Cluster Server Agent for JBoss Application Server Installation and Configuration Guide

Linux, Solaris

7.0

Veritas InfoScale™ Availability Agents

Last updated: 2020-10-08

Legal Notice

Veritas Technologies LLC
2625 Augustine Drive
Santa Clara, CA 95054

http://www.veritas.com

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:

https://www.veritas.com/support

You can manage your Veritas account information at the following URL:

https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:

https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:

infoscaledocs@veritas.com

You can also see documentation information or ask a question on the Veritas community site:

http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:

https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Introducing the agent for JBoss Application Server

This chapter includes the following topics:

- About the Cluster Server agent for JBoss Application Server
- Supported software
- Features of the agent
- How the agent supports intelligent resource monitoring
- JBoss Application Server agent functions

About the Cluster Server agent for JBoss Application Server

Cluster Server (VCS) agents monitor specific resources within an enterprise application. They determine the status of resources and start or stop them according to external events.

The Cluster Server agent for JBoss Application Server provides high availability for the standalone and domain-mode configurations of JBoss Application Server in a cluster.

About JBoss Application Server

JBoss Enterprise Application Platform is an implementation of the Java Enterprise Edition (Java EE) suite of services.

Because it is Java-based, JBoss Enterprise Application Platform is cross-platform and is easy to install and use on any operating system that supports Java.

A single JBoss Application Server installation can have multiple configurations within it; each such configuration is termed a Server Configuration. Individual Server Configurations can be run independently of each other.

Note: The JBoss Application Server 8.0 and later releases are now known as WildFly.

Supported software

For information on the software versions that the Cluster Server agent for JBoss Application Server supports, see the Veritas Services and Operations Readiness Tools (SORT) site: https://sort.veritas.com/agents.

Features of the agent

The following are the features of the Cluster Server agent for JBoss Application Server:

- Support for validation of attributes that are based on the agent functions
  The agent can validate attributes in each agent function before the actual data processing starts.
- Support for First Failure Data Capture (FFDC)
  In case of a fault, the agent generates a huge volume of the debug logs that enable troubleshooting of the fault.
- Support for Fast First Level Monitor (FFLM)
  The agent maintains PID files based on search patterns to expedite the monitoring process.
- Support for external user-supplied monitor utilities
  The agent enables user-specified monitor utilities to be plugged in, in addition to the built-in monitoring logic. This enables administrators to completely customize the monitoring of the application.
- Support for intelligent resource monitoring and poll-based monitoring
  The agent supports the Cluster Server Intelligent Monitoring Framework (IMF) feature. IMF allows the agent to register the resources to be monitored with the IMF notification module so as to receive immediate notification of resource state changes without having to periodically poll the resources.
  See "About Intelligent Monitoring Framework" on page 40.

How the agent supports intelligent resource monitoring

With Intelligent Monitoring Framework (IMF), VCS supports intelligent resource monitoring in addition to the poll-based monitoring. Poll-based monitoring polls the resources periodically whereas intelligent monitoring performs asynchronous monitoring.

When an IMF-enabled agent starts up, the agent initializes the Asynchronous Monitoring Framework (AMF) kernel driver. After the resource is in a steady state, the agent registers with the AMF kernel driver, the details of the resource that are required to monitor the resource. For example, the agent for JBoss Application Server registers the PIDs of the JBoss Application Server processes with the AMF kernel driver. The agent’s imf_getnotification function waits for any resource state changes. When the AMF kernel driver module notifies the imf_getnotification function about a resource state change, the agent framework runs the monitor agent function to ascertain the state of that resource. The agent notifies the state change to VCS, which then takes appropriate action.

For more information, see the *Cluster Server Administrator’s Guide*.

JBoss Application Server agent functions

The operations or functions that the Cluster Server agent for JBoss Application Server can perform are described as follows:

Online

The online function performs the following tasks:

- Verifies that the required attributes are set correctly.
- Verifies that the JBoss Application Server Configuration is not already online. If the Server Configuration is online, the online operation exits immediately.
- If any JBoss Application Server processes remain, the operation kills these processes using the user name associated with that specific resource.
- Attempts to start the JBoss Application Server Configuration with the following command:

Version	Mode	Command
9.x	Server	$ *JBossHome*/bin/jboss-cli.sh --connect --controller=*ManagementIP*:*ManagementPort* command=/host=*HostName*/server-config=*ServerName*:start
9.x	Domain	$ *JBossHome*/bin/domain.sh -Djboss.server.config.dir=*ConfigurationDir*\ -Djboss.server.base.dir=*ServerBaseDir* -Djboss.bind.address.management=\ManagementIP -c *ServerConfig*
6.0 and later	Standalone	$ *JBossHome*/bin/standalone.sh -Djboss.server.config.dir=*ConfigurationDir*\ -Djboss.server.base.dir=*ServerBaseDir* -Djboss.bind.address.management=\ManagementIP -c *ServerConfig*
5.x	Standalone	$ *JBossHome*/bin/run.sh -c *ServerConfig* -Djboss.server.base.dir=\ServerBaseDir -Djboss.server.base.url=file:*ServerBaseDir* -Djboss.\service.binding.set=*BindingSet* -Djava.rmi.server.hostname\\=*IpAddress* -Djboss.bind.address=*IpAddress* *StartArgs*

The command always gets executed in the context of the specified user in the resource configuration.

- Starts the JBoss Application Server in system.slice by using the `systemctl start` *`serviceFileName`* command if the UseSystemD attribute is set on the supported Linux platforms.
- Checks if the server has started up completely.
- Gives the control back to HAD.

Note: When a domain resource is configured to run in system.slice (UseSystemD = 1), all the server resources under that domain run in system.slice by default. This happens even if the server resources are configured to run in a different slice than their domain. However, Veritas recommends that you set the same value for the UseSystemD attribute for all the server resources that is set for their domain resource. The same is applicable to user.slice (UseSystemD = 0) as well.

Offline

The offline function performs the following tasks:

- Verifies that the required attributes are set correctly.
- Verifies that the JBoss Application Server Configuration is not offline.
- If the Server Configuration is already offline, the operation verifies if any processes belonging to this JBoss Application Server resource exist.
- Attempts to stop the JBoss Application Server Configuration with the following command:

Version	Mode	Command
9.x	Server	`$ JBossHome/bin/jboss-cli.sh --connect --controller=ManagementIP:ManagementPort command=/host=HostName/server-config=ServerName:stop`
9.x	Domain	`$ JBossHome/bin/jboss-cli.sh --connect --controller=ManagementIP\:ManagementPort command=/host=HostName:shutdown`
6.0 and later	Standalone	`$ JBossHome/bin/jboss-cli.sh --connect --controller=ManagementIP\:ManagementPort --command=:shutdown`
5.x	Standalone	`$ JBossHome/bin/shutdown.sh -S -u AdminUser -p AdminPassword -s \JNDIUrl StopArgs`

  If the command fails to shut down the application server configuration gracefully, the agent proceeds to kill the process pertaining to that particular server configuration.

  The command always gets executed in the context of the specified user in the resource configuration.

  Then the offline operation kills any existing processes that belong to this JBoss Application Server Configuration.

- Stops the JBoss Application Server in system.slice by using the `systemctl stop serviceFileName` command if the UseSystemD attribute is set on the supported Linux platforms.

Monitor

The monitor function monitors the states of the JBoss Application Server Configurations on all nodes within the cluster. The operation performs the following tasks:

- Conducts a first-level check on the JBoss Application Server Configuration to determine that the Server Configuration process is running. The agent identifies the process for the JBoss Application Server Configuration by applying the pattern matching on command lines of processes running in the system.

  The agent also supports Intelligent Monitoring Framework (IMF) in the first-level check. IMF enables intelligent resource monitoring. The agent for JBoss Application Server is IMF-aware and uses the asynchronous monitoring framework (AMF) kernel driver for resource state change notifications.

  See “How the agent supports intelligent resource monitoring” on page 9.

  You can use the MonitorFreq key of the IMF attribute to specify the frequency at which the agent invokes the monitor function.

  See “MonitorFreq” on page 42.

- Depending on the configuration, the monitor function can conduct a second-level check on the JBoss Application Server Configuration. The second-level check uses the following scripting utilities to check the status of the server:

JBoss Application Server version	Scripting utility used
5.x	`twiddle.sh`
6.x and later	`jboss-cli.sh`

Note: The attribute that is used to configure the second-level check and its frequency depends on the VCS version that is installed: For VCS 5.1 SP1 or later, use the LevelTwoMonitorFreq attribute. For VCS 5.1 or earlier, use the SecondLevelMonitor attribute.

- Depending upon the value of the MonitorProgram attribute, the monitor function can perform a customized check using a user-supplied monitoring utility.

Clean

The clean function performs the following tasks:

- Attempts to gracefully shut down the JBoss Application Server Configuration.

If the agent fails to shut down the JBoss Application Server Configuration gracefully, the agent proceeds to kill the process pertaining to that particular Server Configuration.

- Identifies the process for the JBoss Application Server Configuration and kills it.
- The clean operation kills any remaining process pertaining to this JBoss Application Server Configuration.

imf_init

This function initializes the JBoss Application Server agent to interface with the AMF kernel driver, which is the IMF notification module for the agent for JBoss Application Server. This function runs when the agent starts up.

imf_getnotification

This function gets notifications about resource state changes. This function runs after the agent initializes with the AMF kernel module. This function continuously waits for notification and takes action on the resource upon notification.

imf_register

This function registers or unregisters resource entities with the AMF kernel module. This function runs for each resource after the resource goes into a steady state—online or offline.

