

Node Manager Guide

JEUS 9.1

TMAXSOFT

Copyright

Copyright 2025. TmaxSoft Co., Ltd. All Rights Reserved.

Company Information

TmaxSoft Co., Ltd.

TmaxSoft Tower, 45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, South Korea

Website: <https://www.tmaxsoft.com/en/>

Restricted Rights Legend

All TmaxSoft Software (JEUS®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd. Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features.

This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

Trademarks

JEUS® is registered trademark of TmaxSoft Co., Ltd.

Java, Solaris are registered trademarks of Oracle Corporation and its subsidiaries and affiliates.

Microsoft, Windows, Windows NT are registered trademarks or trademarks of Microsoft Corporation.

HP-UX is a registered trademark of Hewlett Packard Enterprise Company.

AIX is a registered trademark of International Business Machines Corporation.

UNIX is a registered trademark of X/Open Company, Ltd.

Linux is a registered trademark of Linus Torvalds.

Noto is a trademark of Google Inc. Noto fonts are open source. All Noto fonts are published under the SIL Open Font License, Version 1.1. (<https://www.google.com/get/noto/>)

Other products and company names are trademarks or registered trademarks of their respective

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 9.1	3.3.1	2025-12-10	-
JEUS 9	3.1.2	2025-03-24	-
JEUS 9	3.1.1	2024-12-24	-

Table of Contents

1. Introduction	1
1.1. Overview	1
1.1.1. Concepts	1
1.1.2. Purpose	1
1.1.3. Types	2
1.1.4. Constraints	3
1.2. Configuring Node Managers by Type	3
1.3. Common Functions	3
1.4. Managing RQS Processes with Node Manager	4
2. Java Node Manager	6
2.1. Overview	6
2.2. Using the Java Node Manager Functions	6
2.2.1. Starting Managed Servers on a Remote Machine	6
2.2.2. Monitoring Servers	7
2.2.3. Restarting a Server in an Abnormal State	8
2.2.4. Rolling Patch	9
2.2.5. Connecting to Master	10
2.3. Configuring the Environment	10
2.3.1. Configuration File	11
2.3.1.1. nodes.xml	11
2.3.1.2. jeusnm.xml	11
2.3.1.3. <serverName>.properties	12
2.3.2. Required Files	13
2.4. Configuring and Deleting Nodes	14
2.4.1. Configuring a Java Node	14
2.4.1.1. Using the Console Tool	14
2.4.2. Deleting a Java Node	16
2.4.2.1. Using the Console Tool	16
2.5. Starting and Terminating the Node Manager	16
2.5.1. Starting the Java Node Manager	16
2.5.2. Terminating the Java Node Manager	17
2.6. Controlling Servers Using the Java Node Manager	18
2.6.1. Using the Console Tool	18
2.7. Log Files	21
3. SSH Node Manager	22
3.1. Overview	22
3.2. Configuring the SSH Environment	22
3.2.1. Configuring SSH	22

3.3. Configuring and Deleting Nodes	24
3.3.1. Configuring an SSH Node	24
3.3.1.1. Using WebAdmin	24
3.3.1.2. Using the Console Tool	25
3.3.2. Modifying an SSH Node	28
3.3.3. Deleting an SSH Node	29
4. Reliable Queue Server (RQS) Process Management	30
4.1. Overview	30
4.2. Operation of RQS Processes	30
4.2.1. Starting and Terminating RQS Processes	30
4.2.2. Monitoring RQS Processes	31
4.2.3. Restarting RQS Processes in Abnormal Status	31
4.3. Configuring the Environment	31
4.3.1. jeusnm.xml	31
5. Replicating Node Manager	33
5.1. Overview	33
5.2. Operating a Replicated Node Manager	33
5.3. Using a Replicated Node Manager	34
5.3.1. Standby Node Manager	34
5.4. Terminating a Replicated Node Manager	34

1. Introduction

This chapter introduces node managers and describes their common functions.

1.1. Overview

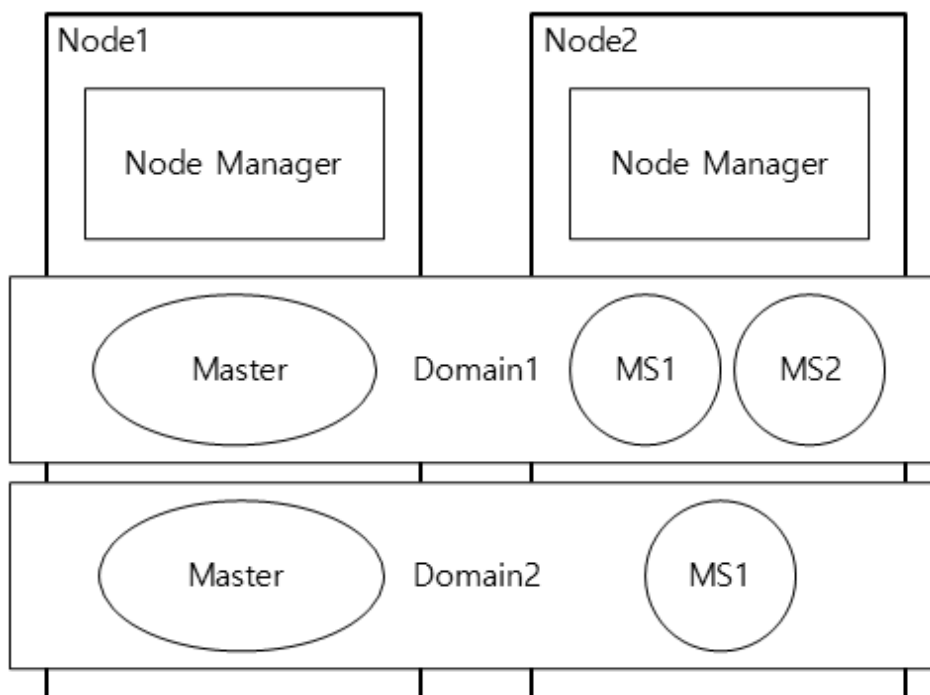
This section describes the basic concepts, purposes, types, and constraints of node managers.

1.1.1. Concepts

The servers that make up a domain can operate from multiple machines, and multiple domains can exist on a single machine. Only one JEUS instance can be installed on a machine, and a JEUS instance can only have one node manager. In this environment, a node manager manages server processes on the machine regardless of the domains. In other words, the node manager manages servers by machine instead of domain. It starts, terminates, or restarts the servers on a machine to provide services.

A JEUS that is installed on a machine is defined as a node. A node is defined as unique by using the address of the machine and the JEUS installation directory path.

The following shows the relationship between nodes, domains, and node managers.



JRelationship between JEUS, Domains, and Node Managers

1.1.2. Purpose

The main purpose of the node manager is to manage server processes that run on a node.

The node manager can start the servers on remote machines where Master Server (hereafter Master) is not running. The node manager can restart a server if the server is abnormally terminated or is in an abnormal state that you have defined. If a server suddenly shuts down, Master can detect this using System Clustering Framework (SCF) and restart it.

One instance of node manager exists per node. The instance can receive and apply a server patch. The servers managed by Master can be started using the console tool (jeusadmin) and start-server command through the node manager. In addition, the node manager manages external processes (currently RQS processes) as well as server processes by using a configuration file; it allows RQS processes to start, terminate, or restart when they were terminated abnormally. These operations are independent of a server process.



A node manager is an optional function that is not required of the server to provide services. However, it is recommended to use the node manager for stable operation of domains.

1.1.3. Types

The following are the two types of node managers provided by JEUS.

- Java node manager

It is implemented in Java, so it can run regardless of the OS.

