Veritas High Availability Agent 5.0 for WebSphere Application Server Installation and Configuration Guide

AIX, HP-UX, Linux, Solaris

N19296C

symantec.

Veritas High Availability Agent 5.0 for WebSphere Application Server
Installation and Configuration Guide

Veritas High Availability Agent 5.0 for WebSphere Application Server

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014
www.symantec.com

Third-party legal notices

Third-party software may be recommended, distributed, embedded, or bundled with this Symantec product. Such third-party software is licensed separately by its copyright holder. All third-party copyrights associated with this product are listed in the accompanying release notes.

AIX is a registered trademark of IBM Corporation.
HP-UX is a registered trademark of Hewlett-Packard Development Company, L.P.
Linux is a registered trademark of Linus Torvalds.
Solaris is a trademark of Sun Microsystems, Inc.

Technical support

For technical assistance, visit http://support.veritas.com and select phone or email support. Use the Knowledge Base search feature to access resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and our customer email notification service.

Contents

Chapter 1

Introducing the Veritas Agent for WebSphere Application Server

The Veritas High Availability Agents monitor specific resources within an enterprise application, determine the status of these resources, and start or stop them according to external events. The Veritas High Availability Agent for WebSphere Application Server provides high availability for WebSphere Application Server in a cluster environment.

What's new in this agent

- Added support for VCS version 5.0.
- Added support for ACC library 5.0, that is compliant with VCS 4.x.

Supported software

The Veritas High Availability Agent for WebSphere Application Server supports the following environments. Contact your Symantec sales representative for the most recent list of supported environments.

Environment	Supported versions
Veritas Cluster Server	AIX—VCS 4.0, 4.1, 5.0
	HP-UX—VCS 4.1, 5.0
	Linux—VCS 4.0, 4.1, 5.0
	Solaris—VCS 4.0, 4.1, 5.0

Environment	Supported versions
Veritas Application Director	AIX—VAD 1.0 Linux—VAD 1.0 Solaris—VAD 1.0
ACC Library	5.0
Operating Systems	AIX 5.1, 5.2, 5.3 on pSeries Red Hat Enterprise Linux 3.0, 4.0 on Intel Solaris 8, 9 on SPARC HP-UX 11i V2 on PA-RISC
WebSphere Application Server	5.0, 5.1, 6.0

About the agent for WebSphere

The Veritas High Availability Agent for WebSphere Application Server brings the server instances online, monitors the processes and server state on all systems in the cluster, detects server failure, and shuts down the server when directed or when circumstances indicate that a failover is required.

The agent supports three types of WebSphere Application Server instances:

- Deployment Manager
- Node Agent
- Application Server

Agent operations

The agent for WebSphere consists of resource type declarations and agent executables. The agent executables are logically organized into the following operations:

- online
- offline
- monitor
- clean

These operations are described below.

Online

The online operation is responsible for starting a WebSphere Application Server. Online performs the following tasks:

- Online operation verifies that the WebSphere Application Server instance is not already online.
- The operation determines the version of the WebSphere Application Server software.
- The operation starts the WebSphere Application Server instance by executing the appropriate start script, which is supplied by the WebSphere installation program. The script executed depends upon the type of server being started:

Server Type	Start Command
Deployment Manager	`<WAS_HOME>/<binDir>/startManager.sh`
Node Agent	`<WAS_HOME>/<binDir>/startNode.shs`
Application Server	`<WAS_HOME>/<binDir>/startServer.sh`

Offline

The offline operation is responsible for stopping a WebSphere Application Server instance. Offline performs the following tasks:

- The offline operation verifies that the WebSphere Application Server instance is not already offline.
- The operation determines the version of the WebSphere Application Server software.
- The operation stops the WebSphere Application Server instance by executing the appropriate stop script, which is supplied by the WebSphere installation program. The script executed depends upon the type of server being stopped:

Server Type	Stop Command
Deployment Manager	`<WAS_HOME>/<binDir>/stopManager.sh`
Node Agent	`<WAS_HOME>/<binDir>/stopNode.sh`
Application Server	`<WAS_HOME>/<binDir>/stopServer.sh`

Monitor

The monitor entry point is responsible for monitoring the state of WebSphere Application Servers on all nodes in the cluster. Monitor performs the following tasks:

- First-level monitoring quickly checks for the existence of the system process (the Java Virtual Machine) that represents the WebSphere Application Server instance. It determines the process existence by scanning the system process table and searching for strings in the process command line that uniquely identify the JVM process associated with the WebSphere Application Server instance. These search strings include the values specified in resource attributes WAS_HOME, WAS_NODE, and ServerName.
- If second-level monitoring is enabled (if SecondLevelMonitor > 0), the monitor entry point performs a deeper, more thorough state check of the WebSphere Application Server. Second-level monitoring uses the IBM-supplied utility program `serverStatus.sh`. The output from this program is parsed to confirm the server is running.
  When enabled, the integer value specified in attribute SecondLevelMonitor determines how frequently the program is executed. For example, if SecondLevelMonitor is set to 1, the monitor entry point executes `serverStatus.sh` during each monitor interval. If SecondLevelMonitor is set to 3, the monitor entry point executes `serverStatus.sh` every third monitor interval. This mechanism lets you control the system load generated by monitoring.

  Note: The `serverStatus.sh` script spawns a Java program that establishes a connection to the WebSphere Application Server. Spawning a JVM every monitor interval places additional load on the system. If performance is more important than a second-level state check, then consider disabling second-level monitoring and only performing the first-level process check.

- The monitor entry point executes a custom monitor program specified in the attribute MonitorProgram. This program does not execute if either the first- or second-level monitor reports that the resource is offline. You can omit second-level monitoring, and attempt running a custom monitor check immediately after first-level monitoring.
  This feature allows VCS or VAD administrators to define custom programs that determine the state of the WebSphere Application Server. For example, the administrator may want to test the status of a J2EE component running inside the server and ensure that the underlying application is functioning properly. Refer to the full description for the attribute MonitorProgram in "Agent attributes" on page 20.

Clean

The clean entry point removes any WebSphere Application Server instance processes remaining after a fault event or after an unsuccessful attempt to online or offline the resource. Clean operation performs the following tasks:

- The clean operation kills the process that starts the WebSphere Application Server instance. It is unlikely that this process exists, but it needs to be removed if for some reason it still exists during clean.
- The operation kills the process that stops the WebSphere Application Server instance. It's unlikely this process will exist, but it needs to be removed if for some reason it still exists during clean.
- The operation kills the JVM process for the WebSphere Application Server instance. This process is identified by searching the system process table using the values specified in attributes WAS_HOME, WAS_NODE, and ServerName.

