          |                 |             |              |
+---------------+----------+-----------------+-------------+--------------+
| active-thread |          | thread(0:0:0)   |             |              |
+---------------+----------+-----------------+-------------+--------------+
| total-remote-t|          |thread(100:100:100)|           |              |
|hread          |          |                 |             |              |
+---------------+----------+-----------------+-------------+--------------+

=============================================================================
```

Running the application-info command displays the following information about all the applications in a domain.

Item	Description
Application ID	Application ID. This value must be unique in a domain.
Application Type	Application type. This value is one of the following: ◦ EAR: Application ◦ EJB: EJB module ◦ WAR: Web application module ◦ RAR: Resource adapter module ◦ CAR: Application client module

Item	Description
state	Status of an application in a domain. The value is one of the following: ◦ INSTALLED ◦ DISTRIBUTED ◦ RUNNING ◦ DEPLOYED For detailed information about each state, see Application Status.
Server Target	Target server where the applications are deployed.
Cluster Target	Target cluster where the applications are deployed.
Application Path	Path to the applications installed in the Master Server.

4.4. Staging Mode Deploy

An application in the exploded module format can be deployed to MSs on other machines by compressing the file. This is called a deployment staging mode. To deploy, place the application file in an application repository, or specify the absolute path to the file on the machine where the Master Server resides. For detailed information about deployment in staging mode, see Staging Mode Deploy.

Using the Console Tool

The following is the process of deploying an application in the Staging mode. Use the **deploy** command along with the staging option in the console tool.

```
[MASTER]domain1.adminServer>deploy exploded -servers server1 -staging
deploy the application for the application [exploded] succeeded.