It can start managed servers (MSs) that are on the nodes where Master is not running. Since the Java node manager detects the server state and abnormal terminations, it can restart a problematic server faster than the SSH node manager. It can also apply patches to the domain. You can use the Java node manager only when JEUS is installed. You cannot install the Java node manager where JEUS is not installed. For more information about the Java node manager, refer to [Java Node Manager](#).

- SSH node manager

It uses an SSH that is provided by the OS. Because SSH is not provided by Windows, if you want to use a node manager in Windows, you need to use the Java node manager.

Like the Java node manager, the SSH node manager can start servers on remote machines, and detect an abnormal termination to restart the server. However, since the SSH node manager does not monitor server processes directly, it is the Master, not the SSH node manager, that detects the abnormal server state by using the SCF. The Master then restarts the remote servers through the SSH node manager. The difference between the SSH node manager and the Java node manager is that JEUS can be installed on a different machine when using the SSH node manager. For more information about the SSH node manager, refer to [SSH Node Manager](#).

1.1.4. Constraints

The following are the constraints for configuring the node manager.

- The node manager name must be unique within a domain.
- A JEUS instance can have only one node manager.
- Since SSH is not provided if the OS is Windows, the Java node manager must be used.
- To use the Java node manager, it is recommended to register it as a service provided by the OS.
- To use the Java node manager, "**useNodeManager**" in the jeusnm.xml file must be set to "true".

1.2. Configuring Node Managers by Type

Node managers can be configured using the console tool.

- **Configuring the SSH node manager**

Use the '**add-ssh-node**' command in the **jeusadmin** tool. Enter the node name and host information and configure the required items for the SSH type. For more information about how to configure the SSH node manager, refer to [Configuring SSH](#).

- **Configuring the Java node manager**

Use the '**add-java-node**' command in the **jeusadmin** tool. Enter the node name, and host information and then configure the required items for Java type. For more information about how to configure the Java node manager, refer to [Configuring a Java Node](#).

1.3. Common Functions

The following functions are commonly provided by the Java node manager and the SSH node manager.

- Starting servers on remote machines

You can start servers that belong to a domain by using the node manager.

If you do not use a node manager, you need to start the servers using a script by accessing the remote machine. But if you use a node manager, you can start the servers on remote machines by using a command from Master. If Master is running, all MSs or clusters in the domain can be started.

The following conditions must be satisfied to start servers using the node manager.

- The node information for starting the servers must be registered in nodes.xml.
- The information about which nodes can start the server must be configured in the server.
- Must be able to access the node manager.

The Java node manager must be running in order to use it. If the SSH node manager is used, the SSH port must be open on the OS.

- When the Java node manager is used, the node manager must be configured in the jeusnm.xml file.
- Restarting a server in abnormal state

When a server that the Java node manager has been monitoring shuts down abnormally, the process must be restarted. For the SSH node manager, If a server goes into an abnormal state (failed) by the SCF of the domain, Master sends the restart command for the server to the server's node manager. The node manager that receives the command from Master restarts the server. To restart the server that has failed, the node must be configured on the server, and the node manager must be running if the Java node manager is being used. If Master is also in an abnormal state when a managed server restarts, the node manager receives the configuration file from the external repository configured in the managed server so that the server restarts.



For more information about the functions provided by each node manager type, refer to [Java Node Manager](#) and [SSH Node Manager](#).

1.4. Managing RQS Processes with Node Manager

A node manager manages the Reliable Queue Server (RQS) processes as well as server processes. It checks the status of RQS processes and restarts an RQS process that was abnormally terminated.

- Starting and managing RQS processes that are set in a node

A node manager can start RQS processes by using configured RQS information and continuously monitor their status by using a configured port. If the node manager detects that an RQS process has terminated abnormally, it will restart it for continuous service.

To start an RQS process by using a node manager, the following are required:

- The RQS process information must be registered in jeusnm.xml.
- A configuration file used to start the RQS processes must be in a correct location, and the path and port of the node manager must be configured appropriately in the file. Since the RQS process refers to the file location according to the RQSDIR environment variable, this variable must be set appropriately.
- The node manager must be available for connection.
- Restarting RQS processes in abnormal status

One major role of a node manager is checking RQS processes and restarting an RQS process that was terminated abnormally. For this, the node manager and RQS processes send and receive messages through a configured port.

When a node manager detects that an RQS terminates abnormally, it restarts the RQS process by using configured information and the -r option which cleans shared resources that were being

used by the RQS process.



For more information about how to manage RQS processes, refer to [Reliable Queue Server \(RQS\) Process Management](#).

2. Java Node Manager

This chapter describes how to configure and operate the Java node manager.

2.1. Overview

The Java node manager can detect the abnormal termination of a server faster than the SSH node manager. The SSH node manager cannot be used if the OS is Windows. The Java node manager can be used regardless of the OS.

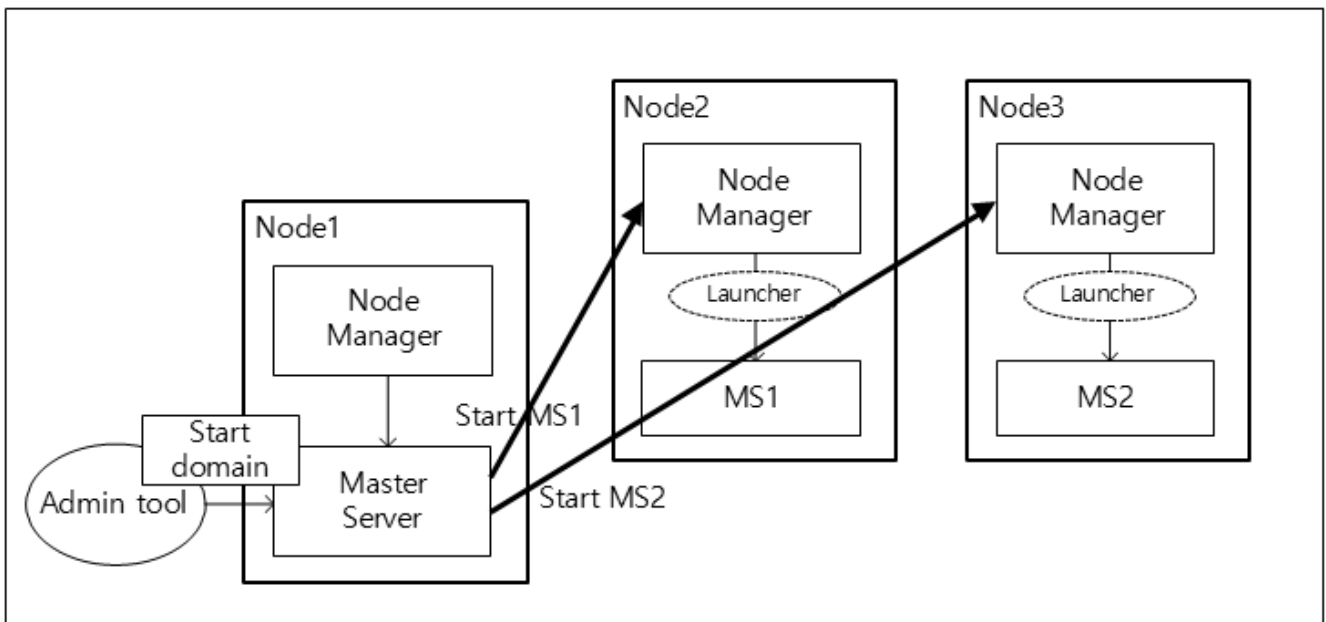
This chapter describes the basic knowledge that is required to use the Java node manager.

2.2. Using the Java Node Manager Functions

This section describes how each Java node manager function works.