Chapter 2

Installing the Veritas Agent for WebSphere Application Server

This chapter describes how to install the Veritas Hight Availability Agent for WebSphere Application Server. You must install the agent for WebSphere on all the systems that will host a WebSphere service group.

Prerequisites

Before installing the agent for WebSphere, ensure that you meet the following requirements.

Prerequisites for installing agent in a VCS environment

- Install and configure Veritas Cluster Server.
- If you are working in a VCS 4.x environment, install the ACC Library 5.0 or later (`VRTSacclib`) if it is not already installed. For more information, refer to "About the ACC library" on page 14.
- Remove any prior version of this agent.

Prerequisites for installing agent in a VAD environment

- Install and configure Veritas Application Director.
- Remove any prior version of this agent.

About the ACC library

The operations for Veritas High Availability Agent for WebSphere Application Server depend on a set of Perl modules known as the ACC library. The library must be installed on each system in the cluster that will run the WebSphere Application Server agent. The ACC library contains common, reusable functions that perform tasks such as process identification, logging, and system calls.

Note: If you are installing the agent for WebSphere in a VCS 5.0 or VAD environment, do not install the ACC library package separately. If you are installing the agent in a VCS 4.x environment, you must install the ACC library package before installing the agent.

To install or update the ACC library package, locate the library and related documentation on the agent CD and in the compressed agent tar file.

Upgrading the agent for WebSphere

If an older version of the agent software is installed on the target system, first follow the instructions in this guide for removing existing agent software. Refer to "Uninstalling the Veritas Agent for WebSphere Application Server" on page 37 for the uninstallation procedure. Then follow the instructions below to install the new agent software.

Installing the agent in a VCS environment

Install the Veritas High Availability Agent for WebSphere Application Server on each node in the cluster.

To install the agent on AIX systems

1. Log in as `root`.
2. Go to the *<cd_mount>*`/aix/application/websphere_agent/`*<vcs_version>*`/pkgs` directory.
3. Install the package:
   ```
   # installp -ac -d VRTSvcswas5.rte.bff VRTSvcswas5.rte
   ```

To install the agent on HP-UX systems

1. Log in as `root`.
2. Go to the *<cd_mount>*`/hpux/application/websphere_agent/`*<vcs_version>*`/pkgs` directory.
3. Install the package:
   ```
   # swinstall -s `pwd` VRTSvcswas5
   ```

To install the agent on Linux systems

1. Log in as `root`.
2. Go to the *<cd_mount>*`/linux/application/weblogic_agent/`*<vcs_version>*`/rpms` directory.
3. Install the package:
   ```
   # rpm -ihv VRTSvcswas5-VersionNumber.rpm
   ```

To install the agent on Solaris systems

1. Log in as `root`.
2. Go to the *<cd_mount>*`/solaris/sparc/application/weblogic_agent/`*<vcs_version>*`/pkgs` directory.
3. Install the package:
   ```
   # pkgadd -d . VRTSvcswas5
   ```

Installing the agent in a VAD environment

This section describes the procedure to install the agent for WebSphere in a VAD environment.

To install the agent

1 Set up an `ssh` communication from the system where you are running the agent installation process, to the systems on which you want to install the agent.

Note: After installing the agent, you must add the agent types file to the Policy Master database. So, either set up the communication on a running PM system, or set up a communication from the system where you are running the installation to the PM system.

---

2 Depending upon the environment in which you are installing the agent, choose either the VAD installation software disk or the agent pack disk.

3 Mount the software disc on the system from where you are running the agent installation process.

4 Go to this directory:

Software disc	Directory
Agent pack CD	`/<platform>/application/websphere_agent/1.0`
VAD installation software disc	`high_availability_agents`

5 Enter this command to begin installing the agent:

```
./installagpack
```

6 Enter the names of the systems on which you want to install the agent.

The installer proceeds to install the agent on the systems. You can view the installation logs in the `/var/VRTS/install/logs` directory.

Chapter 3

Configuring the Veritas Agent for WebSphere Application Server

After installing the agent, you must import the agent configuration file. Then you can create and configure WebSphere Application Server resources. Before configuring a resource, review the attributes table that describes the WebSphere resource type and its attributes. This chapter includes the resource type definition file for your reference.

Importing the agent types file

To use the WebSphere Application Server agent, you must import the agent configuration file into the cluster engine.

To import the WebSphere5Types.cf configuration file to work with VCS

Perform the following steps using the Veritas Cluster Server graphical user interface.

1. Start the Veritas Cluster Manager GUI 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 Version	File to be selected
VCS 4.x	`/etc/VRTSvcs/conf/sample_WebSphere5/WebSphere5Types.cf`
VCS 5.0	`/etc/VRTSagents/ha/conf/WebSphere5/WebSphere5Types.cf`

4 Click **Import**.

5 Save the VCS configuration.

The WebSphere type is now imported to the VCS engine. You can now create WebSphere Application Server resources. For additional information about using Cluster Manager, refer to the *Veritas Cluster Server User's Guide*.

To import the WebSphere5Types.*<platform>*.xml configuration file to work with VAD

Before beginning to work with the agent for WebSphere, you must add the Veritas High Availability agent resource types to the Policy Master database configuration.

1 If you are running the agent installation on the PM system, go to step 2.
If you are running the agent installation on a system other than the PM system, set up `ssh` communication between the system on which you are running the agent installation and the PM system.

2 Ensure that the PM daemon is running:
`haadmin -state`
The output must show ClusterState as RUNNING.

3 Depending upon the platform on which you are installing the agent and the environment, mount either the VAD installation disk or the agent pack disk on the system.

4 Go to this directory:

Software disc	Directory
Agent pack CD	`/<platform>/application/websphere_agent/1.0`
VAD installation software disc	`high_availability_agents`

5 Enter this command to add resource types to the PM configuration database:
`./installagpack -addtypes`

6 When prompted, enter the virtual IP address of the PM system.

The installer adds the agent types to the PM database configuration, and copies the appropriate `types.xml` files to the PM system. You can view the installation logs in the `/var/VRTS/install/logs` directory.

Sample agent type definition

If you are working with VCS 4.x

After importing the agent types into the cluster, if you save the configuration on your system disk using the `haconf -dump` command, you can find the `WebSphere5Types.cf` file in the `/etc/VRTSvcs/conf/config` cluster configuration directory.