[MASTER]domain1.adminServer>application-info
Application information for the domain [domain1].
================================================================================
+----------+-----------+---------+----------+----------+--------------------+
| Applicati| Application|  State  |  Server  |  Cluster |  Application Path  |
|  on ID   |   Type    |         | Targets  |  Targets |                    |
+----------+-----------+---------+----------+----------+--------------------+
| exploded | WAR       | RUNNING | server1  |          | /home/user1/apps/exp|
|          |           |         |          |          |loded               |
+----------+-----------+---------+----------+----------+--------------------+
| myApp    | EAR       | RUNNING | server1  |          | ${INSTALL_HOME}/myAp|
|          |           |         |          |          |p/myApp.ear         |
+----------+-----------+---------+----------+----------+--------------------+
================================================================================
```

4.5. Deployment by Using a Deployment Plan

A deployment plan is an external configuration file of an application that can be used to modify the application DD at the time of deployment. JEUS defines deployment plan in XML format, and it can be applied to an EJB, web application, EAR standard DD, and JEUS DD. When deploying applications, if you specify to use a deployment plan, the deployment plan configurations and DD will be merged at deployment time to determine the final application configuration settings before starting deployment.

This section describes about deployment plan configurations and operation methods. This section also describes how to deploy applications using a deployment plan.

4.5.1. Configuring a Deployment Plan and Operation Methods

This section describes how to configure a deployment plan and its operation methods through an example.

Deployment Plan Settings

The following example shows how to implement a deployment plan.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jeus-deployment-plan xmlns="http://www.tmaxsoft.com/xml/ns/jeus"
                      xmlns:jeus="http://www.tmaxsoft.com/xml/ns/jeus"
                      xmlns:jakartaee="https://jakarta.ee/xml/ns/jakartaee">
    <descriptors>
        <!-- For standalone EJB -->
        <descriptor>
            <uri>META-INF/ejb-jar.xml</uri>
            <configurations>
                <configuration>
                    <action>REPLACE</action>
                    <xpath>//jakartaee:ejb-name[.='ByeBean']</xpath>
                    <value>
                        <![CDATA[<ejb-name>HiBean</ejb-name>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>DELETE</action>
                    <xpath>/child::jakartaee:ejb-jar/child::jakartaee:enterprise-beans/
 child::jakartaee:session/child::jakartaee:local-home[.='HelloHomeLocal']</xpath>
                </configuration>
                <configuration>
                    <action>APPEND_CHILD</action>
                    <xpath>/child::jakartaee:ejb-jar/descendant::jakartaee:session[2]</xpath>
                    <value>
                        <![CDATA[<transaction-type>Bean</transaction-type>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>INSERT_BEFORE</action>
                    <xpath>/jakartaee:ejb-jar/jakartaee:enterprise-beans/jakartaee:session/
```

```
jakartaee:ejb-name[.='HelloBean']/../jakartaee:transaction-type</xpath>
                    <value>
                        <![CDATA[<session-type>Stateless</session-type>]]>
                    </value>
                </configuration>
            </configurations>
        </descriptor>
        <descriptor>
            <uri>META-INF/jeus-ejb-dd.xml</uri>
            <configurations>
                <configuration>
                    <action>REPLACE</action>
                    <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:export-name[.='ByeBean']</xpath>
                    <value>
                        <![CDATA[<export-name>HiBean</export-name>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>DELETE</action>
                    <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:local-export-name[.='HelloBeanLocal']</xpath>
                </configuration>
                <configuration>
                    <action>APPEND_CHILD</action>
                    <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean</xpath>
                    <value>
                        <![CDATA[<jeus-rmi>false</jeus-rmi>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>INSERT_BEFORE</action>
                    <xpath>/jeus:jeus-ejb-dd/jeus:beanlist/jeus:jeus-bean/
jeus:ejb-name[.='HiBean']/../jeus:jeus-rmi</xpath>
                    <value>
                        <![CDATA[<thread-max>100</thread-max>]]>
                    </value>
                </configuration>
            </configurations>
        </descriptor>
        <!-- For standalone web application -->
        <descriptor>
            <uri>WEB-INF/web.xml</uri>
            <configurations>
                <configuration>
                    <action>REPLACE</action>
                    <xpath>/child::jakartaee:web-app/child::jakartaee:servlet-mapping/
child::jakartaee:servlet-name[.='HiServlet']</xpath>
                    <value>
                        <![CDATA[<servlet-name>HelloServlet</servlet-name>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>DELETE</action>
                    <xpath>//jakartaee:load-on-startup</xpath>
                </configuration>
                <configuration>
                    <action>APPEND_CHILD</action>
                    <xpath>/jakartaee:web-app/descendant::jakartaee:login-config</xpath>
```

```
                    <value>
                        <![CDATA[<auth-method>BASIC</auth-method>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>INSERT_BEFORE</action>
                    <xpath>/jakartaee:web-app/jakartaee:env-entry/
jakartaee:env-entry-value[.='value1']</xpath>
                    <value>
                        <![CDATA[<env-entry-type>java.lang.String</env-entry-type>]]>
                    </value>
                </configuration>
            </configurations>
        </descriptor>
        <descriptor>
            <uri>WEB-INF/jeus-web-dd.xml</uri>
            <configurations>
                <configuration>
                    <action>REPLACE</action>
                    <xpath>//jeus:enable-jsp</xpath>
                    <value>
                        <![CDATA[<enable-jsp>true</enable-jsp>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>DELETE</action>
                    <xpath>/jeus:jeus-web-dd/child::jeus:max-instance-pool-size</xpath>
                </configuration>
                <configuration>
                    <action>INSERT_BEFORE</action>
                    <xpath>/jeus:jeus-web-dd/descendant::jeus:enable-jsp</xpath>
                    <value>
                        <![CDATA[<context-path>/hello</context-path>]]>
                    </value>
                </configuration>
                <configuration>
                    <action>APPEND_CHILD</action>
                    <xpath>//jeus:jeus-web-dd</xpath>
                    <value>
                        <![CDATA[<webinf-first></enabled>false</enabled></webinf-first>]]>
                    </value>
                </configuration>
            </configurations>
        </descriptor>
        <!-- For EAR -->
        <descriptor>
            <uri>META-INF/application.xml</uri>
            <configurations>
                <configuration>
                    <action>REPLACE</action>
                    <xpath>/jakartaee:application/jakartaee:library-directory</xpath>
                    <value>
                        <![CDATA[<library-directory>mylib</library-directory>]]>
                    </value>
                </configuration>
            </configurations>
        </descriptor>
        <descriptor>
            <uri>ejb.jar/META-INF/ejb-jar.xml</uri>
```

```
            <configurations>
                <configuration>
                    <action>REPLACE</action>
                    <xpath>/jakartaee:ejb-jar/jakartaee:enterprise-beans/
jakartaee:session/jakartaee:ejb-class</xpath>
                    <value>
                        <![CDATA[<ejb-class>HelloBean</ejb-class>]]>
                    </value>
                </configuration>
            </configurations>
        </descriptor>
            <descriptor>
            <uri>web.war/WEB-INF/web.xml</uri>
            <configurations>
                <configuration>
                    <action>REPLACE</action>
                    <xpath>/jakartaee:web-app/jakartaee:servlet-mapping/
jakartaee:servlet-name</xpath>
                    <value>
                        <![CDATA[<servlet-name>HelloServlet</servlet-name>]]>
                    </value>
                </configuration>
            </configurations>
        </descriptor>
    </descriptors>
</jeus-deployment-plan>
```

The following describes the tags under <descriptor>.