2.2.1. Starting Managed Servers on a Remote Machine

The following is the process of starting servers on remote machines in the domain through the node manager.



Starting Servers on Remote Machines via Node Managers

1. The node manager creates the lock file of the server and gets the FILE LOCK.
2. After starting the server, it creates the pid and state files, and then records the PID (process ID) and server state.
3. It updates the username, password, and masterurl in the `<serverName>.properties` file.

4. The server starts the NodeManagerService, creates the address file, and records its host information.



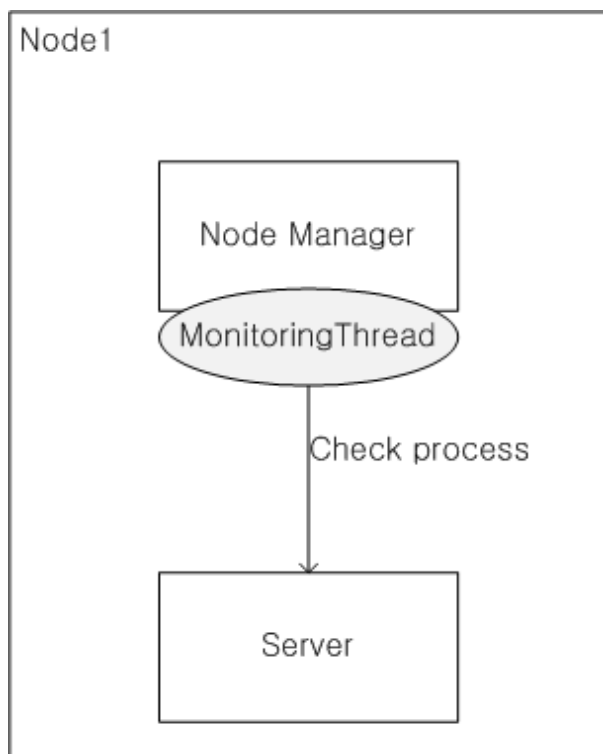
The same process is used to restart the server.

2.2.2. Monitoring Servers

The Java node manager monitors the state of server processes regularly to update the state file. If an error occurs, the node manager restarts the server process.

The node manager does not monitor all the servers on a node, but only the server processes that were started through the node manager. Therefore, to restart a server immediately in case of an error occurrence, the server must be started by the node manager. Even when a node manager is restarted, it continues to monitor the server that it started.

The following shows the process of the Java node manager monitoring a server.



Monitoring a Server using the Java Node Manager

The following shows how a node manager finds the server that it was monitoring after it is restarted.

1. When a node manager is restarted, the node manager finds the domain names that are managed by the node manager in the following path.

JEUS_HOME/domains

2. The node manager finds the server names that are managed by the node manager in the server

directory in the following path.

```
JEUS_HOME/domains/<domain-name>/servers
```

3. It checks if the files `<serverName>.address`, `<serverName>.lck`, `<serverName>.state`, `<serverName>.pid` are in the server directory. If all the files exist, then it knows that the server is running.
4. It gets FILE LOCK from the lock file.
5. It checks the server state in the state file and determines whether the server can run the services.
 - A service can run if it is in one of the following states: RUNNING, STANDBY, SUSPENDED, RESUMING, SUSPENDING, and STARTING
 - A service cannot run if it is in one of the following states: SHUTDOWN and SUNTTING_DOWN
6. It checks if the process is running with the server process ID in the pid file. If the server is down, the node manager restarts the server.
7. It makes a socket connection based on the host information of the server that is recorded in the address file and checks if the server process is running normally. If the server is down, the node manager restarts the server.
8. If it is determined that the server is ready to run the services, the node manager monitors the server by regularly checking the server state.

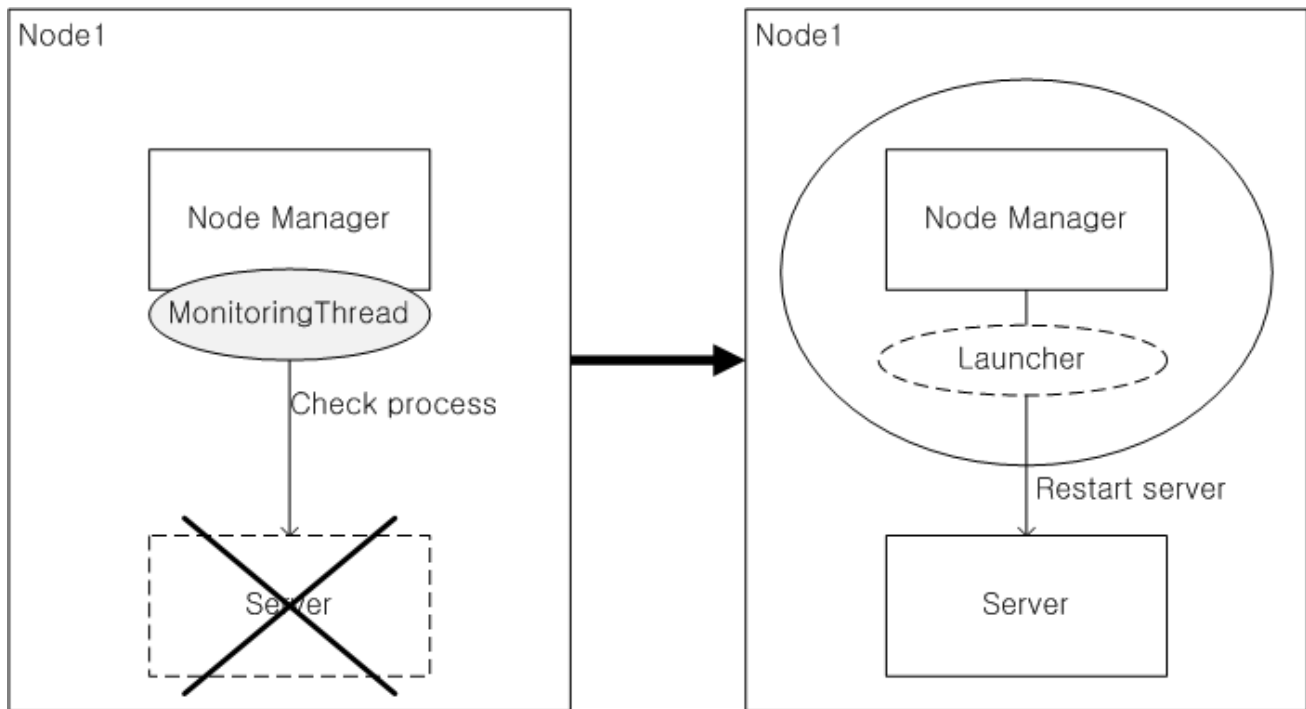


The SHUTDOWN_NEED_TO_RESTART state in the state file is used to restart the server, which was previously running on the node, when the node manager restarts. It is usually used to restart the previously running server after a server shutdown when the node manager is registered as an OS service, like a Windows service.

2.2.3. Restarting a Server in an Abnormal State

The Java node manager restarts the server processes if the server processes were terminated abnormally, and also if the server processes go into an abnormal state.

Each server sets a memory ratio, and if heap memory is used more than the specified ratio, the server terminates its own process. The node manager detects this and restarts the server.



Restarting the Server from the Java Node Manager

2.2.4. Rolling Patch

A patch can be applied to a domain by using the Java node manager. The node manager can send a patch file to the servers on remote machines or can delete the patch file. The node manager can also restart the servers on remote machines to apply the patch.

Master can access node managers on remote machines to send, delete, or apply a patch file.