An excerpt from this file follows.

```
type WebSphere5 (
   static str ArgList [] = { ResLogLevel, State, IState, ServerName,
      WAS_NODE, WAS_HOME ,User, ServerProfile, ServerType,
      StartOptions, StopOptions, MonitorProgram, SecondLevelMonitor}
   str  ResLogLevel = INFO
   str  ServerName
   str  WAS_NODE
   str  WAS_HOME
   str  User
   str  ServerProfile
   str  ServerType
   str  StartOptions
   str  StopOptions
   str  MonitorProgram
   int  SecondLevelMonitor
   )
```

If you are working with VCS 5.0

After importing the agent types into the cluster, if you save the configuration on your system disk using the `haconf -dump` command, you can find the `WebSphere5Types.cf` file in the `/etc/VRTSagents/ha/conf/config` cluster configuration directory.

An excerpt from this file follows.

```
type WebSphere5 (
    static str AgentFile = "/opt/VRTSvcs/bin/ScriptAgent"
    static str AgentDirectory = "/opt/VRTSagents/ha/bin/WebSphere5"
   static str ArgList [] = { ResLogLevel, State, IState, ServerName,
      WAS_NODE, WAS_HOME ,User, ServerProfile, ServerType,
      StartOptions, StopOptions, MonitorProgram, SecondLevelMonitor}
   str  ResLogLevel = INFO
   str  ServerName
   str  WAS_NODE
```

```
str  WAS_HOME
str  User
str  ServerProfile
str  ServerType
str  StartOptions
str  StopOptions
str  MonitorProgram
int  SecondLevelMonitor
)
```

If you are working with VAD

After installing the agent, go to the `/etc/VRTSagents/ha/conf/WebSphere5/` directory to view the `WebSphere5Types.<platform>.xml` agent definition file.

Agent attributes

Refer to the following table of required and optional attributes while configuring the agent. For important configuration tips, refer to “Important considerations while configuring the agent” on page 25.

Required attributes

Required attribute	Description
ResLogLevel *String*	The logging detail performed by the agent for the resource. Valid values are: **ERROR**: Only logs error messages. **WARN**: Logs above plus warning messages. **INFO**: Logs above plus informational messages. **TRACE**: Logs above plus trace messages. **TRACE** is very verbose and should only be used during initial configuration or for troubleshooting and diagnostic operations. Example: **TRACE** Default: **INFO**
SecondLevelMonitor *Integer*	Specifies if second-level monitor is enabled and how frequently it is performed. Second-level monitor is a deeper, more thorough state check of the WebSphere resource, performed by executing the IBM-supplied utility program `serverStatus.sh`. The output from this program is parsed to confirm the server status is running. The integer value specified by this attribute determines how frequently the second-level monitor program is executed. For example, if SecondLevelMonitor is set to **1**, the monitor entry point will execute `serverStatus.sh` during each monitor interval. A value of **3** executes the program every third monitor interval. If SecondLevelMonitor is set to **0**, the monitor entry point will never perform the second-level monitor. Example: **1** Default: **0**

Required attribute	Description
ServerName *String*	Contains the server name assigned to the WebSphere Server during its installation. In Network Deployment configurations, the default ServerName for Deployment Managers is `dmgr` and the default ServerName for the Node Agents is `nodeagent`, but these names are not mandatory. Refer to "Uniquely identifying WebSphere instances" on page 25 for information on setting this attribute. Example: **`server1`** Default: ""
ServerProfile *String*	Server profile name of the WebSphere Server instance. This attribute is applicable to WebSphere version 6 only, and must be null if the WebSphere major version number is 5. Specifying this attribute is required if the resource manages a WebSphere Application Server version 6.0. Example: **`Dmgr01`** Default: ""
ServerType *String*	Type of WebSphere Application Server that the cluster will manage. Valid names are as follows: ■ **`DeploymentManager`**: Resource is a Deployment Manager. ■ **`NodeAgent`**: Resource is a Node Agent. ■ **`ApplicationServer`**: Resource is an Application Server, which may be a stand-alone server or may be part of a Network Deployment and is a member of a WebSphere Cell. The agent uses this value to determine how to manage the WebSphere Application Server within a cluster. Refer to the WebSphere documentation for a full explanation of the purposes and use of each WebSphere Application Server type. Example: **`DeploymentManager`** Default: ""

Required attribute	Description
User *String*	The UNIX user name used to run the programs that start, stop, and monitor the WebSphere resource, which include the program specified in the MonitorProgram attribute. IBM recommends using the root account, but you may use any account. If User is not set to **root**, the user name must be synchronized across the systems within the cluster. In other words, the user name must resolve to the same UID and have the same default shell on each system in the cluster. Example: **root** Default: ""
WAS_HOME *String*	The absolute path to the WebSphere Application Server or WebSphere Application Server Network Deployment root installation directory. This attribute is used to locate programs executed by the agent. It is also where the `<binDir>/setupCmdLine.sh` file resides. The value is also used to uniquely identify the ServerType processes. Using WAS_HOME to uniquely identify an Application Server's process IDs requires that WAS_HOME be unique compared to WAS_HOME for all other WAS instances in the cluster. Refer to "Uniquely identifying WebSphere instances" on page 25 for information on setting this attribute. **Note:** Both WAS_HOME and WAS_ND_HOME are defined as WAS_HOME in the standard environment file `setupCmdLine.sh`, which is supplied with WebSphere. Example: **/ibm/was/v51/cell1/node2** Default: ""
WAS_NODE *String*	The WebSphere Node Name to which the server instance belongs. The Node Name is an administrative identifier that is internal to the WebSphere environment and is assigned when the node is installed. WebSphere requires that a Node Name must be unique within a WebSphere cell. Refer to "Uniquely identifying WebSphere instances" on page 25 for information on setting this attribute. Example: **was51c1n2** Default: ""

Optional attributes

Optional Attribute	Definition
MonitorProgram *String*	The full pathname and command-line arguments for an externally-provided custom monitor program. The program is executed within the security context of the UNIX account specified in attribute User. The program must be completely self-contained and independent, and it must return one of the following exit codes: **110** or **0**: The WebSphere Application Server is ONLINE. **100** or **1**: The WebSphere Application Server is OFFLINE. All other: The WebSphere Application Server state is UNKNOWN. Symantec recommends storing the external monitor program on the shared storage device, in the directory specified by the WAS_HOME attribute, to ensure the file is always available on the online system. Example: **/usr/WAS51/server1/bin/mymonitor.sh** Default: ""
StartOptions *String*	The command-line options that are passed to the WebSphere start script when it is executed within the online entry point. Multiple options should be separated by a space. Refer to the WebSphere product documentation for a list and description of supported start options. Example: **"-replacelog -trace"** Default: ""
StopOptions *String*	The command-line options that are passed to the WebSphere stop script when it is executed within the offline entry point. Multiple options should be separated by a space. Refer to the WebSphere product documentation for a list and description of supported stop options. Example: **"-replacelog -trace"** Default: ""

Uniquely identifying WebSphere instances

Set the WAS_HOME, WAS_NODE, and ServerName attributes such that the combined values are unique for each WebSphere Application Server instance.