Chapter 2

Installing, upgrading, and removing the agent for JBoss Application Server

This chapter includes the following topics:

Before you install the Cluster Server agent for JBoss Application Server

You must install the Cluster Server agent for JBoss Application Server on all the systems that will host JBoss Application Server service groups.

Before you install the agent for JBoss Application Server, ensure that the following prerequisites are met.

- Install and configure Cluster Server.

For more information on installing and configuring Cluster Server, refer to the Cluster Server installation and configuration guides.

- Install the latest version of ACC Library.
  To install or update the ACC Library package, locate the library and related documentation in the Agent Pack tarball.
  See "About the ACC library" on page 15.

About the ACC library

The operations of a Cluster Server agent depend on a set of Perl modules known as the ACC library. The library must be installed on each system in the cluster that runs the agent. The ACC library contains common, reusable functions that perform tasks, such as process identification, logging, and system calls.

Instructions to install or remove the ACC library on a single system in the cluster are given in the following sections. The instructions assume that the ACCLib tar file has already been extracted.

Note: The LogDbg attribute should be used to enable debug logs for the ACCLib-based agents when the ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later.

Installing the ACC library

Install the ACC library on each system in the cluster that runs an agent that depends on the ACC library.

To install the ACC library

1. Log in as a superuser.
2. Download ACC Library.

   You can download either the complete Agent Pack tar file or the individual ACCLib tar file from the Veritas Services and Operations Readiness Tools (SORT) site (https://sort.veritas.com/agents).
3. If you downloaded the complete Agent Pack tar file, navigate to the directory containing the package for the platform running in your environment.

Linux	*cd1*/linux/generic/vcs/application/acc_library/*version*_library/rpms
Solaris	*cd1*/solaris/*dist_arch*/vcs/application/acc_library/*version*_library/pkgs

4. If you downloaded the individual ACCLib tar file, navigate to the pkgs directory (for AIX and Solaris), or rpms directory (for Linux).

5. Install the package. Enter **Yes**, if asked to confirm overwriting of files in the existing package.

Linux

```
# rpm -i \
VRTSacclib-VersionNumber-GA_GENERIC.noarch.rpm
```

Solaris

```
# pkgadd -d VRTSacclib.pkg
```

Note: The LogDbg attribute should be used to enable debug logs for the ACCLib-based agents when the ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later.

Installing the ACC library IPS package on Oracle Solaris 11 systems

Install the ACC library IPS package on an Oracle Solaris 11 system.

To install the ACC library IPS package on Oracle Solaris 11 systems

1. Copy the `VRTSacclib.p5p` package from the `pkgs` directory to the system in the `/tmp/install` directory.

2. Disable the publishers that are not reachable as package install may fail, if any, of the already added repositories are unreachable.

```
# pkg set-publisher --disable <publisher name>
```

3. Add a file-based repository in the system.

```
# pkg set-publisher -g /tmp/install/VRTSacclib.p5p Veritas
```

4. Install the package.

```
# pkg install --accept VRTSacclib
```

5. Remove the publisher from the system.

```
# pkg unset-publisher Veritas
```

6. Enable the publishers that were disabled earlier.

```
# pkg set-publisher --enable <publisher name>
```

Installing the ACC library package on Solaris brand non-global zones

With Oracle Solaris 11, you must install the ACC library package inside non-global zones. The native non-global zones are called Solaris brand zones.

To install the ACC library package on Solaris brand non-global zones

1. Ensure that the SMF services, `svc:/application/pkg/system-repository:default` and `svc:/application/pkg/zones-proxyd:default`, are online on the global zone.

   ```
   # svcs svc:/application/pkg/system-repository:default
   ```

   ```
   # svcs svc:/application/pkg/zones-proxyd:default
   ```

2. Log on to the non-global zone as a superuser.
3. Ensure that the SMF service `svc:/application/pkg/zones-proxy-client:default` is online inside the non-global zone:

   ```
   # svcs svc:/application/pkg/zones-proxy-client:default
   ```

4. Copy the `VRTSacclib.p5p` package from the `pkgs` directory to the non-global zone (for example, at the `/tmp/install` directory).
5. Disable the publishers that are not reachable, as package install may fail, if any of the already added repositories are unreachable.

   ```
   # pkg set-publisher --disable <publisher name>
   ```

6. Add a file-based repository in the non-global zone.

   ```
   # pkg set-publisher -g/tmp/install/VRTSacclib.p5p Veritas
   ```

7. Install the package.

   ```
   # pkg install --accept VRTSacclib
   ```

8. Remove the publisher on the non-global zone.

   ```
   # pkg unset-publisher Veritas
   ```

9. Clear the state of the SMF service, as setting the file-based repository causes the SMF service `svc:/application/pkg/system-repository:default` to go into the maintenance state.

   ```
   # svcadm clear svc:/application/pkg/system-repository:default
   ```

10. Enable the publishers that were disabled earlier.

    ```
    # pkg set-publisher --enable <publisher>
    ```

Note: Perform steps 2 through 10 on each non-global zone.

Installing the agent in a VCS environment

Install the agent for JBoss Application Server on each node in the cluster.

To install the agent in a VCS environment

1. Download the agent from the Veritas Services and Operations Readiness Tools (SORT) site: https://sort.veritas.com/agents.

   You can download either the complete Agent Pack tar file or an individual agent tar file.

2. Uncompress the file to a temporary location, say /tmp.
3. If you downloaded the complete Agent Pack tar file, navigate to the directory containing the package for the platform running in your environment.

   Linux

   ```
   cd1/linux/generic/vcs/application/jboss_agent/
   vcs_version/version_agent/rpms
   ```

   Solaris

   ```
   cd1/solaris/dist_arch/vcs/application/jboss_agent/
   vcs_version/version_agent/pkgs
   ```

   where, *dist_arch* is sol_sparc

   If you downloaded the individual agent tar file, navigate to the pkgs directory (for AIX and Solaris), or rpms directory (for Linux).

4. Log in as a superuser.
5. Install the package.

   Linux

   ```
   # rpm -ihv \
   VRTSjboss-AgentVersion-GA_GENERIC.noarch.rpm
   ```

   Solaris

   ```
   # pkgadd -d . VRTSjboss
   ```

After installing the agent package, you must import the agent type configuration file.

See "Importing the agent types files in a VCS environment" on page 27.

Installing the agent IPS package on Oracle Solaris 11 systems

To install the agent IPS package on an Oracle Solaris 11 system

1. Copy the `VRTSjboss.p5p` package from the `pkgs` directory to the system in the `/tmp/install` directory.
2. Disable the publishers that are not reachable as package install may fail, if any of the already added repositories are unreachable.

   ```
   # pkg set-publisher --disable <publisher name>
   ```

   where the publisher name is obtained using the `pkg publisher` command.
3. Add a file-based repository in the system.

   ```
   # pkg set-publisher -g /tmp/install/VRTSjboss.p5p Veritas
   ```
4. Install the package.

   ```
   # pkg install --accept VRTSjboss
   ```
5. Remove the publisher from the system.

   ```
   # pkg unset-publisher Veritas
   ```
6. Enable the publishers that were disabled earlier.

   ```
   # pkg set-publisher --enable <publisher name>
   ```

Installing agent packages on Solaris brand non-global zones

To install the agent package on Solaris brand non-global zones

1. Ensure that the SMF services, `svc:/application/pkg/system-repository:default` and `svc:/application/pkg/zones-proxyd:default`, are online on the global zone.

   ```
   # svcs svc:/application/pkg/system-repository:default
   ```

   ```
   # svcs svc:/application/pkg/zones-proxyd:default
   ```
2. Log on to the non-global zone as a superuser.
3. Ensure that the SMF service `svc:/application/pkg/zones-proxy-client:default` is online inside non-global zone:

   ```
   # svcs svc:/application/pkg/zones-proxy-client:default
   ```
4. Copy the `VRTSjboss.p5p` package from the `pkgs` directory to the non-global zone (for example, at the `/tmp/install` directory).

5. Disable the publishers that are not reachable, as package install may fail, if any of the already added repositories are unreachable.

   ```
   # pkg set-publisher --disable <publisher name>
   ```

6. Add a file-based repository in the non-global zone.

   ```
   # pkg set-publisher -g/tmp/install/VRTSjboss.p5p Veritas
   ```

7. Install the package.

   ```
   # pkg install --accept VRTSjboss
   ```

8. Remove the publisher on the non-global zone.

   ```
   # pkg unset-publisher Veritas
   ```

9. Clear the state of the SMF service, as setting the file-based repository causes the SMF service `svc:/application/pkg/system-repository:default` to go into the maintenance state.

   ```
   # svcadm clear svc:/application/pkg/system-repository:default
   ```

10. Enable the publishers that were disabled earlier.

    ```
    # pkg set-publisher --enable <publisher>
    ```

Note: Perform steps 2 through 10 on each non-global zone.

Installing the agent in a Solaris 10 brand zone

To install the JBoss Application Server agent in a Solaris 10 brand zone:

- Ensure that the ACC library package, VRTSacclib, is installed in the non-global zone.
  To install VRTSacclib in the non-global zone, run the following command from the global zone:

  ```
  # pkgadd -R /zones/zone1/root -d VRTSacclib.pkg
  ```

- To install the agent package in the non-global zone, run the following command from the global zone:

  ```
  # pkgadd -R zone-root/root -d . VRTSjboss
  ```

  For example: `# pkgadd -R /zones/zone1/root -d . VRTSjboss`

Note: You can ignore the following messages that might appear:

```
## Executing postinstall script.

ln: cannot create /opt/VRTSagents/ha/bin/JBoss/imf_getnotification:
File exists

ln: cannot create /opt/VRTSagents/ha/bin/JBoss/imf_register: File
exists

or ## Executing postinstall script.

ln: cannot create /opt/VRTSagents/ha/bin/JBoss/imf_getnotification:
No such file or directory

ln: cannot create /opt/VRTSagents/ha/bin/JBoss/imf_register: No such
file or directory
```

Uninstalling the agent in a VCS environment

You must uninstall the agent for JBoss Application Server from a cluster while the cluster is active.