- Sending a patch file

```
domain1.adminServer>apply-patch
```

- Deleting a patch file

```
domain1.adminServer>remove-patch
```

- Applying a patch file

The patch can be applied to each server using the `-rolling` option. When the option is used, the patch is applied to each node in order to prevent errors that may occur for replicated services. If multiple servers are operating on the same node, each server must be terminated in order and after applying the patch file, the servers must be restarted one by one.

```
domain1.adminServer>apply-patch -rolling
domain1.adminServer>remove-patch -rolling
```

- Reverting if installing a patch fails.

The `-action` option can be used to define an action for a failed patch.

```
domain1.adminServer>apply-patch -rolling -action CONTINUE|STOP|ROLLBACK
```

Use one of the following options (CONTINUE, STOP, or ROLLBACK).

Option	Description
CONTINUE	Option to proceed even if applying the patch on a node has failed.
STOP	Option to stop the patch installation and return if applying the patch on a node has failed.
ROLLBACK	Option to stop the patch installation and rollback the successfully applied patches if applying the patch on a node has failed. When deleting the patches, there might be nodes from which the patch has already been deleted. In this case, even when this option is set, deleting the patch will stop as with the STOP option.

2.2.5. Connecting to Master

The servers that were started without using the node manager are not managed by the node manager, and the monitoring or patch functions are not provided for them.

For Master, however, the process can be monitored and the patch can be delivered even if it was started without using the node manager. When Master starts, it connects to and sends its information to the node manager, and the node manager can use this information to monitor Master.

The rolling patch upgrade is possible only when Master is connected to the node manager. Although Master was not started using the node manager, since it is connected to the node manager the patch can be sent to other nodes as well as to its own node.



The servers that are not started by the node manager cannot be monitored. If the Java node manager is used and a server is started by the node manager, the node manager detects any abnormal terminations and restarts the server before it goes into the FAILED state.

2.3. Configuring the Environment

This section describes how to configure the environment needed for using the Java node manager.

2.3.1. Configuration File

The following files must be configured in order to use the Java node manager which requires other configuration in addition to the nodes.xml file.

- [nodes.xml](#)
- [jeusnm.xml](#)
- [<serverName>.properties](#)

2.3.1.1. nodes.xml

Configure a node manager type to use in the nodes.xml file that is located in the following path.

```
JEUS_HOME/domains
```

The nodes.xml, which is referenced by Master, defines the physical address of the node. The information of the node where each MS is running is defined in the domain configuration. This file is shared by multiple domains where the installed JEUS belongs to. Master in each domain accesses the node manager of each node according to the configuration in the node.xml file.

Nodes can be added or deleted using the console tool. For more information, refer to [Configuring and Deleting Nodes](#).



The machine settings in the nodes.xml file and the **"host"** and **"port"** settings in the jeusnm.xml file must match.

2.3.1.2. jeusnm.xml

The jeusnm.xml is a configuration file that defines how the node manager works, and it is in the following path.

```
JEUS_HOME/nodemanager
```

The following are the configuration items that can be set in the jeusnm.xml file which must be modified manually.

Item	Description
useNodeManager	Option to use the Java node manager. (Default: true)
host	Listen address of the node manager. (Default: localhost)
port	Listen port of the node manager. (Default: 7730)
serverMonitoringPeriod	Monitoring cycle for the server process state. (Default: 500, Units: ms)

Item	Description
serverAutoRestart	Option to reboot the server automatically if the server process has been terminated abnormally. (Default: true)
serverRestartTryCount	<p>Retry count for server restart. (Default: 5)</p> <p>If the attempts to restart the server exceeds this value during the time set in 'serverRestartDurationTime', it is determined that there's a problem with the server process, and server restart will no longer be attempted.</p>
serverRestartDurationTime	<p>Server restart cycle. (Default: 120000, Units: ms)</p> <p>If the number of server restart attempts during this cycle exceeds the value set in 'serverRestartTryCount' then it is determined that there is a problem with the server process and server restart will no longer be attempted.</p>
serverRetryRestart	Option to retry restarting the server when the attempt to restart the server fails. If set to true, the server is restarted until it starts successfully. (Default: false)
useSSLListener	Whether the node manager uses SSL. (Default: false)
keystoreFile	Path to the keystore file that is used during authentication when SSL is used. (Default: NODEMANAGER_HOME/keystore)
keystorePass	Password of the keystore file that is used during authentication when SSL is used. (Default: jeuskeypass)
truststoreFile	Path to the truststore file that is used during authentication when SSL is used. (Default: NODEMANAGER_HOME/truststore)
truststorePass	Password of the truststore file that is used during authentication when SSL is used. (Default: jeustruststorepass)
logFileName	Log file location of the node manager. If the location is an absolute path, a log file will be created in the path. If the location is a relative path, a log file will be created in a default path. (Default: JEUS_HOME/nodemanager/logs/JeusNodeManager.log)
standbyPort	Enables to use a node manager's standby process when a node manager is terminated abnormally by configuring a port to use for this. For more information, see Standby Node Manager .
processList	List of information about RQS processes. For more information, see Configuring the Environment .

2.3.1.3. <serverName>.properties

The <serverName>.properties file is the cache file that stores the information that is required to restart the server. The file is created in the following path after the server is restarted by the node manager.

```
SERVER_HOME/nodemanager
```

The node manager can store the options that are required to start the server in a file and can use it to start the server. This information is used only when the node manager starts the server directly using the console tool. The options are not used when an MS is started by Master.

The following are the cached items.

Item	Description
masterurl	Host information of Master. It is cached only for MSs.
username	Server account that is required to start the server.
password	Server password that is required to start the server.
sslArguments	SSL properties information that is required for SSL authentication when the node manager accesses the server and the server uses SSL.

2.3.2. Required Files

The following describes the additional files that are required for the node manager to start and monitor the server. Each file is located in the following path.

```
JEUS_HOME/domains/<domain_name>/servers/<server_name>/nodemanager
```

File	Description
<serverName>.lck	<p>This file is created by the node manager to prevent the server process from being started multiple times. This file is created before the node manager starts the server process and is deleted when the server process terminates successfully.</p> <p>When the node manager is restarted, it determines whether to keep monitoring the server based on the existence of this file.</p>
<serverName>.pid	<p>If the server process is restarted successfully, the node manager creates this file to store the server process ID. This file is created when the node manager starts the server process successfully and is deleted when the server process is terminated successfully.</p> <p>When the node manager is restarted, it determines whether to keep monitoring the server, depending on the existence of this file. Also, it can be determined whether the server is abnormally terminated depending on the existence of the PID process in the file. When the node manager is restarted, the server can be restarted and monitored (by the node manager) if the server has been terminated abnormally.</p>

File	Description
<code><serverName>.state</code>	<p>This file is created by the node manager to record the server state after the server process is started successfully. This file is created when the node manager starts the server process successfully and is deleted when the server process is terminated successfully. The node manager regularly checks and updates the server state in the file.</p> <p>When the node manager is restarted, it determines whether to keep monitoring the server depending on the existence of this file. When the node manager is restarted, the server state recorded in the file can be used to determine if the server has been terminated abnormally and whether to keep monitoring the server.</p>
<code><serverName>.address</code>	<p>This file is created when the server is started. It records the Listen Address and the Listen Port of the server. If the server is terminated successfully, this file will be deleted.</p> <p>When the node manager is restarted, it determines whether to keep monitoring the server depending on the existence of this file and the host information in the file can be used to continue to monitor the server. If the node manager is restarted, it sends an MBean request to the server to regularly monitor and detect abnormal termination of the server.</p>

2.4. Configuring and Deleting Nodes

The console tool is used to configure or delete Java nodes.

2.4.1. Configuring a Java Node

This section describes how to use the console tool to configure a Java node.