- <descriptor>

  The <descriptor> tag consists of multiple <configurations> child tags.

  ◦ <configuration>

  The <configuration> tag is a unit of configuration that applies to target DD. It consists of
  <action>, <xpath>, and <value> child tags.

Tag	Description
<action>	Specifies which configuration will be changed for a certain tag of DD. The configuration values are as follows: ◦ DELETE: Deletes the specified tag from DD. ◦ REPLACE: Replaces the specified tag with a tag in DD. ◦ APPEND_CHILD: Adds a tag as the last child in DD. ◦ INSERT_BEFORE: Adds a tag as a previous sibling in DD.

Tag	Description
\<xpath\>	Configures the specified DD tag using an xpath expression. In this case, all tags, which are placed in the \<xpath\> path, must be expressed as qualified names with XML namespace according to the \<xpath\> standard. Thus, you must declare XML Name Space in each DD with a namespace prefix in the deployment plan. For example, if a tag in JEUS DD is set as \<xpath\>, all tags in the path \<xpath\> should be expressed as qualified names with JEUS XML namespace (http://www.tmaxsoft.com/xml/ns/jeus). Since JEUS XML namespace is mapped to the 'jeus' prefix in the deployment plan, you can form qualified names by attaching the 'jeus' prefix in front of all tag names that are placed in the \<xpath\> path.
\<value\>	The \<value\> tag is valid only when the \<action\> tag value is REPLACE, APPEND_CHILD, or INSERT_BEFORE. The configuration values according to the \<action\> value are as follow: ◦ REPLACE: Sets a new tag to replace the current \<xpath\> tag. ◦ APPEND_CHILD: Sets a tag, which will be added as the last child to the \<xpath\> tag. ◦ INSERT_BEFORE: Sets a tag, which will be added as a previous sibling of the \<xpath\> tag. Since the tag set to \<value\> assumes the form of an XML fragment with depth, the \<value\> value is wrapped in CDATA section. Construct the XML fragment necessary for performing an action in the CDATA section.

Deployment Plan Operation Methods

This section describes deployment plan operations by tags with the previous example.

• \<descriptor\>

A deployment plan consists of multiple \<descriptor\> tags.

Use the \<uri\> tag to specify the target DD, which becomes a target in units of \<descriptor\> tag. Using the application file as the root, specify \<uri\> value as a relative path to DD to determine the target DD.

◦ For example, the standard DD of a standalone EJB module is always placed in META-INF/ejb-jar.xml and JEUS DD in META-INF/jeus-ejb-dd.xml from the application root. Therefore the first \<descriptor\> tag of the previous deployment plan is for the standard DD of a standalone EJB module obtained from the \<uri\> value. The second \<descriptor\> tag is for JEUS DD of a standalone EJB module obtained from the \<uri\> value. The same rule applies to standalone web applications or EAR.

◦ If the \<uri\> value is 'ejb.jar/META-INF/ejb-jar.xml', the \<descriptor\> value is an EJB module (file

name is ejb.jar), which belongs to EAR. If the <uri> value is 'web.war/WEB-INF/web.xml', the <descriptor> value is a WEB module (file name is web.war), which belongs to EAR.

  ◦ One deployment plan can be used to deploy various applications. Only the <descriptor>'s, which match the <uri> value of application DD, are selected for deployment. Thus, other <descriptor>'s have absolutely no effect on deployment.

- <configuration>

  The following example uses the first <descriptor> configuration to describe how DD is changed according to the <descriptor> configuration.

  ◦ As explained earlier, the first <descriptor> tag configures the standard DD of a standalone EJB module. The first action performed is replacing a specific tag and is expressed in the first <configuration> tag. It shows that the <ejb-name> tag with the value, 'ByeBean', is replaced with the <ejb-name> tag with the value, 'HiBean'.

  ◦ The second <configuration> expresses the action of deleting a <local-home> tag with the value, 'HelloHomeLocal'.

  ◦ The third <configuration> expresses the action of adding a <transaction> tag as the last child of the second <session> tag.