To uninstall the agent in a VCS environment

1. Log in as a superuser.
2. Set the cluster configuration mode to read/write by running the following command from any node in the cluster:

   ```
   # haconf -makerw
   ```

3. Remove all JBoss Application Server resources from the cluster. Run the following command to verify that all resources have been removed:

   ```
   # hares -list Type=JBoss
   ```

4. Remove the agent type from the cluster configuration by running the following command from any node in the cluster:

   ```
   # hatype -delete JBoss
   ```

   Removing the agent’s type file from the cluster removes the include statement for the agent from the `main.cf` file, but the agent's type file is not removed from the cluster configuration directory. You can remove the agent’s type file later from the cluster configuration directory.

5 Save these changes. Then set the cluster configuration mode to read-only by running the following command from any node in the cluster:

```
# haconf -dump -makero
```

6 Use the platform's native software management program to remove the agent for JBoss Application Server from each node in the cluster.

Run the following command to uninstall the agent:

Linux

```
# rpm -e VRTSjboss
```

Solaris

```
# pkgrm VRTSjboss
```

Note: To uninstall the agent IPS package on a Solaris 11 system, run the following command:

```
# pkg uninstall VRTSjboss
```

Removing the ACC library

Perform the following steps to remove the ACC library.

To remove the ACC library

1 Ensure that all agents that use ACC library are removed.

2 Run the following command to remove the ACC library package:

Linux

```
# rpm -e VRTSacclib
```

Solaris

```
# pkgrm VRTSacclib
```

Note: To uninstall the ACCLib IPS package on a Solaris 11 system, run the following command:

```
# pkg uninstall VRTSacclib
```

Upgrading the agent for JBoss Application Server

Perform the following steps to upgrade the agent with minimal disruption.

To upgrade the agent for JBoss Application Server

1. Verify the version of the agent for JBoss Application Server.

Linux	`# rpm -qi VRTSjboss	grep Version`
Solaris	`# pkginfo -l VRTSjboss	grep VERSION`

   The output resembles:

   ```
   Version : 5.1.1.0
   ```

2. Save the VCS configuration.

   ```
   # haconf -dump -makero
   ```

3. Identify the JBoss Application Server resource and service group.

   ```
   # hatype -resources JBoss
   ```

   The output resembles:

   ```
   jboss
   ```

   ```
   # hares -display jboss | grep Group
   ```

   The output resembles:

   ```
   jboss      Group      global      JBoss_grp
   ```

4. Check the type-level attribute—Version.

   ```
   # hatype -display JBoss | grep Version
   ```

5. Freeze the JBoss service group.

   ```
   # hagrp -freeze JBoss_grp
   ```

6. Check if the JBoss Application Server agent is running.

   ```
   # haagent -display JBoss | grep Running
   ```

   The output resembles:

   ```
   JBoss        Running        Yes
   ```

7. If the agent is running, stop the agent.

   ```
   # haagent -stop JBoss -force -sys hostname
   ```

8 Verify the status of the agent.

```
# haagent -display JBoss | grep Running
```

The output resembles:

```
JBoss          Running          No
```

9 Uninstall the `VRTSjboss` agent package from all the nodes. Use the platform's native software management program to remove the agent package (`VRTSjboss`) for JBoss Application Server from each node in the cluster.

Run the following command to uninstall the agent:

Linux	`# rpm -e VRTSjboss`
Solaris	`# pkgrm VRTSjboss`

10 Install the latest agent.

See “Installing the agent in a VCS environment” on page 18.

On successful installation of the agent, the `UpdateTypes.pl` script runs automatically and adds the new attributes, if any, to the configuration.

11 Start the JBoss Application Server agent.

```
# haagent -start JBoss -sys hostname
```

The output resembles:

```
VCS NOTICE V-16-1-10001 Please look for messages in the log file
```

12 Verify the status of the agent.

```
# haagent -display JBoss | grep Running
```

The output resembles:

```
JBoss          Running          Yes
```

13 Unfreeze the JBoss service group.

```
# hagrp -unfreeze JBoss_grp
```

Virtualizing JBoss Application Server

To ensure that your JBoss Application Server can function properly on any node of the cluster, you need to virtualize all the parameters that could be dependent on a particular node.

Review the following basic notes for virtualization:

Host names	When installing and configuring the JBoss Application Server, ensure that you enter the virtual host name associated with the IP address used to configure the IP resource. This ensures that if the application needs to be migrated, you are not tied down by the physical IP address given to the JBoss Application Server.
Path names	Ensure that your application gets installed on a shared disk so that it is not constrained by anything that is local to the node. If this is not possible every time, make sure that the local data is available on each configured node.

Chapter 3

Configuring the agent for JBoss Application Server

This chapter includes the following topics:

About configuring the Cluster Server agent for JBoss Application Server

After installing the Cluster Server agent for JBoss Application Server, you must import the agent type configuration file. After importing this file, review the attributes table that describes the resource type and its attributes, and then create and configure JBoss Application Server resources.

To view the sample agent type definition and service groups configuration:

See "About sample configurations for the agents for JBoss Application Server" on page 60.

Importing the agent types files in a VCS environment

To use the agent for JBoss Application Server, you must import the agent types file into the cluster. You can import the agent types file using the VCS graphical user interface or using the command line interface.

To import the agent types file using the VCS Java GUI

1. Start the Cluster Manager (Java Console) and connect to the cluster on which the agent is installed.
2. Click **File > Import Types**.
3. In the **Import Types** dialog box, select the following file:

VCS 5.x and later	Linux	`/etc/VRTSagents/ha/conf/JBoss/JBossTypes.cf`
VCS 5.0	Solaris SPARC	`/etc/VRTSagents/ha/conf/JBoss/JBossTypes50.cf`
VCS 5.1 and later	Solaris SPARC	`/etc/VRTSagents/ha/conf/JBoss/JBossTypes51.cf`

4. Click **Import**.
5. Save the VCS configuration.

You can now create JBoss Application Server resources. For additional information about using the VCS GUI, refer to the *Cluster Server Administrator's Guide*.

To import the agent types file using the CLI

1. Log on to any one of the systems in the cluster as the superuser.
2. Run the following command:

```
# sh /etc/VRTSagents/ha/conf/JBoss/JBossTypes.cmd
```

3. To verify that the agent types file is successfully imported to the VCS engine, run the following command:

```
# hatype -display JBoss
```

If the file is successfully imported, you can proceed to create JBoss Application Server resources.

JBoss Application Server agent attributes

Refer to the required and optional attributes while configuring the agent for JBoss Application Server.

Table 3-1 lists the required attributes for the JBoss Application Server agent.

Table 3-1 Required attributes

Attribute	Description
ResLogLevel	Specifies the logging detail that the agent performs for the resource. The valid values are as follows: ■ ERROR: Only logs error messages. ■ WARN: Logs error messages and warning messages. ■ INFO: Logs error messages, warning messages, and informational messages. ■ TRACE: Logs error messages, warning messages, informational messages, and trace messages. TRACE is very verbose and should be used only during initial configuration or for troubleshooting and diagnostic operations. Type and dimension: string-scalar Default Value: INFO Example: TRACE **Note:** You must use the LogDbg attribute instead of the ResLogLevel attribute to enable debug logs for the ACCLib-based agents, when the ACCLib version is 6.2.0.0 or later and the VCS version is 6.2 or later. The agent captures the first failure data of the unexpected events and automatically logs debug messages in their respective agent log files.
Version	Specifies the version of the JBoss Application Server. The value of this attribute must be set identical to the version of the installed JBoss Application Server. Default Value: "" Example: 5.0, 6.0.1, 8.1.0, 9.0

Table 3-1 Required attributes *(continued)*

Attribute	Description
Component	Specifies the component or mode for which the resource is being configured. In this release, the agent supports the Standalone, Domain, and Server modes. This attribute is applicable to JBoss Application Server 6.0 and later. Default Value: "" Example: Standalone
ServerConfig	For JBoss Application Server 6.0 and later, this attribute specifies the filename of the configuration file that the agent must use. Ensure that you specify only the filename of the configuration file here; you may use the ConfigurationDir optional attribute to specify the location of the configuration file. Example: `standalone.xml`, `domain.xml` For JBoss Application Server 5.x, this attribute specifies the name of the Server Configuration that the agent must use. Example: production, web Default Value: ""
User	The UNIX user name used to start and stop the JBoss Application Server Configuration. If MonitorProgram is specified, the agent uses this user's credentials to run the defined program. You must synchronize the user name across the systems within the cluster. This user name must resolve to the same UID and have the same default shell on each system in the cluster. The agent operations use the getpwnam (3C) function system call to obtain UNIX user attributes. Hence you can define the user name locally or in a common repository such as NIS, NIS+, or LDAP. Type and dimension: string-scalar Default Value: "" Example: root

Table 3-1 Required attributes *(continued)*

Attribute	Description
AdminUser	The administrative user of the JBoss Application Server configuration. This user name is specified during the installation of JBoss Application Server. This attribute is applicable to JBoss Application Server 5.x. Default Value: "" Example: admin
AdminPassword	The password of the user of the JBoss Application Server. This password is specified during the installation of JBoss Application Server. The password is encrypted using the VCS encrypt utility, vcsencrypt (1m). This attribute is applicable to JBoss Application Server 5.x. **Note:** You need not encrypt the password if you are using the VCS GUI to enter the password. VCS GUI automatically encrypts the password. Default Value: "" Example : jxmXkvVvkVnvWvsVx