To start a server using the console tool, the node name must be configured on the server.

2.4.1.1. Using the Console Tool

The following shows how to configure a node using the console tool.

1. Add a node named node1 by using the **add-java-node command**.

```
[MASTER]domain1.adminServer>add-java-node node1 -host 192.168.34.65 -port 7730
The node [node1] was successfully added.
```

2. Check the configuration of the added node by using the **show-node** command. The following Java node "node1" is displayed.

```
[MASTER]domain1.adminServer>show-node node1
=====
+-----+-----+
| Property | Value |
+-----+-----+
| Node Name | node |
| Host | 192.168.34.65 |
| Mapped Servers | |
| Node Type | JAVA |
| NodeManager Port | 7730 |
| Use SSL | false |
+-----+-----+
=====
```

To check the nodes in the domain, use the **list-nodes** command.

```
[MASTER]domain1.adminServer>list-nodes
=====
+-----+-----+-----+-----+
| Node name | Type | Under control | JEUS version |
+-----+-----+-----+-----+
| node1 | JAVA | N | - |
+-----+-----+-----+-----+
=====
```

3. To modify a node, use the **modify-java-node** or **modify-node** command.

```
[MASTER]domain1.adminServer>modify-java-node node1 -port 7731
The node [node1] was modified successfully. Check the results using "show-node"
[MASTER]domain1.adminServer>modify-node node1 -port 7732
The node [node1] was modified successfully. Check the results using "show-node"
```

4. To check the modified node information, use the **show-node** command.

```
[MASTER]domain1.adminServer>show-node node1
=====
+-----+-----+
| Property | Value |
+-----+-----+
| Node Name | node |
| Host | 192.168.34.65 |
| Mapped Servers | |
| Node Type | JAVA |
| NodeManager Port | 7732 |
| Use SSL | false |
+-----+-----+
=====
```



For more information about the `add-java-node` and `list-nodes` commands, refer to "Node Management Commands" in *JEUS Reference Guide*.

2.4.2. Deleting a Java Node

This section describes how to delete a Java node using the console tool.

2.4.2.1. Using the Console Tool

The following shows how to delete a node using the console tool.

1. Delete the node "node1" using the `remove-node` command.

```
[MASTER]domain1.adminServer>remove-node node1
The node [node1] was successfully removed.
```

2. To check if the node has been deleted successfully, check the list of nodes using the `list-nodes` command. The Java node "node1" should no longer be in the list.

```
[MASTER]domain1.adminServer>list-nodes
=====
+-----+-----+-----+-----+
| Node name | Type | Under control | JEUS version |
+-----+-----+-----+-----+
(No data available)
=====
```



For more information about the `remove-node` and `list-node` commands, refer to "Node Management Commands" in *JEUS Reference Guide*.

2.5. Starting and Terminating the Node Manager

Unlike the SSH node manager, the Java node manager requires a separate starting and terminating process.

2.5.1. Starting the Java Node Manager

The Java node manager can be executed using a script. When executing **startNodeManager** under the '`JEUS_HOME/bin`' folder, the node manager becomes ready to receive commands from Master or from the console tool to control the servers. Available options can be checked using the '`startNodeManager -h`' command.

```

JEUS_HOME/bin$ startNodeManager
*****
- JEUS Home      : /home/jeus/jeus9
- Added Java Option :
- Java Vendor    : Sun
*****
...
[2016.07.28 13:55:32][2] [nodemanager-1] [NodeManager-0101] The node manager is starting.
[2016.07.28 13:55:32][2] [nodemanager-1] [NodeManager-0102] Initializing the node manager
configuration.
[2016.07.28 13:55:32][2] [nodemanager-1] [NodeManager-0108] Beginning to listen:
localhost/127.0.0.1:7730.
[2016.07.28 13:55:32][2] [nodemanager-10] [NodeManager-0109] Processing the request.....

```

If the node manager is restarted after being terminated while monitoring the servers, the node manager performs preparation tasks to monitor the servers again and the following logs are generated.

```

JEUS_HOME/bin$ startNodeManager
*****
- JEUS Home      : /home/jeus/jeus9
- Added Java Option :
- Java Vendor    : Sun
*****
...
[2016.07.28 13:59:16][2] [nodemanager-1] [NodeManager-0101] The node manager is starting.
[2016.07.28 13:59:16][2] [nodemanager-1] [NodeManager-0102] Initializing the node manager
configuration.
[2016.07.28 13:59:16][2] [nodemanager-1] [NodeManager-0108] Beginning to
listen:localhost/127.0.0.1:7730.
[2016.07.28 13:59:16][2] [nodemanager-1] [NodeManager-0115] Domain=[domain1], Server=[adminServer]
[2016.07.28 13:59:16][2] [nodemanager-10] [NodeManager-0109] Processing the request.....
[2016.07.28 13:59:16][2] [nodemanager-1] [NodeManager-0115] Domain=[domain1], Server=[server1]
[2016.07.28 13:59:16][2] [nodemanager-11] [NodeManager-0137] Beginning to monitor the
server[adminServer] in the domain[domain1].
[2016.07.28 13:59:16][2] [nodemanager-12] [NodeManager-0137] Beginning to monitor the server[server1]
in the domain[domain1].
[2016.07.28 13:59:16][2] [nodemanager-11] [NodeManager-0145] The process is alive(Server=adminServer,
Process ID=4856).
[2016.07.28 13:59:17][2] [nodemanager-12] [NodeManager-0145] The process is alive(Server=server1,
Process ID=5376).

```



When the node manager is started, Master may also be started. When the domain is configured and the node manager is about to run for the first time, the Master startup option enables Master to start on the node.

2.5.2. Terminating the Java Node Manager

The Java node manager can be terminated using a script or the console tool.

- **Using a script**

The node manager can be terminated by using a script as in starting a Java node manager. When the **stopNodeManager** command is executed, the Java node manager is terminated.

```
JEUS_HOME/bin$ stopNodeManager -host localhost -port 7730
*****
- Usage : stopNodeManager -host host -port port
*****
Succeed to stop the node manager.
```

When terminating the node manager, the path of the jeusnm.xml file may be specified using the -properties option. The host and port information is read from the node manager's configuration file to terminate the node manager.

```
JEUS_HOME/bin$ stopNodeManager -properties JEUS_HOME/nodemanager/jeusnm.xml
*****
- Usage : stopNodeManager -host host -port port
*****
Succeed to stop the node manager.
```

- **Using the console tool**

When the **nm-stop** command is executed in the console tool, the node manager is terminated. For more information about the nm-stop command, refer to "stop-nodemanager" in *JEUS Reference Guide*.

```
JEUS_HOME/bin$ jeusadmin
JEUS 9 Administration Tool
To view help, use the 'help' command.
offline>nm-stop -host localhost -port 7730
Succeed to stop the node manager.
```

2.6. Controlling Servers Using the Java Node Manager

The Java node manager controls the servers. This section describes how to control the servers using the console tool.

2.6.1. Using the Console Tool

The console tool can be used to access the node manager to start/terminate a server and to check the server state.

The console tool and node manager must be connected in order to control a server using the console tool. After completing the necessary tasks, terminate the connection.