You can *virtualize* a WebSphere instance using the cluster. You can easily manage a large set of WebSphere Application Servers in a single cluster, using shared storage and Virtual IP address assignments. WebSphere Application Servers can run on separate cluster nodes or can run concurrently on a single node. If WebSphere Application Servers run concurrently on a single node, you must ensure that the agent can uniquely identify each WebSphere Application Server on a host system that is running more than one WebSphere Application Server.

For unique identification, the agent's monitor and clean entry points use the values specified by attributes WAS_HOME, WAS_NODE, and ServerName to uniquely identify each running WebSphere Server JVM process.

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

Important considerations while configuring the agent

While configuring the agent, be sure of the following settings:

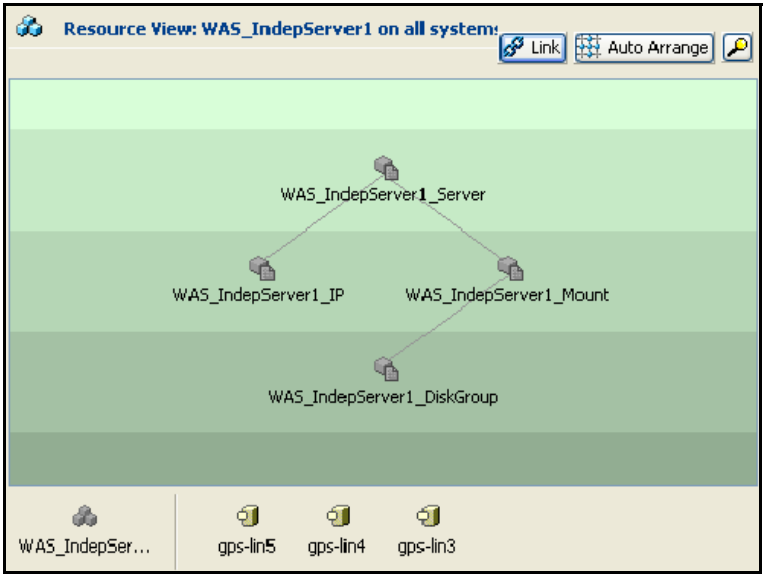
- The time required to fully start a WebSphere instance depends on the number, size, and complexity of Java applications started within the server. Be sure to compare the value of the OnlineTimeout attribute with the actual time required to fully initialize the WebSphere Application Server. Large WebSphere Application Server deployments may require a larger OnlineTimeout. Properly tuning this attribute ensures that the cluster does not time out the online entry point while a WebSphere Application Server is initializing.
- Allow ample time for the WebSphere Application Server to shut down completely before probing the resource to determine if the request to stop was successful. Depending upon the environment, you may need to adjust the OfflineTimeout attribute for this resource to allow the instance ample time to shut down. Properly tuning this attribute ensures that the cluster does not time out the offline entry point while a WebSphere Application Server is completing a graceful shut down.

- Once a WebSphere Application Server is placed under cluster control, do not attempt to start or stop the instance without using a cluster interface. Only use the Web Console, Java Console, or command-line interface, to start or stop a managed WebSphere instance.

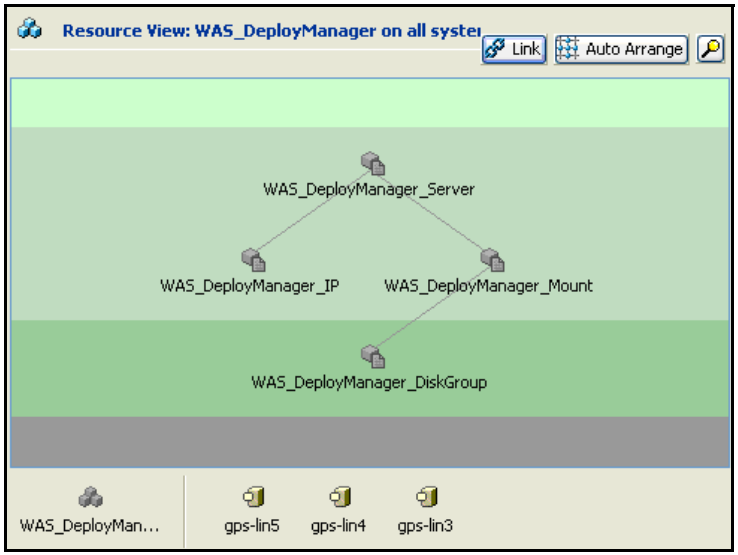
Service group configuration options

Your WebSphere deployment type and strategy determines the number of service groups in a cluster required and the number of WebSphere Application Servers managed within each service group. Although not comprehensive, the examples in the following figures depict common scenarios to consider.

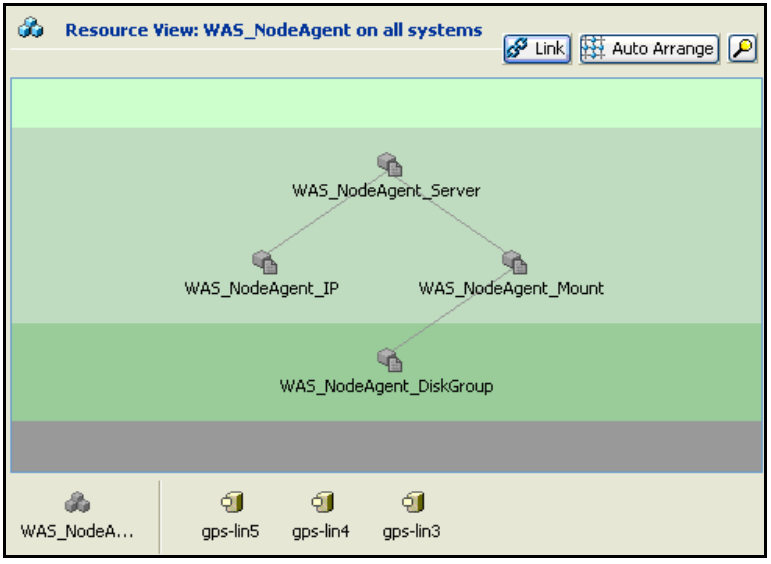
Following figure depicts a service group that manages one independent, stand-alone WebSphere Application Server. This configuration does not include Cells, Deployment Managers, and Node Agents.