JBossHome	The absolute path of the JBoss home directory. The bin directory is present under this directory. This directory is required to access the startup scripts.
	Note: This is a directory name used in pattern matching for running process; do not keep extra trailing forward slashes (/) in the directory name.
	Default Value: none
	Examples
	`/opt/EnterprisePlatform-5.1.2/jboss-eap-5.1/jboss-as`
	`/opt/jboss/EAP-6.0.0.GA/jboss-eap-6.0`

Table 3-1 Required attributes *(continued)*

Attribute	Description
ManagementAddress	A combination of an IP address and port number that is used to specify the management bind address and the controller address. Specify this combination in the format IP:Port, where, IP is the management bind address and Port is the controller parameter that is used when executing the jboss-cli.sh utility. To specify an IPv6 address, use [], as per the standard IPv6 syntax. For example: "[::1]:9999", "[2001:db8:85a3:8d3:1319:8a2e:370:7348]:9999" Default Value: "" Example: 192.168.1.100:12345, myhostname:6789, myhostname.mydomain.com:7890

ServerBaseDir	The absolute path of the server's base directory.
	Note: This is a directory name used in pattern matching for running process; do not keep extra trailing forward slashes (/) in the directory name.
	For JBoss Application Server 6.0 and later in the standalone or domain mode, the value of this attribute is passed as the command line parameter '-Djboss.server.base.dir' to the startup script.
	Default Value: None
	If no value is specified for JBoss Application Server 6.0 and later, the agent uses the value `*JBossHome*/standalone` or `*JBossHome*/domain` as appropriate for the configuration.
	Examples
	`/opt/EnterprisePlatform-5.1.2/jboss-eap-5.1/jboss-as/server`
	`/opt/jboss/EAP-6.0.0.GA/jboss-eap-6.0/standalone`

BindingSet	The name of the set of bindings to use for the JBoss Application Server Configuration. Every Server Configuration must use a different binding set. This attribute is applicable to JBoss Application Server 5.x. Default Value: "" Example: ports-default, ports-01, ports-02, and so on.

Table 3-1 Required attributes *(continued)*

Attribute	Description
JNDIUrl	The JNDI Url field according to the BindingSet. As per the default installation, each binding set increases the port value by hundred. This attribute is applicable to JBoss Application Server 5.x. Default Value: "" For example: ■ ports-default corresponds to jnp://127.0.0.1:1099 ■ ports-01 corresponds to jnp://127.0.0.1:1199 ■ ports-02 corresponds to jnp://127.0.0.1:1299.
EnableCmdOutLogging	Determines whether to log the JBoss application command output in a file. When this attribute is set to 1, the command output is stored in a file. When this attribute is set to 0, the output is only displayed as appropriate. **Note:** The JBoss startup command runs in background, so a large amount of data may accumulate in this file over a period of time. Type and dimension: boolean-scalar Example: 1 Default: 0
UseSystemD	SystemD is a system and service manager for Linux operating systems. It helps manage applications across Linux distributions that support the SystemD feature. The UseSystemD attribute is applicable only on those systems that use the supported distributions of RHEL and SLES. ■ When this attribute is set to 1, the JBoss agent uses the JBoss service file for the online and the offline operations. The JBoss resource comes online or goes offline as a service in system.slice. ■ When this attribute is set to 0, a typical online function starts the resource or a typical offline function stops the resource in user.slice. Type and dimension: boolean-scalar Example: 1 Default: 0

Table 3-2 lists the optional attributes for the JBoss Application Server agent.

Table 3-2 Optional attributes

Attribute	Description
LogDbg	For ACCLib-based agents, you must use the LogDbg resource type attribute to enable the debug logs when the ACCLib version is 6.2.0.0 or later and the VCS version is 6.2 or later. Set the LogDbg attribute to DBG_5 to enable debug logs for ACCLIB based agent. By default, setting the LogDbg attribute to DBG_5 enables debug logs for all JBoss resources in the cluster. If debug logs must be enabled for a specific JBoss resource, override the LogDbg attribute. Type and dimension: string-keylist Default: {} (none) For more information on how to use the LogDbg attribute, refer to the *Cluster Server Administrator’s Guide*.
EnvFile	Specifies the full path to the file that the agent sources to set the environment before starting the server configuration. Typically, this attribute can be used to set environment variables like JAVA_HOME to make JBoss Application Server use the correct Java Runtime Environment. Veritas recommends storing the file on the shared disk. This ensures that the same file is available on each failover node. The shell environments supported are ksh, sh, and csh. Default Value: "" Example: /home/jboss/myenv.sh
MonitorProgram	Absolute path name of an external, user-supplied monitor executable. Default Value: "" Example: /home/jboss/mymonitorprogram.sh

Table 3-2 Optional attributes *(continued)*

Attribute	Description
ConfigurationDir	The absolute path of the configuration XML file. For the Standalone mode, the value of this attribute is passed as the command-line parameter '-Djboss.server.config.dir' to the startup script. Default: none If no value is specified for JBoss Application Server 6.0 and later, the agent uses the value `JBossHome/standalone/configuration` or `JBossHome/domain/configuration` as appropriate for the configuration. Example: `/opt/jboss/EAP-6.0.0.GA/jboss-eap-6.0/standalone/configuration`
SecondLevelMonitor	Used to enable second-level monitoring and specify how often it is run. Second-level monitoring is a deeper, more thorough state check of the server configuration. The numeric value specifies how often the second-level monitoring routines must run. For example, 0 means never run the second-level monitoring routines, 1 means run routines every monitor interval, 2 means run routines every second monitor interval. Care should be taken when setting this attribute to large numbers. For example, if the MonitorInterval is set to 60 seconds, and the SecondLevelMonitor is set to 100, then the second level check would only get performed every 100 minutes, which may not be as often as intended. To provide maximum flexibility, the value set is not checked for an upper limit. You can set the second level check to occur once a month, if that is desired. **Note:** The SecondLevelMonitor attribute is applicable to VCS version 5.1 and earlier. To enable second-level monitoring for VCS versions 5.1 SP1 and later, use the type-level attribute LevelTwoMonitorFreq. Default Value: 0 Example: 2

Table 3-2 Optional attributes *(continued)*

Attribute	Description
LevelTwoMonitorFreq	A type-level attribute that specifies the frequency at which the agent for this resource type must perform second-level or detailed monitoring. You can also override the value of this attribute at the resource level. The value indicates the number of monitor cycles after which the agent will monitor the JBoss Application Server Configuration in detail. For example, the value 5 indicates that the agent will monitor the JBoss Application Server Configuration in detail after every five online monitor intervals. The software-provided tool 'jboss-cli.sh' is used to query the running JBoss Application Server to perform an in-depth health-check. **Note:** This attribute is applicable to VCS version 5.1 SP1 or later. Type and dimension: integer-scalar Default: 0
StartArgs	Additional startup arguments to pass to the JBoss Application Server startup script. The most common requirement could be the use of '-b 0.0.0.0' to enable remote administration. This attribute is applicable to JBoss Application Server 5.x.
StopArgs	Additional shutdown arguments to pass to the JBoss Application Server shutdown script. This attribute is applicable to JBoss Application Server 5.x.

Table 3-2 Optional attributes *(continued)*

Attribute	Description
IMF	This resource type-level attribute determines whether the JBoss Application Server agent must perform intelligent resource monitoring. You can override the value of this attribute at the resource level. This attribute includes the following keys: ■ Mode: Define this attribute to enable or disable intelligent resource monitoring. Valid values are as follows: ■ 0—Does not perform intelligent resource monitoring ■ 1—Performs intelligent resource monitoring for offline resources and performs poll-based monitoring for online resources ■ 2—Performs intelligent resource monitoring for online resources and performs poll-based monitoring for offline resources ■ 3—Performs intelligent resource monitoring for both online and for offline resources. Type and dimension: integer-association Default values: 0 for VCS 5.1 SP1, 3 for VCS 6.0 and later. ■ MonitorFreq: This key value specifies the frequency at which the agent invokes the monitor agent function. The value of this key is an integer. Default: 1 You can set this key to a non-zero value for cases where the agent requires to perform both poll-based and intelligent resource monitoring. If the value is 0, the agent does not perform poll-based process check monitoring. After the resource registers with the AMF kernel driver, the agent calls the monitor agent function as follows: ■ After every (MonitorFreq x MonitorInterval) number of seconds for online resources ■ After every (MonitorFreq x OfflineMonitorInterval) number of seconds for offline resources ■ RegisterRetryLimit: If you enable intelligent resource monitoring, the agent invokes the imf_register agent function to register the resource with the AMF kernel driver. The value of the RegisterRetryLimit key determines the number of times the agent must retry registration for a resource. If the agent cannot register the resource within the limit that is specified, intelligent monitoring is disabled until the resource state changes or the value of the Mode key changes. Default: 3.

Table 3-2 Optional attributes *(continued)*

Attribute	Description
IMFRegList	An ordered list of attributes whose values are registered with the IMF notification module. You can override the value of this attribute at the resource level Type and dimension: string-vector Default: None
ServerName	JBoss Application Server versions 9.x require this attribute in the Server mode configuration. Example: server-one Type and dimension: string-scalar
HostName	JBoss Application Server versions 9.x require this attribute in the Domain and Server mode configurations. Example: master Type and dimension: string-scalar
ServiceName	The name of the unit service file that is used to start and stop the JBoss application. If this attribute is not specified, the agent uses the default service name for the respective platform. Type and dimension: string-scalar Example: wildfly Default: No default value

Executing second-level monitoring

Depending upon the JBoss Application Server version that is in use, the agent executes second-level monitoring in the following manner:

- For JBoss Application Server 5.x
  The monitor function performs tests as part of the second-level state check. JBoss Application Server bundles a helper script — twiddle.sh. This script is a JMX client that can be used to determine the state of the JBoss Application Server Configuration.
  The appropriate JBoss Application Server Configuration should have been started using the correct value for the *BindingSet* attribute. The *BindingSet* attribute determines which ports the JBoss Application Server Configuration will use. For example:

ports-default corresponds to 1099

ports-01 corresponds to 1199, and so on.