- **Connecting to the node manager**

To connect to the node manager, use the **nm-connect** command.

```
JEUS_HOME/bin$ jeusadmin
JEUS 9 Administration Tool
To view help, use the 'help' command.
offline>nm-connect -host localhost -port 7730 -domain domain1
The connection to the node manager domain1 has been established.
[NodeManager]domain1>
```

- **Starting a server using the node manager**

To start a server, use the **nm-start-server** command.

```
[NodeManager]domain1>nm-start-server -server adminServer -u jeus -p jeus
succeed to start server[adminServer].
    RUNNING
[NodeManager]domain1>nm-start-server -server server1 -u jeus -p jeus
-masterurl 192.168.0.4:9736
succeed to start server[server1].
    RUNNING
```

The **nm-start-server** command also connects the server to the node manager.

```
JEUS_HOME/bin$ jeusadmin
offline>nm-start-server -host localhost -port 7730 -domain domain1 -server adminServer -u jeus -p
jeus
succeed to start server[adminServer].
    RUNNING
```

- **Using the script to start Master controlled by the node manager**

Master can be connected to and be controlled by the node manager even if the server was not started by the node manager.

The **startMasterServerNM** script accesses the node manager and executes the start command on Master which enables the node manager to start Master. A single script is provided for connecting to the node manager through the console tool and starting the server.



When Master is started using the startMasterServer script, it can be managed by the node manager. Master connects to the node manager and registers its information. If Master information is registered to the node manager, the node manager can check the Master state to detect any errors and restart it.

```
JEUS_HOME/bin$ startMasterServerNM -host 192.168.0.26 -port 7730 -domain domain1 -server
adminServer -u jeus -p jeus
```



```
*****
- Usage : startMasterServerNM -host host -port port -domain domain -server
server1 -u username -p password
*****

succeed to start server[adminServer].
RUNNING
```

- **Checking the server state using the node manager**

Use the **nm-state-server** command to check the server state.

```
[NodeManager]domain1>nm-state-server -server adminServer -u jeus -p jeus
server[adminServer] : RUNNING
[NodeManager]domain1>nm-state-server -server server1 -u jeus -p jeus
server[server1] : RUNNING
```

- **Terminating the server using the node manager**

Use the **nm-stop-server** command to terminate the server.

```
[NodeManager]domain1>nm-stop-server -server server1 -u jeus -p jeus
succeed to stop server[server1].
[NodeManager]domain1>nm-stop-server -server adminServer -u jeus -p jeus
succeed to stop server[adminServer].
```

- **Disconnecting from the node manager**

Use the **nm-disconnect** command to disconnect the node manager.

```
[NodeManager]domain1>nm-disconnect
disconnect to node manager.
offline>
```

- **Starting a server in Master**

When starting an MS using the **start-server** command, the server is started by the node manager.

```
[MASTER]domain1.adminServer>start-server server1
The server [server1] was successfully started.
```



For more information about the commands to connect and disconnect from the Java node manager, and the server control commands through the Java node manager in the console tool, refer to "Node Management Commands" in *JEUS Reference Guide*.

2.7. Log Files

The node manager logs are stored in the 'JeusNodeManager.log' file under the 'JEUS_HOME/nodemanager/logs' folder. The node manager records the log messages that are generated during server startup and monitoring. It also records the booting log messages, which are generated during server startup.

To set the log level of the node manager, configure the following setting in the node manager startup script.

```
-Djeus.nodemanager.log.level=FINEST
```



For more information about logs and log levels, refer to JEUS Server "Logging" in *JEUS Server Guide*.

3. SSH Node Manager

This chapter describes how to configure the SSH node manager.

3.1. Overview

The SSH node manager can run on UNIX or UNIX-like operating systems by executing SSH (Secure Shell) commands on the node. It allows easy installation of the JEUS server on the node, and Master can use the SSH node manager to configure a local node. However, Windows does not support the SSH node manager.



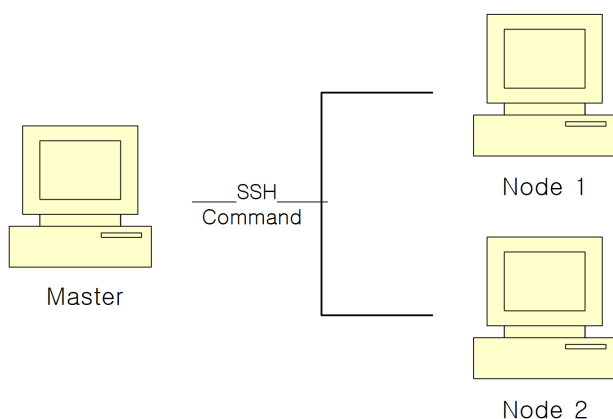
This chapter uses remote nodes and remote servers to distinguish between remote and local (Master machine) configurations.

3.2. Configuring the SSH Environment

To use the SSH node manager, SSH must be configured first.

3.2.1. Configuring SSH

To use the SSH node manager functions on UNIX or UNIX-like operating systems, the Master machine must be able to connect to a node via SSH. In other words, the Master machine acts as the SSH client, and the remote node acts as the SSH server. This section does not describe how to start the SSH server, but it only describes the settings required for connecting to a node via SSH.



Connecting to a Remote Node via SSH

A private key is required to connect to a remote node via SSH. To generate it, the Master machine uses `ssh-keygen` to create a public/private key pair. Because JEUS does not support passphrases, do not set a passphrase when generating the keys.

The created public and private key pair is stored as the `"id_rsa"` and `"id_rsa.pub"` files under the

'USER_HOME/.ssh' directory. If the contents of the public key's "id_rsa.pub" file are added to the 'USER_HOME/.ssh/authorized_keys' file on the remote node, then the private key can be used to connect to the remote machine.

To check if it is possible to connect to a remote node via SSH using the private key, run the following SSH command.

```
ssh <ssh-user-name>@<remote-host-address>
```

The following is an example of the SSH command.

```
$ ssh jeus@192.168.23.129
```

To start a remote server using the SSH command, the environment variable, JAVA_HOME, must be configured on the node. Since the environment variables configured in the 'USER_HOME/.bashrc' file on the remote machine are used when executing the SSH command, configure the .bashrc file as in the following example. By default, the SSH commands are executed in a non-interactive mode in JEUS. Since the configured environment variables might not work properly in the non-interactive mode, they must be configured at the root level as in the following example.

```
# ~/.bashrc

export JAVA_HOME='/home/java/jdk17'
export PATH='/home/java/jdk17/bin':$PATH

# If not running interactively, don't do anything
[ -z "$PS1" ] && returnssh
...
```

In the Korn shell, configure the previous settings in the USER_HOME/.kshrc file. Then, create the file, 'USER_HOME/.ssh/environment', and add 'ENV=~/.kshrc' to the file.

```
$ cat ~/.ssh/environment
ENV=~/.kshrc
```

For the previous configuration to work properly, set PermitUserEnvironment setting in the '/etc/ssh/sshd_config' file to 'yes', and then restart SSHD.

Configure the JAVA_HOME environment variable on the remote node and check if the configuration has been applied successfully by executing the following command on the Master machine.

```
$ ssh jeus@192.168.23.129 java -version
java version "17.0.6" 2023-01-17 LTS
Java(TM) SE Runtime Environment (build 17.0.6+9-LTS-190)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.6+9-LTS-190, mixed mode, sharing)
```

If the command executes successfully, then SSH is ready for use.

3.3. Configuring and Deleting Nodes

An SSH node can be configured and deleted by using WebAdmin and the console tool.

3.3.1. Configuring an SSH Node

After the SSH configuration is complete, the node configuration must be applied to JEUS.

A node requires the following basic information. Configure the required settings according to the node type.

- Node name that is used as the ID
- Address of the remote machine
- JEUS installation directory of the remote machine

The SSH node manager can be supported in UNIX environments. The port number, user name, and the path to the private key file must be configured to access SSH.

This section describes how to configure the SSH node manager in JEUS using WebAdmin and the console tool.

3.3.1.1. Using WebAdmin

The following shows how to configure an SSH node using WebAdmin.