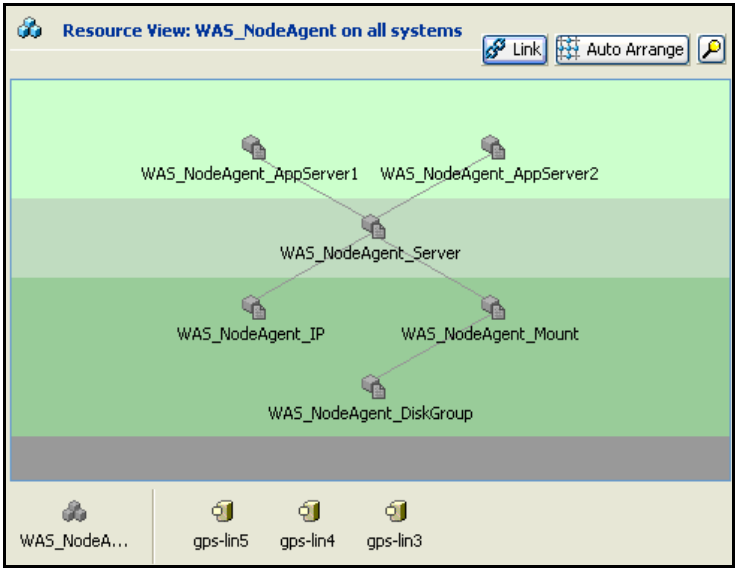
Following figure depicts a service group that manages a Deployment Manager Server. Other service groups manage WebSphere Servers of the type Node Agent and Application Server.



Following figure depicts a service group that manages a Node Agent Server. In this configuration, the cluster does not control the Application Servers managed by this Node Agent instance. Thus, the Node Agent Server may fully manage and monitor its managed Application Servers without conflict with the cluster.



Following figure depicts a service group that controls a Node Agent Server and its two managed Application Servers. In this configuration, the cluster controls the Application Servers that are managed by this Node Agent instance. Thus, the Node Agent Server should be configured to not monitor and restart its failed Application Servers, as this would conflict with cluster actions in response to the failure.



Chapter 4

Clustering WebSphere Application Servers

This chapter provides an overview of the clustering process, as well as information about configuring service groups on a cluster.

Overview of the clustering process

While various methods and procedures can be used to install and cluster a WebSphere Application Server, Symantec recommends the following general process:

Allocate shared disk resource for the WebSphere node

A WebSphere node is a logical group of WebSphere Application Servers that are located on the same physical machine. This machine is also called a host. Multiple WebSphere nodes can exist on a single node.

Symantec recommends installing each WebSphere node to be clustered on a separate, dedicated shared disk resource (e.g. LUN). Work with the appropriate administrative group in your organization to obtain a shared disk resource for the WebSphere node.

Create a Veritas disk group, volume, and file system

Create a Veritas disk group, volume, and file system on the shared disk resource allocated for the WebSphere node.

Although not recommended, WebSphere Application Servers can be clustered without using Veritas Volume Manager or Veritas File System. But the tight integration between the cluster, Volume Manager, and File System ensures a more comprehensive and resilient high availability solution for your WebSphere Application Server.

Obtain dedicated virtual IP addresses and host names

Obtain dedicated virtual IP addresses and host names required to support the WebSphere node IP network configuration.

Several configurations are possible. For example, a Node agent, which is an administrative process that manages all servers running on a WebSphere node, can share one IP address and host name with all of its managed servers. Alternatively, the Node agent and each of its managed servers could be assigned its own IP address and host name.

No matter which configuration you deploy, these network addresses and host names will be used exclusively by this WebSphere node, regardless of which system in the cluster is running it.

Obtain a dedicated user account if needed

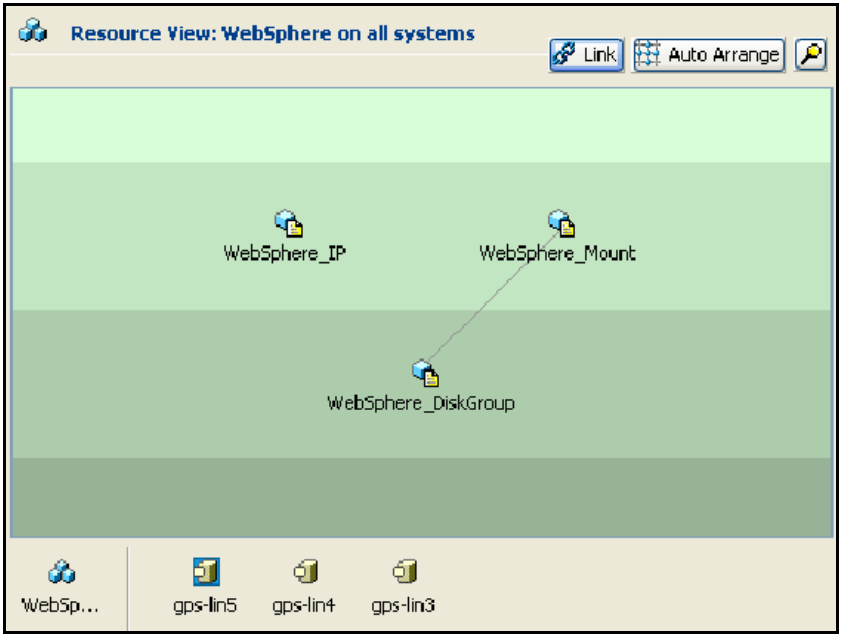
If the WebSphere Application Server will not run using the root account, obtain a dedicated UNIX account for the WebSphere Application Server. Refer to the description of attribute User for important instructions and requirements to create the account.

Create Service Group and supporting resources

First create a Service Group on a cluster to contain the resources supporting the WebSphere node.

Then create the appropriate cluster resources and links to place the previously created shared disk and networking objects under cluster control.

Test the service group configuration by placing it online. Your service group should appear similar to the following:



Install the WebSphere software

With the disk and network resources now available and online in the cluster, you are ready to install the WebSphere software.

Follow the instructions in the WebSphere product documentation and install the WebSphere Application Server software. Be sure to instruct the installation program to install the software on the shared disk file system previously established for this WebSphere node.

A well-designed directory structure for your WebSphere Application Server instances will simplify the cluster configuration and create a storage environment that is more intuitive and easier to manage. Assuming that all WebSphere Application Server instances will be clustered and installed on shared disk, Symantec recommends a directory structure similar to the following:

Directory	Purpose
`/WAS`	Top level directory under which all WebSphere nodes are installed.