- For JBoss Application Server 6.0 and later

  The monitor function performs tests as part of the second-level state check. JBoss Application Server bundles a helper script — jboss-cli.sh. This script uses the ManagementAddress attribute to determine the state of the running JBoss Application Server.

Executing a customized monitoring program

You can configure the monitor function to execute a custom monitor program to perform a user-defined JBoss Application Server state check. Based on the UNIX user defined in the User attribute, this MonitorProgram runs in this user-defined shell.

- The MonitorProgram attribute value is set to a valid executable program.
- The first-level process check indicates that the JBoss Application Server Configuration is online.
- The LevelTwoMonitorFreq attribute is either set to 0 (false), or LevelTwoMonitorFreq is set to 1 (true) and the second-level check indicates that the JBoss Application Server Configuration is online.

  This feature allows cluster administrators to define custom programs that can further determine the state of the JBoss Application Server Configuration. For example, if the administrator wants to test the status of a J2EE component running inside the JBoss Application Server Configuration, the administrator can execute a custom program to determine that the underlying application is working properly.

The monitor function interprets the utility exit code as follows:

110 or 0	JBoss Application Server Configuration is online
100 or 1	JBoss Application Server Configuration is offline
99	JBoss Application Server Configuration is unknown
Any other value	JBoss Application Server Configuration is unknown

To ensure that the custom monitor program is always available to the agent application, Veritas recommends storing the file on the shared storage device so that it is available on all the nodes of the cluster.

Uniquely identifying JBoss Application Server Configurations

You can virtualize a JBoss Application Server Configuration using a cluster. Using shared disk and virtual IP addresses, you can manage a large set of JBoss Application Server Configurations in a single cluster.

JBoss Application Server Configurations can run on separate cluster nodes or concurrently on a single node. In the latter case, it is important that the agent for JBoss Application Server can uniquely identify a JBoss Application Server Configuration on a node that is hosting more than one simultaneous Server Configuration.

Differentiating JBoss Application Server Configurations is especially important when the agent for JBoss Application Server must kill the processes of a non-responsive or failed Server Configuration. Failure to define unique names for each Server Configuration can result in a clean operation that erroneously kills processes for more than one Server Configuration.

To uniquely identify a Server Configuration, the combination of JBossHome, ServerConfig, and ServerBaseDir must be unique for each JBoss Application Server Configuration.

Chapter 4

Enabling the agent for JBoss Application Server to support IMF

This chapter includes the following topics:

About Intelligent Monitoring Framework

With the IMF feature, VCS supports intelligent resource monitoring in addition to the poll-based monitoring. Poll-based monitoring polls the resources periodically whereas intelligent monitoring performs asynchronous monitoring. You can enable or disable the intelligent resource monitoring functionality of the JBoss Application Server agent.

VCS process and mount-based agents use the AMF kernel driver that provides asynchronous event notifications to the agents that are enabled for IMF.

You can enable the JBoss Application Server agent for IMF, provided the following software versions are installed:

- Cluster Server (VCS) 5.1 SP1 or later

- Cluster Server agent for JBoss Application Server version 5.1.0.0 or later

Refer to the *Cluster Server Administrator's Guide* for more information about IMF notification module functions and administering the AMF kernel driver.

Benefits of IMF

IMF offers the following benefits:

- Performance
  Enhances performance by reducing the monitoring of each resource at a default of 60 seconds for online resources, and 300 seconds for offline resources. IMF enables the agent to monitor a large number of resources with a minimal effect on performance.
- Faster detection
  Asynchronous notifications would detect a change in the resource state as soon as it happens. Immediate notification enables the agent to take action at the time of the event.

Agent functions for the IMF functionality

If the JBoss Application Server agent is enabled for IMF support, the agent supports the following functions, in addition to the functions mentioned in the JBoss Application Server agent functions topic.

imf_init

This function initializes the JBoss Application Server agent to interface with the AMF kernel driver, which is the IMF notification module for the agent for JBoss Application Server. This function runs when the agent starts up.

imf_getnotification

This function gets notifications about resource state changes. This function runs after the agent initializes with the AMF kernel module. This function continuously waits for notification and takes action on the resource upon notification.

imf_register

This function registers or unregisters resource entities with the AMF kernel module. This function runs for each resource after the resource goes into a steady state—online or offline.

Attributes that enable IMF

If the agent for JBoss Application Server is enabled for IMF support, the agent uses type-level attributes in addition to the agent-specific attributes.

IMF

This resource type-level attribute determines whether the JBoss Application Server agent must perform intelligent resource monitoring. You can also override the value of this attribute at the resource level.

This attribute includes the following keys:

Mode

Define this attribute to enable or disable intelligent resource monitoring. Valid values are as follows:

- 0—Does not perform intelligent resource monitoring
- 1—Performs intelligent resource monitoring for offline resources and performs poll-based monitoring for online resources
- 2—Performs intelligent resource monitoring for online resources and performs poll-based monitoring for offline resources
- 3—Performs intelligent resource monitoring for both online and for offline resources.

Note: The agent for JBoss Application Server supports intelligent resource monitoring for online resources only. Hence, Mode should be set to either 0 or 2.

Type and dimension: integer-association

Default: 0 for VCS 5.1 SP1, 3 for VCS 6.0 and later.

MonitorFreq

This key value specifies the frequency at which the agent invokes the monitor agent function. The value of this key is an integer.

Default: 1

You can set this key to a non-zero value for cases where the agent requires to perform both poll-based and intelligent resource monitoring.

If the value is 0, the agent does not perform poll-based process check monitoring.

After the resource registers with the AMF kernel driver, the agent calls the monitor agent function as follows:

- After every (MonitorFreq x MonitorInterval) number of seconds for online resources
- After every (MonitorFreq x OfflineMonitorInterval) number of seconds for offline resources

RegisterRetryLimit

If you enable intelligent resource monitoring, the agent invokes the imf_register agent function to register the resource with the AMF kernel driver.

The value of the RegisterRetryLimit key determines the number of times the agent must retry registration for a resource. If the agent cannot register the resource within the limit that is specified, then intelligent monitoring is disabled until the resource state changes or the value of the Mode key changes.

Default: 3.

IMFRegList

An ordered list of attributes whose values are registered with the IMF notification module.

Type and dimension: string-vector

Default: No default value

Note: The attribute values can be overriden at the resource level.

Before you enable the agent to support IMF

Before you enable the JBoss Application Server agent to support IMF, ensure that the AMF kernel module is loaded and AMF is configured. For details, refer to the ‘Administering the AMF kernel driver' section of the *Cluster Server Administrator's Guide*. For details about the commands you can configure AMF using the `amfconfig -h` command.

Enabling the agent to support IMF

In order to enable the JBoss Application Server agent to support IMF, you must make the following configuration changes to the attributes of the agent:

- AgentFile: Set the AgentFile attribute to **Script51Agent**
- IMF Mode: Set the IMF Mode attribute to **2**
- IMFRegList: Update the IMFRegList attribute

The following sections provide more information about the commands you can use to make these configuration changes, depending on whether VCS is in a running state or not.

Note: If you have upgraded VCS from an earlier version to version 5.1 SP1 or later, and you already have JBoss Application Server agent 5.1.00 or later installed, ensure that you run the following commands to create appropriate symbolic links:

```
# cd /opt/VRTSagents/ha/bin/JBoss
```

```
# ln -s /opt/VRTSamf/imf/imf_getnotification imf_getnotification
```

```
# ln -s /opt/VRTSagents/ha/bin/JBoss/monitor imf_register
```

If VCS is in a running state

To enable the JBoss Application Server resource for IMF when VCS is in a running state:

1. Make the VCS configuration writable.

   ```
   # haconf -makerw
   ```

2. Run the following command to update the AgentFile attribute.

   ```
   # hatype -modify JBoss AgentFile\
   /opt/VRTSvcs/bin/Script51Agent
   ```

3. For VCS version 6.0 or later, run the following commands to add the IMF attributes:

   ```
   # haattr -add -static  JBoss IMF -integer -assoc Mode 0 \
   MonitorFreq 1 RegisterRetryLimit 3
   ```

   ```
   # haattr -add -static JBoss IMFRegList -string -vector
   ```

   Note: Run these commands only once after you first enable IMF support for the agent.

4. Run the following command to update the IMF attribute.

   ```
   # hatype -modify JBoss IMF Mode num MonitorFreq num
   RegisterRetryLimit num
   ```

   For example, to enable intelligent monitoring of online resources, with the MonitorFreq key set to 5, and the RegisterRetryLimit key is set to 3, run the following command:

   ```
   # hatype -modify JBoss IMF Mode 2 MonitorFreq 5 \
   RegisterRetryLimit 3
   ```

   Note: The valid values for the Mode key of the IMF attribute are 0 (disabled) and 2 (online monitoring).

   ---

5. Run the following command to update the IMFRegList attribute:

   ```
   # hatype -modify JBoss IMFRegList JBossHome ServerBaseDir \
   ServerConfig User
   ```

6. Save the VCS configuration.

   ```
   # haconf -dump -makero
   ```

7. If the JBoss Application Server agent is running, restart the agent.

   For information on the commands you can use to restart the agent, see Restarting the agent.