1. All node-related settings can be configured from the **[Node Configuration]** menu. Select **[Node Configuration]** from the main page of WebAdmin.
2. The list of existing nodes is displayed on the **Nodes** page. The list contains the nodes that were automatically created when the JEUS instance was installed even if the user did not add any nodes. The list shows the node name, type, whether the node can be controlled, and JEUS version. To add a node, click **[Add]**.
3. On the **Nodes** page, enter the node name, host address, and JEUS installation path. Select **"Ssh"** for the node type, configure the required items, and then click **[OK]**.
4. If the node is added successfully, a success message is displayed on the upper part of the **Nodes** page and the node information is displayed.

If JEUS is successfully installed on a remote node, the installed JEUS version can be checked in the **"JEUS version"** column. If JEUS is not installed, the JEUS version will not be shown. If the node can be controlled, JEUS can be installed by clicking **[install]**. The node must be mapped to the server in order to start the JEUS server on the node.

5. To run the server on the node where JEUS is installed, map the defined node to the server. Select

the node name in the "**Node Name**" field to map to the server.

6. The servers that are mapped to the node where JEUS is installed can be checked from the **[Monitoring] > [Servers]** menu on the left of the screen.

Select a server from the server list and click **[start]**. On the page, configure each setting and click **[OK]** to start the server.



The server can be started from the **[Servers]** menu or the **[Monitoring] > [Servers]** menu. For more information about the menus, refer to [Java Node Manager](#).

7. If the server starts successfully on the remote node, a success message appears on the upper part of the screen, and the server's status is displayed as "RUNNING" in the "**Status**" column.

3.3.1.2. Using the Console Tool

The following shows how to configure nodes using the console tool.

1. Use the **add-ssh-node** command to add an SSH node to JEUS.

```
[MASTER]domain1.adminServer>add-ssh-node node2 -host 192.168.23.129
-dir /home/jeus/jeus9 -user jeus -privatekey /home/jeus/.ssh/id_rsa
The node [node2] was successfully added.
```

2. The **check-ssh-node** command can be used to check if the added SSH node is running normally. The command executes a Java command via SSH on the node. If the Java command execute successfully as in the following, it means that the added node is running successfully.

```
[MASTER]domain1.adminServer>check-ssh-node node2
The Domain Administration Server can execute the "java" process via SSH.
```

3. The node configuration is stored in the file, nodes.xml, and it can be checked by using the **show-node** command.

```
[MASTER]domain1.adminServer>show-node node2
=====
+-----+-----+
| Property | Value |
+-----+-----+
| Node name | node2 |
| Host | 192.168.23.129 |
| Mapped servers | |
| Node Type | SSH |
| Installed directory | /home/jeus/jeus9 |
| SSH user name | jeus |
| SSH private key | /home/jeus/.ssh/id_rsa |
+-----+-----+
```

SSH port	22
----------	----

- The node type, whether the node can be controlled, and the JEUS version can be checked by using the **list-nodes** command.

```
[MASTER]domain1.adminServer>list-nodes
=====
+-----+-----+-----+-----+
| Node name | Type | Under control | JEUS version |
+-----+-----+-----+-----+
| node1     | JAVA | N             | -            |
| node2     | SSH  | Y             | JEUS 9.1     |
+-----+-----+-----+-----+
=====
```

If the node is configured as in the previous example, the **install-jeus** command can be used in the console tool to install JEUS on the node.

```
[MASTER]domain1.adminServer>install-jeus node2
JEUS was successfully installed on the node [node2].
```



Since the **install-jeus** command copies the file from the Master machine, the node must have the same OS as the Master machine. If the OS is different, the node cannot operate properly because the native libraries cannot be used. Note that depending on the environment, the execution time may take longer.

- The configured node must be mapped to the server for it to operate on the node. Use the **modify-server** command, with the **-node** option to create the mapping.

The following example shows how to map a server to a node.

```
[MASTER]domain1.adminServer>modify-server server1 -node node2
Successfully performed the MODIFY operation for server (server1).
Check the results using "list-servers server1 or modify-server server1"
```

- The **show-node** command can be used to check the node to see if the server is mapped.

```
[MASTER]domain1.adminServer>show-node node2
=====
+-----+-----+
| Property | Value |
+-----+-----+
| Node name | node2 |
+-----+-----+
```

Host	192.168.23.129
Mapped servers	server1
Node Type	SSH
Installed directory	/home/jeus/jeus9
SSH user name	jeus
SSH private key	/home/jeus/.ssh/id_rsa
SSH port	22
+-----+-----+	
=====	

7. The server mapped to the node can be started or terminated by using the **start-server** command in the console tool, and all settings required to start the server are configured automatically.

```
[MASTER]domain1.adminServer>start-server server1
The server [server1] was successfully started.
```

If the server is terminated by the kill command on the node, Master detects that the server state has changed to "FAILED" and automatically restarts the server.

The **server-info** command can be used to check the changes of the server state.

```
[MASTER]domain1.adminServer>server-info
Information about Domain (domain1)
=====
```

Server	Status	Node Name	PID	Clu ster	Latest Start Time / Shutdown Time	Need to Restart	Listen Ports	Running Engines
admin Server (*)	RUNNING (00:11:17)	N/A	104 36	N/A	2016-08-23 (Tue) PM 12:44:00 KST	false	base-0.0. 0.0:9736 http-serv er-0.0.0.0 :8088 jms-0.0.0 .0:9741	jms, ejb, web
serve r1	FAILED(00:02:33)	nod e2	N/A	N/A	N/A	N/A	N/A	N/A
serve r2	SHUTDOWN	N/A	N/A	N/A	N/A	N/A	N/A	N/A

```
=====
```

```
[MASTER]domain1.adminServer>server-info
Information about Domain (domain1)
=====
```

Server	Status	Node Name	PID	Clu ster	Latest Start Time	Need to	Listen Ports	Running Engines
--------	--------	--------------	-----	-------------	----------------------	------------	-----------------	--------------------

					/ Shutdown Time	Restart			
admin Server (*)	RUNNING (00:14:4 8)	N/A	104 36	N/A	2016-08-23 (Tue) PM 12:44:00 KST	false	base-0.0. 0.0:9736	jms, ejb, web	
							http-serv er-0.0.0.0 :8088		
							jms-0.0.0 .0:9741		
serve r1	RUNNING (00:00:1 2)	nod e2	6516	N/A	2016-08-23 (Tue) PM 12:58:36 KST	false	BASE-0.0. 0.0:9836	jms, ejb, web	
serve r2	SHUTDOWN	N/A	N/A	N/A	N/A	N/A	N/A	N/A	



For more information about the commands that are used in this section, refer to "Server Management Commands", "Domain Configuration Commands", "Node Manager Commands" in *JEUS Reference Guide*.

3.3.2. Modifying an SSH Node

An SSH node can be modified using WebAdmin or the console tool.

1. A node can be modified using the **modify-ssh-node** or **modify-node** command.

```
[MASTER]domain1.adminServer>modify-ssh-node node2 -host 192.168.23.128
The node [node2] was modified successfully. Check the results using "show-node"
[MASTER]domain1.adminServer>modify-node node2 -host 192.168.23.127
The node [node2] was modified successfully. Check the results using "show-node"
```

2. To check the modified node, use the **show-node** command.

```
[MASTER]domain1.adminServer>show-node node2
=====
+-----+-----+
| Property | Value |
+-----+-----+
| Node Name | node2 |
| Host | 192.168.23.127 |
| Mapped Servers | server2 |
| Node Type | SSH |
| Installed directory | /home/jeus/jeus9 |
| SSH User Name | jeus |
| SSH Private Key | /home/jeus/.ssh/id_rsa |
| SSH Port | 22 |
+-----+-----+
```

=====

3.3.3. Deleting an SSH Node

An SSH node can be deleted by using WebAdmin or the console tool. Since deleting an SSH node is same as deleting a Java node, the explanation is omitted in this section. For more information, refer to [Deleting a Java Node](#).