Directory	Purpose
`/WAS/cell1`	Subdirectory under which all WebSphere nodes assigned to `cell1` are installed.
`/WAS/cell1/depmgr`	Subdirectory is the mount point for the shared disk resource dedicated to the Deployment Manager instance supporting `cell1`.
`/WAS/cell1/node1`	Subdirectory is the mount point for the shared disk resource dedicated to the WebSphere node named `node1`, which belongs to `cell1`. The WebSphere software supporting this node agent and its managed Application Servers is installed in this directory.
`/WAS/cell1/node2`	Subdirectory is the mount point for the shared disk resource dedicated to the WebSphere node named `node2`, which belongs to `cell1`. The WebSphere software supporting this node agent and its managed Application Servers is installed in this directory.
`/WAS/cell1/node3`	Subdirectory is the mount point for the shared disk resource dedicated to the WebSphere node named `node3`, which belongs to `cell1`. The WebSphere software supporting this node agent and its managed Application Servers is installed in this directory.

Continue with the same naming pattern for all remaining cells and WebSphere Application Servers.

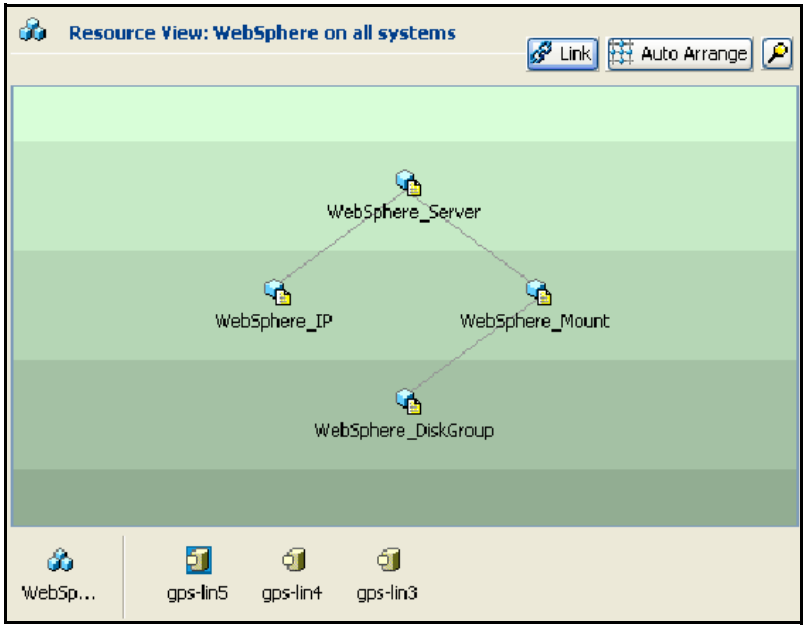
During the installation, be sure to set the node's Host Name to the dedicated virtual IP host name previously allocated to this node.

Finally, be sure to configure the server's port numbers to avoid conflicts with the port numbers of other WebSphere Application Servers that may be running simultaneously on the same system. Configuring the port numbers is especially important in a cluster environment where WebSphere nodes can be easily moved around the systems in the cluster in almost any combination.

Place the WebSphere Application Server under cluster control

After the WebSphere Application Server installation is complete, create a cluster resource using the agent for WebSphere to place the server under cluster control.

Your service group should now appear similar to the following:



Caution: Once a WebSphere Application Server is placed under cluster control, do not attempt to start or stop the instance without using a cluster interface. Only use the Web Console, Java Console, or command-line interface to start or stop a managed WebSphere instance.

Chapter 5

Uninstalling the Veritas Agent for WebSphere Application Server

Follow the steps below to remove the Veritas High Availability agent for WebSphere Application Server from the cluster. You must perform these steps while the cluster is active.

To uninstall the agent

1. Log in as `root`.
2. Set the cluster configuration mode to read/write by typing the following command from any system in the cluster.

   ```
   # haconf -makerw
   ```
3. Remove all WebSphere resources from the cluster. Use the following command to verify that all resources have been removed.

   ```
   # hares -list Type=WebSphere5
   ```
4. Remove the agent type from the cluster configuration by typing the following command from any system in the cluster.

   ```
   # hatype -delete WebSphere5
   ```

   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 typing the following command from any system in the cluster:

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

6 Use the platform's native software management program to remove the agent for WebSphere from each node in the cluster. Execute the following command to uninstall the agent:

Platform	Command
AIX	`# installp -u VRTSvcswas5.rte`
Linux	`# rpm -e VRTSvcswas5`
HP-UX	`# swremove VRTSvcswas5`
Solaris	`# pkgrm VRTSvcswas5`

Chapter 7

Troubleshooting the Veritas Agent for WebSphere Application Server

This chapter covers tips and pointers on using the agent for WebSphere with Veritas high availability products. To resolve issues effectively, follow the steps in the order presented below. You may come across unique issues, but make sure that you follow these steps in the presented order to avoid unnecessary issues.

These troubleshooting tips and pointers are applicable whether working with VCS or with VAD clustering technologies. Wherever applicable, VCS and VAD are referred to as cluster.

Testing WebSphere Application Servers 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 WebSphere Application Server independent of the cluster framework. Refer to the cluster documentation for information about disabling the resource.

You can then restart the WebSphere Application Server outside the cluster framework.

Note: Use the same parameters that the resource attributes define within the cluster framework while restarting the resource outside the framework.

To start a WebSphere Deployment Manager outside the cluster framework

1. Using the user name specified in the User attribute, log into the host on which the WebSphere Deployment Manager application is to run.
2. Use the values specified in the agent attributes to start the WebSphere Deployment Manager.
   For example, assume that the WebSphere Deployment Manager environment is set as follows:

Attribute	Value
ServerType	**DeploymentManager**
ServerName	**dmgr**
WAS_NODE	**was60c1dmsol**
WAS_HOME	**/ibm/was/v60/cell1/depmgr**
ServerProfile	**Dmgr01** **Note:** Specify this attribute for WebSphere version 6.0 only.

3. Go to specified directory:

WebSphere Version	Directory
5.x	`/ibm/was/v60/cell1/depmgr/bin`
6.0	`/ibm/was/v60/cell1/depmgr/profiles/Dmgr0/bin`

4. Using the `startManager.sh` script, start the Deployment Manager:

WebSphere Version	Command
5.x	`/ibm/was/v60/cell1/depmgr/bin/startManager.sh`
6.0	`/ibm/was/v60/cell1/depmgr/profiles/Dmgr01/bin/startManager.sh`

5. Ensure that the Deployment Manager Server starts successfully.

If the Deployment Manager works properly outside the cluster framework, you can attempt to implement the server within the framework.