Restarting the agent

To restart the agent:

1. Run the following command to stop the agent forcefully:

   ```
   # haagent -stop JBoss -force -sys <system>
   ```

   Note: Stopping the agent forcefully eliminates the need to take the resource offline.

   ---

2. Run the following command to start the agent:

   ```
   # haagent -start JBoss -sys <system>.
   ```

If VCS is not in a running state

To change the JBoss type definition file when VCS is not in a running state:

1. Update the AgentFile attribute.

   ```
   static str AgentFile = "/opt/VRTSvcs/bin/Script51Agent"
   ```

2. Update the IMF attribute.

   The valid values for the Mode key of the IMF attribute are 0 (disabled) and 2 (online monitoring).

   ```
   static int IMF{} = { Mode=num, MonitorFreq=num,
   RegisterRetryLimit=num }
   ```

   For example, to update the IMF attribute such that the Mode key is set to 2, the MonitorFreq key is set to 5, and the RegisterRetryLimit key is set to 3:

   ```
   static int IMF{} = { Mode=2, MonitorFreq=5, RegisterRetryLimit=3
   }
   ```

3. Update the IMFRegList attribute.

   ```
   static str IMFRegList[] = { JBossHome, ServerBaseDir,
   ServerConfig, User }
   ```

Disabling intelligent resource monitoring

To disable intelligent resource monitoring

1. Make the VCS configuration writable.

   ```
   # haconf -makerw
   ```

2. To disable intelligent resource monitoring for all the resources of a certain type, run the following command:

   ```
   # hatype -modify JBoss IMF -update Mode 0
   ```

3. To disable intelligent resource monitoring for a specific resource, run the following command:

   ```
   # hares -override resource_name IMF
   ```

   ```
   # hares -modify resource_name IMF -update Mode 0
   ```

4. Save the VCS configuration.

   ```
   # haconf -dump -makero
   ```

Configuring the service groups for JBoss Application Server using the CLI

This chapter includes the following topics:

About configuring service groups for JBoss Application Server

Configuring the JBoss Application Server service group involves creating the JBoss Application Server service group, its resources, and defining attribute values for the configured resources. You must have administrator privileges to create and configure a service group.

Before configuring the service groups for JBoss Application Server

Before you configure the JBoss Application Server service group, you must:

- Verify that the Cluster Server components are installed and configured on all nodes in the cluster where you will configure the service group.
  For more information on installing the components, refer to the *InfoScale Availability Installation Guide*.
- Verify that the Cluster Server agent for JBoss Application Server is installed on all nodes in the cluster.
  See "Installing the agent in a VCS environment" on page 18.

Configuring service groups for JBoss Application Server

Assuming that the target implementation has licensed the Storage Foundation and High Availability products, perform the following steps to configure service groups for JBoss Application Server.

To configure service groups:

1. Create UNIX user and group accounts.

   Veritas recommends creating a separate user under whose credentials the JBoss Application Server processes will run.

2. Create the supporting directory structure.

   A well-designed directory structure for your JBoss Application Server Configuration simplifies the cluster configuration and creates a storage environment that is intuitive and easier to manage.

   Assuming that all JBoss Application Server Configurations will be clustered and installed on shared disk, Veritas recommends a directory structure similar to the following:

Directory	Purpose
`/opt/EnterprisePlatform-5.1.2/`	The root directory for the JBoss Application Server installation This installs a few default directories.
`/opt/EnterprisePlatform-5.1.2/jboss-eap-5.1/jboss-as/`	JBoss Home Directory (JBossHome attribute)
`/opt/EnterprisePlatform-5.1.2/jboss-eap-5.1/jboss-as/server`	The base directory for the Server Configuration.

   The installation directory and the Server Configuration directory must be available on all the nodes of the cluster where JBoss Application Server is configured to run.

3. Create a disk group and volume.

   Consult the *Storage Foundation Administrator's Guide* for details on how to provision a disk group and volume resources.

4. Create the file system.

5. Create a virtual IP address.

   Provision a virtual IP address in the network namespace (DNS). Ensure that the virtual hostname that is assigned to the virtual IP can be resolved from all the nodes in the cluster. If the virtual hostname is being used from the local host map, make sure that the host map entry record is present on all the nodes in the cluster.

6. Create service group and resources on a cluster.

   Create a service group on a cluster and define resources for the NIC, IP, DiskGroup, and Mount resources. Bring these newly created resources online on one node in the cluster.

7. Install and configure JBoss Application Server.

   Install JBoss Application Server on the newly-created and mounted file system. After it is installed, change the file and group ownership to reflect the JBoss Application Server UNIX user and group accounts created earlier. Modify the JBoss Application Server Configuration to use the virtual IP address and port. Configuring the JBoss Application Server to bind is essential to ensure that it always listens on the same virtual IP address and port number regardless of the system in the cluster on which it is running.

8. Finalize and test the configuration as follows:

   - Create the JBoss Application Server resource.
   - Bring the newly-created resource online.
   - Test that the Server Configuration starts up, shuts down, and switches over as required, confirming overall availability requirements.
     To refer to a sample configuration service group: See "Sample service group configurations for JBoss Application Server" on page 60.

Creating service groups for JBoss Application Server under Solaris non-global zones

To configure zones on each cluster node

1. Set up the non-global zone configuration.

   ```
   hazonesetup servicegroup_name zoneres_name zone_name password
   systems
   ```

   For example:

   ```
   hazonesetup -g servicegroup_name -r zoneres_name -z zone_name
    -p password -s systems
   ```

2. Verify the non-global zone configuration.

   ```
   hazoneverify servicegroup_name
   ```

3. Whenever you make a change that affects the zone configuration, run the `hazonesetup` command to reconfigure the zones in VCS.

4. Make sure that the zone configuration files are consistent on all nodes at all times. The file is located at `/etc/zones/zone_name.xml`.
5. Make sure that the application is identical on all nodes. If you update the application configuration on one node, apply the same updates to all nodes.
6. Configure the service groups for JBoss Application Server.

Chapter 6

Troubleshooting the agent for JBoss Application Server

This chapter includes the following topics:

- Using the correct software and operating system versions
- Meeting prerequisites
- Verifying virtualization
- Configuring JBoss Application Server resources
- Starting the JBoss Application Server Configuration outside a cluster
- Reviewing error log files
- Troubleshooting the configuration for IMF
- Known issues

Using the correct software and operating system versions

Ensure that you use correct software and operating system versions.

For information on the software versions that the agent for JBoss Application Server supports, see the Veritas Services and Operations Readiness Tools (SORT) site: https://sort.veritas.com/agents.

Meeting prerequisites

Before installing the agent for JBoss Application Server, ensure that the following prerequisites are met.

For example, you must install the ACC library on VCS before installing the agent for JBoss Application Server.

Install all the JBoss Application Server components including JBossWS-CXF.

See “Before you install the Cluster Server agent for JBoss Application Server” on page 14.

Verifying virtualization

Verify that your application does not use anything that ties it down to a particular node of the cluster.

See “Virtualizing JBoss Application Server ” on page 25.

Configuring JBoss Application Server resources

Before using JBoss Application Server resources, ensure that you configure the resources properly. For a list of attributes used to configure all JBoss Application Server resources, refer to the agent attributes. See “JBoss Application Server agent attributes” on page 28.

Starting the JBoss Application Server Configuration outside a cluster

If you face problems while working with a resource, you must disable the resource within the cluster framework. A disabled resource is not under the control of the cluster framework, and so you can test the JBoss Application Server Configuration independent of the cluster framework.

Note: Use the same parameters that the resource attributes defined within the cluster framework while restarting the resource outside the framework, like the owner of the application, the environment file, and so on.

A sample procedure to start a JBoss Application Server Configuration outside the cluster framework, is illustrated as follows

To restart a JBoss Application Server Configuration outside the cluster framework

1. Log on, as the specified user of the JBoss Application Server Configuration, to the host on which the JBoss Application Server application is to run.
2. Use the values defined in the agent attributes to initiate the JBoss Application Server start program.

   For example, assume that the following values are assigned:

Attribute	Value
User	jboss
JBossHome	/opt/EnterprisePlatform-5.1.2/jboss-eap-5.1/jboss-as
ServerConfig	production
ServerBaseDir	/opt/EnterprisePlatform-5.1.2/jboss-eap-5.1/jboss-as/server

3. Go to the bin directory within the JBossHome attribute:

   ```
   # cd /opt/EnterprisePlatform-5.1.2/jboss-eap-5.1/jboss-as/bin
   ```

4. Start the JBoss Application Server Scripting Tool:

   ```
   # ./run.sh -c production
   ```

   The following message appears in a few minutes:

   ```
   18:27:44,535 INFO [JMXKernel] Legacy JMX core initialized ......
   ```

   If the JBoss Application Server Configuration starts outside VCS successfully, you can attempt to configure the Server Configuration with the VCS framework.

Reviewing error log files

If you face problems while using JBoss Application Server or the agent for JBoss Application Server, use the log files described in this section to investigate the problems.

Reviewing cluster log files

In case of problems while using the agent for JBoss Application Server, you can access the engine log file for more information about a particular resource. The engine log file is located at `/var/VRTSvcs/log/engine_A.log`.

Using trace level logging

The ResLogLevel attribute controls the level of logging that is written in a cluster log file for each JBoss Application Server resource. You can set this attribute to TRACE, which enables very detailed and verbose logging.

The logs for the JBoss resources are located at `/var/VRTSvcs/log/JBoss_A.log`.

If you set ResLogLevel to TRACE, a very high volume of messages are produced. Veritas recommends that you localize the ResLogLevel attribute for a particular resource.