  ◦ The fourth <configuration> expresses the action of adding a <session-type> tag as a previous sibling of a <session> tag, whose <ejb-name> tag value is 'HelloBean'.

4.5.2. Installing Deployment Plan

Before deploying applications using deployment plan, the deployment plan should be installed on a domain. Only the deployment plans that are installed on a domain can be deployed like applications. When installing deployment plan, the identifier of the deployment plan can be set on a domain.

A deployment plan can be installed by using the console tool.

Using the Console Tool

In the console tool, execute the **install-deployment-plan** command to install the deployment plan.

```
[MASTER]domain1.adminServer>install-deployment-plan -path /home/user1/plans/jeus-deployment-plan.xml
-name plan1
Installing the deployment plan [plan1] was successful.
```

> For more information on how to use the install-deployment-plan command, see "install-deployment-plan" in JEUS Reference Guide.

4.5.3. Verifying an Installed Deployment Plan

Users can check the installed deployment plan by using the console tool. Each deployment plan is distinguished by the deployment plan identifier that is assigned during the installation. You can also display the list of applications by deployment plan that is applied to the application, as well as the actual deployment file content.

Using the Console Tool

By using the **deployment-plan-info** command in the console tool, users can obtain the list of applications, to which deployment plans installed on a domain apply to. If a particular deployment plan is selected, its details are displayed.

- Checking the deployment plan list

```
[MASTER]domain1.adminServer>deployment-plan-info
The list of deployment plans installed in the domain and the applications to which each
deployment plan applies
=========================================================
+---------------------------------------+-------------+
|             Deployment plan            | Applications |
+---------------------------------------+-------------+
| plan1                                 |             |
+---------------------------------------+-------------+
=========================================================
```

- Checking the details of a deployment plan

```
[MASTER]domain1.adminServer>deployment-plan-info -name plan1
<?xml version="1.0" encoding="UTF-8"?>
<jeus-deployment-plan xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
    ...
</jeus-deployment-plan>
```

> For more information on how to use the **deployment-plan-info** command, see "deployment-plan-info" in JEUS Reference Guide.

4.5.4. Deployment by Using a Deployment Plan

Applications can be deployed using an installed deployment plan by using the console tool.

Using the Console Tool

In a console tool, applications can be deployed using a deployment plan by executing the **deploy-application** command.

```
[MASTER]domain1.adminServer>deploy-application webapp -all -plan plan1
deploy the application for the application [webapp] succeeded.
```

4.5.5. Verifying an Application-Applied Deployment Plan

Users can verify the deployment plan that was applied to an application using the console tool.

Using the Console Tool

In the console tool, a deployment plan that has been applied to an application can be checked by executing the **application-info** command.

```
[MASTER]domain1.adminServer>application-info -id webapp -detail
Application information for the domain [domain1].
================================================================================
+------+-------+-----+------+-------+-------------+---------+---------+-----+
| Appli| Applic|State|Server|Cluster|   Running   | Applicati| Applicati| Plan|
|cation| ation |     |Target|Targets|   Servers   | on Path | on Time |Name |
|  ID  | Type  |     |   s  |       |             |         |         |     |
+------+-------+-----+------+-------+-------------+---------+---------+-----+
|webapp| WAR   | RUNN| ALL  | ALL   | server1,serve| ${INSTALL| Tue May |plan1|
|      |       | ING |      |       |    |r2,server3,adm|HOME}/web|28     |     |
|      |   |   |   |   |   |inServer   |app/deploy|22:45:13 |   |
|      |   |   |   |   |   |ment_plan|KST 2013 |    |
|      |   |   |   |   |   |             |web.war   |         |     |
+------+-------+-----+------+-------+-------------+---------+---------+-----+
================================================================================
```

4.5.6. Uninstalling Deployment Plan

Users can uninstall a deployment plan on a domain by using the console tool. Once uninstalled, the deployment plan is no longer valid and thus cannot be used for deployment.

Using the Console Tool

In the console tool, deployment plans can be uninstalled by executing the **uninstall-deployment-plan** command.

```
[MASTER]domain1.adminServer>uninstall-deployment-plan plan1
Uninstalling the deployment plan was successful.
```

> For more information on how to use the uninstall-deployment-plan command, see "uninstall-deployment-plan" in JEUS Reference Guide.

4.5.7. Redeploying Deployment Plan

If an application was deployed using a deployment plan, the deployment plan is automatically applied by default when the application is redeployed. However, if a new deployment plan is specified for redeployment, the new deployment plan is applied.