To start a WebSphere Node Agent outside the cluster framework

1. Using the user name specified in the User attribute, log into the host on which the WebSphere Node Agent application is to run.
2. Use the values specified in the agent attributes to start the WebSphere Node Agent.
   For example, assume that the WebSphere Node Agent environment is set as follows:

Attribute	Value
ServerType	**NodeAgent**
ServerName	**nodeagent**
WAS_NODE	**was60c1n1sol**
WAS_HOME	**/ibm/was/v60/cell1/node1**
ServerProfile	Default

3. Go to specified directory:

WebSphere Version	Directory
5.x	`/ibm/was/v60/cell1/node1/bin`
6.0	`/ibm/was/v60/cell1/node1/profiles/default/bin`

4. Using the `startNode.sh` script, start the Node Agent:

WebSphere Version	Command
5.x	`/ibm/was/v60/cell1/node1/bin/startNode.sh`
6.0	`/ibm/was/v60/cell1/node1/profiles/default/bin/startNode.sh`

5. Ensure that the Node Agent starts successfully.

If the Node Agent works properly outside the cluster framework, you can attempt to implement the server within the framework.

To start a WebSphere Application Server outside the cluster framework

1 Using the user name specified in the User attribute, log into the host on which the WebSphere Application Server is to run.

2 Use the values specified in the agent attributes to start the WebSphere Application Server.
For example, assume that the WebSphere Application Server environment is set as follows:

Attribute	Value
ServerType	**ApplicationServer**
ServerName	**server1**
WAS_NODE	**was60c1n1sol**
WAS_HOME	**/ibm/was/v60/cell1/node1**
ServerProfile	Default

3 Go to specified directory:

WebSphere Version	Directory
5.x	`/ibm/was/v60/cell1/node1/bin`
6.0	`/ibm/was/v60/cell1/node1/profiles/default/bin`

4 Using the `startServer.sh` script, start the Application Server:

WebSphere Version	Command
5.x	`/ibm/was/v60/cell1/node1/bin/startServer.sh server1`
6.0	`/ibm/was/v60/cell1/node1/profiles/default/bin/startServer.sh server1`

5 Ensure that the Application Server starts successfully.

If the Application Server works properly outside the cluster framework, you can attempt to implement the server within the framework.

Using correct software and operating system versions

To ensure that no issues arise due to incorrect software and operating system versions, refer to “Supported software” on page 7 for the correct versions of operating system and software to be installed on the resource systems.

Meeting prerequisites

Before installing the agent for WebSphere, double check that you meet the prerequiste requirements. For example, you must install the ACC library before installing the agent for WebSphere. Refer to “Prerequisites” on page 13 for the list of prerequisites.

Configuring WebSphere resources

Before using a WebSphere resource, ensure that you configure the resource properly. Refer to “Agent attributes” on page 20 for the list of resource types with which you can configure all WebSphere resources.

Be sure that you refer to “Important considerations while configuring the agent” on page 25 for important information abouting setting the agent for WebSphere attributes.

Clustering WebSphere Application Server instances

While attempting to cluster WebSphere Application Server instances, you can follow various methods and procedures to install and cluster the WebSphere Application Server. Refer to “Overview of the clustering process” on page 31 for a clustering process that Symantec recommends.

Using log files

If you are facing problems while using the agent for WebSphere or a WebSphere Application Server instance, refer to the following sections to access the relevant files for information about the issue.

Using agent log files

In case of problems while using the agent for WebSphere, you can access the agent log files for more information. The agent saves output of every entry point process in the temporary folder of the resource system. If the temporary folder is `/tmp`, the log files are saved using the following naming format:

`/tmp/.<Resource_Name>.<Entry_Point>.<Process_ID>`

For example, for a resource *`WAS50DeployMgr_dmgr`*:

`/tmp/.WAS50DeployMgr_dmgr.online.<Process_ID>`
`/tmp/.WAS50DeployMgr_dmgr.offline.<Process_ID>`
`/tmp/.WAS50DeployMgr_dmgr.clean.<Process_ID>`
`/tmp/.WAS50DeployMgr_dmgr.monitor.<Process_ID>`

If a resource, *`WAS50DeployMgr_dmgr`* is unable to bring a WebSphere Node Manager online, you can access the `/tmp/.WAS50DeployMgr_dmgr.online.<Process_ID>` for more information so that you can diagnose the problem.

Caution: These files are overwritten each time you execute the corresponding entry point process. In case of information that you want to save, make a copy of the files at another location.

Using cluster log files

In case of problems while using the agent for WebSphere, you can also access the cluster engine log file for more information about a particular resource.

- The VCS engine log file is `/var/VRTSvcs/log/engine_A.log`.
- The VAD engine log file is `/var/VRTSvad/log/engine_A.log`.

Reviewing WebSphere Application Server log files

If the WebSphere Application Server is facing problems, access the log files of the WebSphere Application Server to further investigate the problem. The log files are located at `<WAS_HOME>/logs/<ServerName>/.*.log`.

Using trace level logging

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

If you set ResLogLevel to **`TRACE`**, a very high volume of messages is produced. Symantec recommends that you must localize the ResLogLevel attribute for particular resource.

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. Test the identified resource. The operation reproduces the problem that you are attempting to diagnose.
5. Set the ResLogLevel attribute back to **INFO** for the identified resource:
   `hares -modify Resource_Name ResLogLevel INFO -sys SysA`
6. Review the contents of the cluster engine output log file to diagnose the problem.

You may also contact Symantec support for more help.

Appendix A

Sample Configurations

This appendix includes sample `main.cf` and `main.xml` configuration files that you can refer to.

Sample configuration in a VCS environment

The following is an excerpt from a VCS configuration file (`main.cf`) that defines a Network Deployment of WebSphere Application Servers and two independent Application Servers (Application Servers that are not part of a Network Deployment).

This configuration demonstrates that you can combine Network Deployment WebSphere Cells with independent WebSphere Application Servers. In the example, there is one WebSphere Cell comprised of one Deployment Manager named `dmgr`. The WebSphere Cell contains two Node Managers, both named `nodeagent`.