The LogDbg attribute should be used to enable the debug logs for the ACCLib-based agents when the ACCLIB version is 6.2.0.0 or later and the VCS version is 6.2 or later.

To localize ResLogLevel attribute for a resource

1. Identify the resource for which you want to enable detailed logging.
2. Localize the ResLogLevel attribute for the identified resource:

   ```
   # hares -local Resource_Name ResLogLevel
   ```

3. Set the ResLogLevel attribute to TRACE for the identified resource:

   ```
   # hares -modify Resource_Name ResLogLevel TRACE -sys SysA
   ```

4. Note the time before you begin to operate the identified resource.
5. Test the identified resource. The function reproduces the problem that you are attempting to diagnose.
6. Note the time when the problem is reproduced.
7. Set the ResLogLevel attribute back to INFO for the identified resource:

   ```
   # hares -modify Resource_Name ResLogLevel INFO -sys SysA
   ```

8. Save the configuration changes.

   ```
   # haconf -dump
   ```

9. Review the contents of the log file.

   Use the time noted in the previous steps to diagnose the problem.

   You can also contact Veritas support for more help.

To enable debug logs for all resources of type JBoss

◆ Enable the debug log.

```
# hatype -modify JBoss LogDbg DBG_5
```

To override the LogDbg attribute at resource level

◆ Override the LogDbg attribute at the resource level and enable the debug logs for the specific resource.

```
# hares -override JBoss LogDbg
# hares -modify JBoss LogDbg DBG_5
```

Troubleshooting the configuration for IMF

If you face problems with the IMF configuration or functionality, consider the following:

- Ensure that the following attributes are configured with appropriate values.
  - AgentFile
  - IMF
  - IMFRegList
    If IMFRegList is not configured correctly, the JBoss Application Server resources that have been registered for IMF get unregistered every time the monitor function is run.
- If you have configured the required attributes to enable the JBoss Application Server agent for IMF, but the agent is still not IMF-enabled, restart the agent. The imf_init function runs only when the agent starts up, so when you restart the agent, imf_init runs and initializes the JBoss Application Server agent to interface with the AMF kernel driver.
- You can run the following command to check the value of the MonitorMethod attribute and to verify that a resource is registered for IMF.

  ```
  # hares -value resource MonitorMethod system
  ```

  The MonitorMethod attribute specifies the monitoring method that the agent uses to monitor the resource:
  - Traditional—Poll-based resource monitoring
  - IMF—Intelligent resource monitoring
- You can use the `amfstat` to see a list of registered PIDs for a JBoss resource.

  The output of the `ps -ef` command:

```
jboss   5588  0.1 12.71638480996672 ?    S 18:27:27
5:48 /opt/jdk1.6.0_26/bin/java -Dprogram.name=run.sh -server
-Xms1303m -Xmx1303m -XX:MaxPermSize=256m -Dorg.jboss.resolver.
warning=true -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.
dgc.server.gcInterval=3600000 -Dsun.lang.ClassLoader.
allowArraySyntax=true -Djava.net.preferIPv4Stack=true -Djava.
endorsed.dirs=/lhome/jboss/jboss-eap-5.1/jboss-as/lib/endorsed
-classpath /lhome/jboss/jboss-eap-5.1/jboss-as/bin/run.jar:/opt
/jdk1.6.0_26/lib/tools.jar org.jboss.Main -c production -Djboss.
server.base.dir=/lhome/jboss/jboss-eap-5.1/jboss-as/server -Djboss.
server.base.url=file:/lhome/jboss/jboss-eap-5.1/jboss-as/server
-Djboss.service.binding.set=ports-default -Djava.rmi.server.hostname=
127.0.0.1 -Djboss.bind.address=127.0.0.1
```

The `amfstat` command shows the PIDs monitored by the JBoss Application Server agent.

```
$ /opt/VRTSamf/bin/amfstat

AMF Status Report

Registered Reapers (1):
=======================
 RID     PID     MONITOR    TRIGG   REAPER
 2       6505      2        0       JBoss

Process ONLINE Monitors (2):
==============================
 RID        R_RID    PID          GROUP
 9           2       7241        JBoss_Prod
 11          2       13869       JBoss_Web
```

- Run the following command to set the ResLogLevel attribute to TRACE. When you set ResLogLevel to TRACE, the agent logs messages in the JBoss_A.log file.

  ```
  # hares -modify ResourceName ResLogLevel TRACE
  ```

  For more information about the ResLogLevel attribute, See “JBoss Application Server agent attributes” on page 28.

- Run the following command to view the content of the AMF in-memory trace buffer.

  ```
  # amfconfig -p dbglog
  ```

Known issues

This release of the agent for JBoss Application Server has the following known issues:

Problem

An error message might appear when you run the `hares -offline` command to take a resource offline.

Description

When a resource is taken offline, it is unregistered from the AMF module. However, the imf_register function attempts to unregister the resource again.

Workaround

It is safe to ignore this error message.

Known issues

This release of the agent for JBoss Application Server has the following known issues:

Warning message appears when the process for JBoss Application Server Configuration runs

At times, when the process for JBoss Application Server Configuration runs, the following messages might appear:

```
ffdc:20037:<JBoss::GetResourceState> Did not find all required processes.
Returning [100]
ffdc:20038:<JBoss::GetResourceState> Populating global running proclist
with [0] PIDs []
ffdc:20126:<main::MonitorEntryPoint> Resource [Jboss_res] NOT online
with state [100]
after first level check.
ffdc:10271:VCSagentFW:entryPointExit:Subroutine VCSagentFW::entryPoint
Exit called with:
Exit code [100]
ffdc:10272:VCSagentFW:entryPointExit:Exiting entry point [monitor] with
exit code [100]
ffdc:10274:VCSagentFW:entryPointExit:Generating FFDC messages for
resource [Jboss_res]
```

It is safe to ignore these messages.

Error message might appear when you run the `hares -offline` command

When a resource is taken offline, it is unregistered from the AMF module. However, the imf_register function attempts to unregister the resource again, and consequently, displays an error message when you run the `hares -offline` command.

It is safe to ignore this error message.

Sample Configurations

This appendix includes the following topics:

- About sample configurations for the agents for JBoss Application Server
- Sample service group configurations for JBoss Application Server
- Sample agent type definition type for JBoss Application Server
- Sample configuration files
- Sample IMF configurations

About sample configurations for the agents for JBoss Application Server

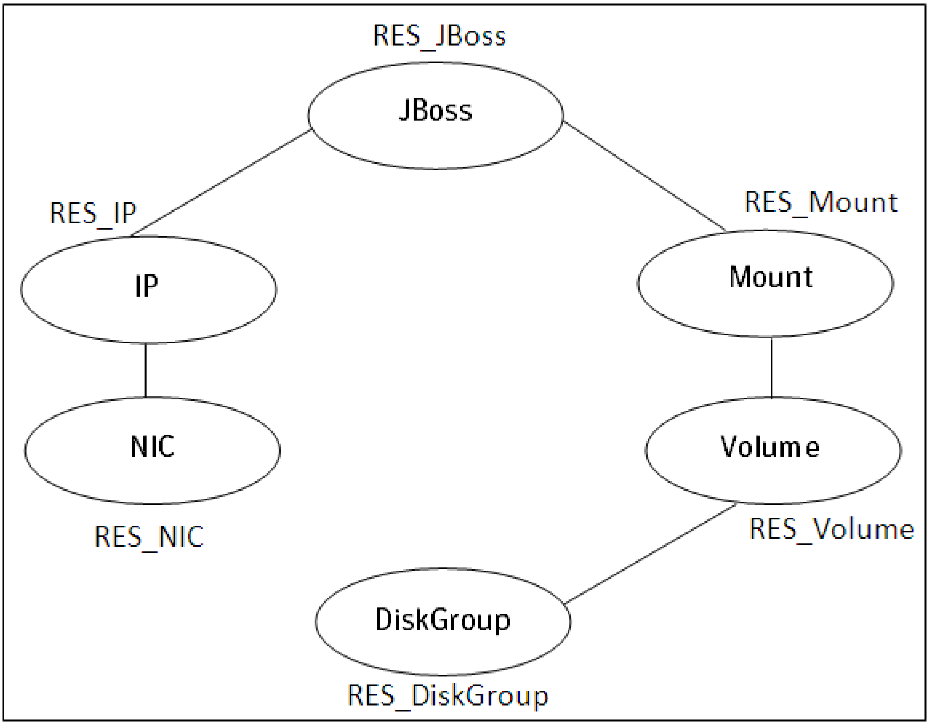
The sample configuration graphically depicts the resource types, resources, and resource dependencies within the service group. Review these dependencies carefully before configuring the agents for JBoss Application Server. For more information about these resource types, refer to the *Cluster Server Bundled Agents Reference Guide*.

Sample service group configurations for JBoss Application Server

This section includes sample service groups configurations in a VCS environment.

Figure A-1 shows a service group with a JBoss Application Server Configuration running in a VCS environment.