4. Reliable Queue Server (RQS) Process Management

This chapter describes a node manager that manages Reliable Queue Server (RQS) processes.

4.1. Overview

A node manager manages non-server processes (currently RQS processes) as well as server processes. It starts and terminates RQS processes, checks whether they are alive, and restarts them if they were terminated abnormally. For this, the node manager operates regardless of server or node configuration and uses separately configured information.



RQS processes are managed only by the node manager located in the same machine (node). RQS processes on other machines or those not managed by a node manager cannot be controlled.

4.2. Operation of RQS Processes

This section describes how a node manager manages RQS processes.

4.2.1. Starting and Terminating RQS Processes

An RQS process is started by using a file path and options set in a configuration file. Since the process is independent of a server, it starts and terminates regardless of a server. A node manager reads RQS-related configuration during startup and monitors RQS processes according to the information. If the node manager detects that no RQS process operates, it will start the process.

1. A node manager reads RQS-related information from the jeusnm.xml file and saves it.
2. The node manager creates an RQS management object by using RQS-related information and then a monitoring thread.
3. The monitoring thread accesses RQS processes by using the RQS-related information and monitors the processes.
4. If an RQS process that is terminated abnormally or does not send a response is detected, it will be restarted.



RQS processes are restarted in the same order as above.

4.2.2. Monitoring RQS Processes

While RQS processes are monitored, a monitoring-related file is not created. Messages sent and received between a monitoring thread and RQS processes are used to access and monitor RQS processes and determine whether RQS processes are terminated normally. If a node manager cannot receive a response message from an RQS process after sending a message for checking the status of the RQS process several times, it will determine that the RQS process is terminated abnormally.

4.2.3. Restarting RQS Processes in Abnormal Status

If a node manager determines that an RQS process is terminated abnormally, the process will be restarted. Before restarting the process, the process is killed to make sure that the process is terminated. After restarting the process, a monitoring thread is created to monitor the restarted process.

4.3. Configuring the Environment

A node manager uses RQS-related configuration to manage RQS processes. RQS processes are started and managed through the configuration.

4.3.1. jeusnm.xml

Since RQS processes operate independently of servers and other systems, they rely entirely on the jeusnm.xml file. The processes can be added or deleted by modifying this file.

The rqsList element in the jeusnm.xml file contains information about RQS processes. The RQS item under processList defines the RQS process configurations that the node manager uses to manage the processes.

- processList

Contains information about RQS processes.

Each RQS item under processList contains information about a single process. To manage multiple RQS processes, add as many RQS items under processList as the number of RQS processes.

- RQS

Each of the following items contains a description of each process. (*: Required)

Item	Description
domainName *	<p>Name of an RQS process group.</p> <p>Multiple RQS processes are included in a group with this name.</p>
processName *	<p>Name of an RQS process.</p> <p>This name is used to start or stop the RQS process.</p>
path *	<p>Location of an executable file for an RQS process.</p> <p>The executable file in this location is executed to start the RQS process.</p>
port *	<p>Port number used to monitor an RQS process.</p> <p>Messages are sent and received between a monitoring thread and the QS process through this port.</p>
option	Argument required to execute RQS. You can specify options by referring to the RQS guide.
rqsdir	<p>A configuration file is required to start each RQS process. That is, multiple configuration files are required to start multiple RQS processes.</p> <p>Through the RQSDIR environment variable, a configuration file can be accessed.</p>
retryCount	Maximum number of retrying connection. (Default value: 5)
monitoringPeriod	Time interval of retrying connection. (Default value: 500)

5. Replicating Node Manager

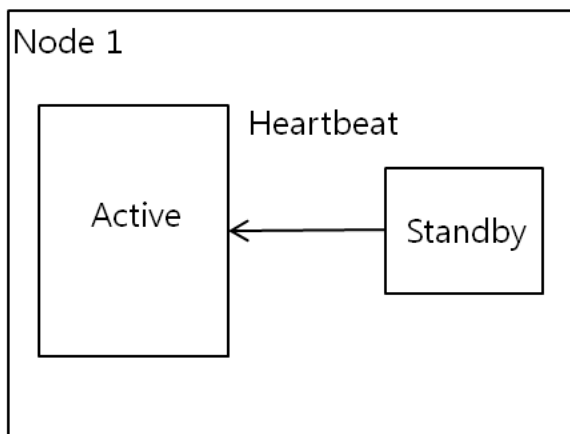
This chapter describes how to replicate a node manager to recover the node manager when it was terminated abnormally.

5.1. Overview

You can use a node manager to operate and terminate a server and restart a server when it fails. In case the node manager also fails, it can be replicated, and a replicated node manager can operate instead of a failed node manager.

5.2. Operating a Replicated Node Manager

A replicated node manager is a pair of an active and standby node managers. The active node manager manages and processes requests of Master or servers.



Operation of a Replicated Node Manager

A standby node manager monitors the status of the active node manager. If the standby node manager detects a failure in the active node manager, it determines that the active node manager is terminated abnormally and replaces the active manager. To monitor the standby node manager that has become active, another standby node manager is executed. The detailed procedure is as follows:

1. Active and standby node managers start at the same time.
2. The standby node manager monitors status of the active node manager.
3. The standby node manager may detect a failure in the active node manager.
4. The standby node manager starts another standby node manager.
5. The existing standby node manager becomes an active node manager and processes requests from a server.

The status of a node manager is checked through a port set in a configuration file. If the port is not set, the server regards that node manager replication is not used.



Whenever a standby node manager becomes an active node manager due to a failure in an active node manager, a standby node manager is executed additionally and monitors the previous standby node manager.

5.3. Using a Replicated Node Manager

To replicate a node manager, you need to configure the port used for sending and receiving messages between the active and standby nodes. This port is specified using the `standbyPort` item as described in [Configuration File](#). The active and standby node managers communicate through this port.

If you do not want to replicate a node manager, do not configure this port. In that case, a standby node manager will not be started.

5.3.1. Standby Node Manager

A standby node manager starts when an active node manager starts, and monitors the active node manager's status in standby mode.

```
[nodemanager-1] [NodeManager-0201] The standby node manager is starting.  
[nodemanager-1] [NodeManager-0102] Initializing the node manager configuration.
```

A standby node manager does not process requests of a server, and it saves the status information to a log file. This log file is created where a node manager log file is located with the name of a node manager name followed by the `'_standby'` string. This log file is used only by standby node managers.

When a standby node manager becomes an active node manager, this information is recorded in a log file for standby node managers. Then, the new active node manager uses the log file for active node managers, and a new standby node manager uses the log file for standby node managers. The log file for active node managers records information related to server requests and management, and the log file for standby node managers records the start of standby node managers and their communication with an active node manager.

A standby node manager gets and records the PID of the active node manager in a log when it first connects. The active process can be identified using this PID.

5.4. Terminating a Replicated Node Manager

A replicated node manager operates without downtime because a standby node manager replaces an active node manager that was terminated abnormally. Therefore, if you want to terminate a replicated node manager, you need to use the `stopNodeManager` script. This script makes an active

node manager send a termination message to a standby node manager. The standby node manager ends a connection, logs its own termination, and then terminates safely.



If a node manager is terminated forcibly due to an issue, its standby node manager may not be terminated. Therefore, the node manager must be terminated after terminating the standby node manager. It is recommended to use the `stopNodeManager` script because forcible termination may record log incorrectly.