For more information, refer to "Service group configuration options" on page 26 to configure a service group that manages one independent, stand-alone Application Server.

```
group WAS51Cell1DM (
        SystemList = { sysa = 0, sysb = 1, sysc = 2 }
        )

        DiskGroup WAS51Cell1DM_dg (
                DiskGroup = was51c1dm
                )

        IP WAS51Cell1DM_ip (
                Device = hme0
                Address = "10.136.228.11"
                NetMask = "255.255.248.0"
                )
```

```
        Mount WAS51Cell1DM_mnt (
                MountPoint = "/WAS51/cell1/depmgr"
                BlockDevice = "/dev/vx/dsk/was51c1dm/was"
                FSType = vxfs
                FsckOpt = "-y"
                )

        WebSphere5 WAS51Cell1DM_was (
                ServerName = dmgr
                WAS_NODE = was51c1dm
                WAS_HOME = "/WAS51/cell1/depmgr"
                User = root
                ServerType = DeploymentManager
                )

        WAS51Cell1DM_was requires WAS51Cell1DM_ip
        WAS51Cell1DM_was requires WAS51Cell1DM_mnt
        WAS51Cell1DM_mnt requires WAS51Cell1DM_dg


group WAS51Cell1Node1 (
        SystemList = { sysb = 0, sysa = 1, sysc = 2 }
        )

        DiskGroup WAS51Cell1Node1_dg (
                DiskGroup = was51c1n1
                )

        IP WAS51Cell1Node1_ip (
                Device = hme0
                Address = "10.136.228.18"
                NetMask = "255.255.248.0"
                )

        Mount WAS51Cell1Node1_mnt (
                MountPoint = "/WAS51/cell1/node1"
                BlockDevice = "/dev/vx/dsk/was51c1n1/was"
                FSType = vxfs
                FsckOpt = "-y"
                )

        WebSphere5 WAS51Cell1Node1_NA_was (
                ServerName = nodeagent
                WAS_NODE = was51c1n1
                WAS_HOME = "/WAS51/cell1/node1"
                User = root
                ServerType = NodeAgent
                )

        WebSphere5 WAS51Cell1Node1_AS1_was (
                ServerName = server1
                WAS_NODE = was51c1n1
```

```
                WAS_HOME = "/WAS51/cell1/node1"
                User = root
                ServerType = ApplicationServer
                )

        WebSphere5 WAS51Cell1Node1_AS2_was (
                ServerName = server2
                WAS_NODE = was51c1n1
                WAS_HOME = "/WAS51/cell1/node1"
                User = root
                ServerType = ApplicationServer
                )

        WAS51Cell1Node1_AS2_was requires WAS51Cell1Node1_NA_was
        WAS51Cell1Node1_mnt requires WAS51Cell1Node1_dg
        WAS51Cell1Node1_NA_was requires WAS51Cell1Node1_ip
        WAS51Cell1Node1_NA_was requires WAS51Cell1Node1_mnt
        WAS51Cell1Node1_AS1_was requires WAS51Cell1Node1_NA_was


group WAS51Cell1Node2 (
        SystemList = { sysa = 0, sysa = 1, sysb = 2 }
        )

        DiskGroup WAS51Cell1Node2_dg (
                DiskGroup = was51c1n2
                )

        IP WAS51Cell1Node2_ip (
                Device = eri0
                Address = "10.136.228.19"
                NetMask = "255.255.248.0"
                )

        Mount WAS51Cell1Node2_mnt (
                MountPoint = "/WAS51/cell1/node2"
                BlockDevice = "/dev/vx/dsk/was51c1n2/was"
                FSType = vxfs
                FsckOpt = "-y"
                )

        WebSphere5 WAS51Cell1Node2_NA_was (
                ServerName = nodeagent
                WAS_NODE = was51c1n2
                WAS_HOME = "/WAS51/cell1/node2"
                User = root
                ServerType = NodeAgent
                )

        WebSphere5 WAS51Cell1Node2_AS1_was (
                ServerName = server1
                WAS_NODE = was51c1n2
```

```
                WAS_HOME = "/WAS51/cell1/node2"
                User = root
                ServerType = ApplicationServer
                )

        WAS51Cell1Node2_mnt requires WAS51Cell1Node2_dg
        WAS51Cell1Node2_NA_was requires WAS51Cell1Node2_ip
        WAS51Cell1Node2_NA_was requires WAS51Cell1Node2_mnt
        WAS51Cell1Node2_AS1_was requires WAS51Cell1Node2_NA_was


group WAS51AppSrvr1 (
        SystemList = { sysa = 0, sysb = 1, sysc = 2 }
        )

        DiskGroup WAS51AppSrvr1_dg (
                DiskGroup = was51appsrvr1
                )

        IP WAS51AppSrvr1_ip (
                Device = hme0
                Address = "10.136.228.21"
                NetMask = "255.255.248.0"
                )

        Mount WAS51AppSrvr1_mnt (
                MountPoint = "/WAS51/srvr1"
                BlockDevice = "/dev/vx/dsk/was51appsrvr1/was"
                FSType = vxfs
                FsckOpt = "-y"
                )

        WebSphere5 WAS51AppSrvr1_AppSrvr_was (
                ServerName = server1
                WAS_NODE = was51srv1
                WAS_HOME = "/WAS51/srvr1"
                User = root
                ServerType = ApplicationServer
                )

        WAS51AppSrvr1_mnt requires WAS51AppSrvr1_dg
        WAS51AppSrvr1_AppSrvr_was requires WAS51AppSrvr1_ip
        WAS51AppSrvr1_AppSrvr_was requires WAS51AppSrvr1_mnt


group WAS51AppSrvr2 (
        SystemList = { sysa = 0, sysb = 1, sysc = 2 }
        )

        DiskGroup WAS51AppSrvr2_dg (
                DiskGroup = was51appsrvr2
                )
```

```
IP WAS51AppSrvr2_ip (
        Device = hme0
        Address = "10.136.228.22"
        NetMask = "255.255.248.0"
        )

Mount WAS51AppSrvr2_mnt (
        MountPoint = "/WAS51/srvr2"
        BlockDevice = "/dev/vx/dsk/was51appsrvr2/was"
        FSType = vxfs
        FsckOpt = "-y"
        )

WebSphere5 WAS51AppSrvr2_AppSrvr_was (
        ServerName = server1
        WAS_NODE = was51srv1
        WAS_HOME = "/WAS51/srvr2"
        User = root
        ServerType = ApplicationServer
        )

WAS51AppSrvr2_mnt requires WAS51AppSrvr2_dg
WAS51AppSrvr2_AppSrvr_was requires WAS51AppSrvr2_ip
WAS51AppSrvr2_AppSrvr_was requires WAS51AppSrvr2_mnt
```

Sample configuration in a VAD environment

A sample `main.xml` file is in the `/etc/VRTSagents/ha/conf/WebSphere5/` directory.

This sample configuration defines a Network Deployment of WebSphere Application Servers and an independent Application Server, which is not part of a Network Deployment.

Index

S

U

V

W