Figure A-1 Sample service group for a JBoss Application Server Configuration



Sample agent type definition type for JBoss Application Server

```
type JBoss (
static boolean AEPTimeout = 1
static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
static str AgentDirectory = "/opt/VRTSagents/ha/bin/JBoss"
static str ArgList[] = { ResLogLevel, State, IState, MonitorProgram,
  User, Version, JBossHome, Component, ServerConfig, ServerBaseDir,
  ConfigurationDir, ManagementAddress, MasterHCAddress, ServerName,
  EnvFile, AdminUser, AdminPassword,  StartArgs, StopArgs, BindingSet,
  JNDIUrl, SecondLevelMonitor, ServiceName, UseSystemD }
str ResLogLevel = INFO
str MonitorProgram
str User
str Version
str JBossHome
str Component
str ServerConfig
str ServerBaseDir
str ConfigurationDir
str ManagementAddress
str MasterHCAddress
str ServerName
str EnvFile
str AdminUser
str AdminPassword
str StartArgs
str StopArgs
str BindingSet
str JNDIUrl
int SecondLevelMonitor = 0
str ServiceName
boolean UseSystemD = 0
```

Sample configuration files

A sample `main.cf` file for a JBoss Application Server 6.0 configuration is as follows:

```
include "types.cf"
include "JBossTypes.cf"

cluster JBoss_cluster(
)

system NodeA(
)

system NodeB(
)

group JBoss_grp (
    SystemList = { NodeA = 0,  NodeB = 1 }
)

IP Jboss_ip (
    Device = eth4
    Address = "166.98.9.163"
    NetMask = "255.255.252.0"
    PrefixLen = 64
)

JBoss Jboss_6 (
    ResLogLevel = TRACE
    User = jboss
    Version = "6.0"
    JBossHome = "/opt/Jboss/EAP-6.0.0.GA/jboss-eap-6.0"
    Component = Standalone
    ServerConfig = "standalone.xml"
    ServerBaseDir = "/opt/Jboss/EAP-6.0.0.GA/jboss-eap-6.0/standalone"
    ConfigurationDir = "/opt/Jboss/EAP-6.0.0.GA/jboss-eap-6.0/
      standalone/configuration"
    ManagementAddress = "127.0.0.1:9999"
    SecondLevelMonitor = 1
    UseSystemD = 1
)

NIC Jboss_nic (
    Device = eth4
)

Volume vol_res (
    Volume Jboss_vol
    DiskGroup = Jboss_dg
)

DiskGroup dg_res (
    DiskGroup = jboss_dg
)

Mount mount_res (
    MountPoint = "/opt/JBoss"
    BlockDevice = "/dev/vx/dsk/Jboss_dg/Jboss_vol"
    FSType = ext4
    MountOpt = rw
    FsckOpt = "-y"
)

Jboss_6 requires Jboss_ip
Jboss_ip requires Jboss_nic
Jboss_6  requires mount_res
```

```
mount_res requires vol_res
vol_res requires dg_res
```

A sample `main.cf` file for a JBoss Application Server 5.x configuration is as follows:

```
include "types.cf"
include "JBossTypes.cf"

cluster JBoss_cluster(
)

system NodeA(
)

system NodeB(
)

group JBoss_grp (
    SystemList = { NodeA = 0,  NodeB = 1 }
)

IP Jboss_ip (
    Device = eth4
    Address = "166.98.9.163"
    NetMask = "255.255.252.0"
    PrefixLen = 64
)

JBoss jboss_5 (
    ResLogLevel = TRACE
    User = jboss
    Version = "5.0"
    JBossHome = "/opt/Jboss/EnterprisePlatform-5.1.1/jboss-eap-5.1/
      jboss-as"
    ServerConfig = production
    ServerBaseDir = "/opt/Jboss/EnterprisePlatform-5.1.1/jboss-eap-5.1/
      jboss-as/server"
    AdminUser = admin
    AdminPassword = HVNtKVk
    BindingSet = ports-default
    JNDIUrl = "jnp://127.0.0.1:1099"
)

NIC Jboss_nic (
    Device = eth4
)

Volume vol_res (
    Volume Jboss_vol
    DiskGroup = Jboss_dg
)

DiskGroup dg_res (
    DiskGroup = jboss_dg
)

Mount mount_res (
    MountPoint = "/opt/JBoss"
    BlockDevice = "/dev/vx/dsk/Jboss_dg/Jboss_vol"
    FSType = ext4
    MountOpt = rw
    FsckOpt = "-y"
)

Jboss_5 requires Jboss_ip
Jboss_ip requires Jboss_nic
Jboss_5  requires mount_res
mount_res requires vol_res
vol_res requires dg_res
```

A sample `main.cf` file for a domain-mode JBoss Application Server 9.0 configuration is as follows:

```
include "types.cf"
include "JBossTypes.cf"

cluster Jboss_cluster (
)

system nodeA (
)

system nodeB (
)

group SVG_host (
    SystemList = { nodeA = 0, nodeB = 1 }
    Parallel = 1
```

```
)

JBoss wild_host (
    Critical = 0
    ResLogLevel = TRACE
    User = root
    Version = "9.0"
    JBossHome = "/opt/wildfly-9.0.0.Final"
    Component = Domain
    ServerConfig = "domain.xml"
    ServerBaseDir = "/opt/wildfly-9.0.0.Final/domain"
    ConfigurationDir = "/opt/wildfly-9.0.0.Final/domain/configuration"
    ManagementAddress @nodeA = "10.209.56.47:9990"
    ManagementAddress @nodeB = "10.209.56.48:9990"
    HostName @nodeA = master
    HostName @nodeB = slave
    EnableCmdOutLogging = 0
)

// resource dependency tree
//
//      group SVG_host
//      {
//      JBoss wild_host
//      }

group SVG_s1 (
    SystemList = { nodeA = 0, nodeB = 1 }
    Parallel = 1
)

JBoss wild_s1 (
    Critical = 0
    User = root
    Version = "9.0"
    JBossHome = "/opt/wildfly-9.0.0.Final"
    Component = Server
    ServerConfig = "domain.xml"
    ServerBaseDir = "/opt/wildfly-9.0.0.Final/domain"
    ConfigurationDir = "/opt/wildfly-9.0.0.Final/domain/configuration"
    ManagementAddress @nodeA = "10.209.56.47:9990"
    ManagementAddress @nodeB = "10.209.56.48:9990"
    ServerName = server-one
    HostName @nodeA = master
    HostName @nodeB = slave
    AdminUser = admin
    AdminPassword = bphNepE
)

requires group SVG_host online local firm


// resource dependency tree
//
//      group SVG_s1
//      {
//      JBoss wild_s1
//      }

group SVG_s2 (
    SystemList = { nodeA = 0 }
    Parallel = 1
)

JBoss wild_s2 (
    Critical = 0
    User = root
    Version = "9.0"
    JBossHome = "/opt/wildfly-9.0.0.Final"
    Component = Server
    ServerConfig = "domain.xml"
    ServerBaseDir = "/opt/wildfly-9.0.0.Final/domain"
    ConfigurationDir = "/opt/wildfly-9.0.0.Final/domain/configuration"
    ManagementAddress = "10.209.56.47:9990"
    ServerName = server-two
    HostName = master
    AdminUser = admin
)

requires group SVG_host online local firm

// resource dependency tree
//
//      group SVG_s2
//      {
```

```
//      JBoss wild_s2
//      }
```

Sample IMF configurations

An example of a type definition file for a JBoss Application Server agent that is IMF-enabled is as follows.

In this example, the IMF-related attributes are set to the following values:

AgentFile	/opt/VRTSvcs/bin/Script51Agent
IMF{}	{ Mode=2, MonitorFreq=5, RegisterRetryLimit=3 }
IMFRegList[]	{ JBossHome, ServerBaseDir, ServerConfig, User }

```
type JBoss (
      static int IMF{} = { Mode=2, MonitorFreq=5, RegisterRetryLimit=3 }
      static str IMFRegList[] = { JBossHome, ServerBaseDir, ServerConfig
      , User }
      static str AgentDirectory = "/opt/VRTSagents/ha/bin/JBoss"
      static str AgentFile = "/opt/VRTSvcs/bin/Script51Agent"
      static str ArgList[] = { ResLogLevel, State, IState, MonitorProgram,
      User, Version, JBossHome, Component, ServerConfig, ServerBaseDir,
      ConfigurationDir, ManagementAddress, MasterHCAddress, ServerName,
      EnvFile, AdminUser, AdminPassword, StartArgs, StopArgs, BindingSet,
      JNDIUrl, SecondLevelMonitor }
      static boolean AEPTimeout = 1
      str ResLogLevel = INFO
      str MonitorProgram
      str User
      str Version
      str JBossHome
      str Component
      str ServerConfig
      str ServerBaseDir
      str ConfigurationDir
      str ManagementAddress
      str MasterHCAddress
      str ServerName
      str EnvFile
      str AdminUser
      str AdminPassword
      str StartArgs
      str StopArgs
      str BindingSet
      str JNDIUrl
      int SecondLevelMonitor
)
```

A sample resource configuration from the `/etc/VRTSvcs/conf/config/main.cf` file is as follows:

For JBoss Application Server 5.x:

```
JBoss jboss_5 (
         ResLogLevel = TRACE
         User = jboss
         Version = "5.0"
         JBossHome = "/opt/Jboss/EnterprisePlatform-5.1.1/
         jboss-eap-5.1/jboss-as"
         ServerConfig = production
         ServerBaseDir = "/opt/Jboss/EnterprisePlatform-5.1.1/
         jboss-eap-5.1/jboss-as/server"
         AdminUser = admin
         AdminPassword = HVNtKVk
         BindingSet = ports-default
         JNDIUrl = "jnp://127.0.0.1:1099"
         )
```

For JBoss Application Server 6.0:

```
JBoss Jboss_6 (
          ResLogLevel = TRACE
          User = jboss
          Version = "6.0"
          JBossHome = "/opt/Jboss/EAP-6.0.0.GA/jboss-eap-6.0"
          Component = Standalone
          ServerConfig = "standalone.xml"
          ServerBaseDir = "/opt/Jboss/EAP-6.0.0.GA/jboss-eap-6.0/
          standalone"
          ConfigurationDir = "/opt/Jboss/EAP-6.0.0.GA/jboss-eap-6.0/
          standalone/configuration"
          ManagementAddress = "127.0.0.1:9999"
          SecondLevelMonitor = 1
          )
```