Application Note: Using Agent Builder to Create New Agents

Windows

6.0

March 2012

Symantec

Application Note: Using Agent Builder to Create New Agents

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043
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.

Technical support

Visit http://www.symantec.com/business/support/assistance_care.jsp for product assistance. 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. If you encounter an error when using a product, include the error number preceding the message when contacting Technical Services. You can also use the error number to search for information in TechNotes or documents on the Web site.

Using Agent Builder to create new agents

Introduction

This application note describes the procedure to build an agent using Agent Builder.

This document is meant as a reference for users who want to quickly generate agents with separate agent executables and agent type definitions.

About Agent Builder

The Agent Builder provides a quick and easy way to build a new agent and a new resource type for different applications.

Creating different agents for different applications helps you manage a large number of applications independently. You can define attributes that are specific to an application individually at the resource type level without affecting other applications.

The Agent Builder contains a core agent from which the Agent Builder tool generates code for a new agent type. The core agent is similar in functionality to any other Process or GenericService agent. The agent has well-defined attributes, entry points, and design patterns for clustering enterprise applications.

Key features

The agent created using Agent Builder is fully feature-compatible with the existing Process or GenericService agents. In addition, the agent has the following features:

- Validation of attributes based on entry points
  Facility to validate the attributes in each entry point before the actual data processing begins. Hence, the code is more robust.
- First Failure Data Capture (FFDC)
  In case of fault, the agent generates a huge volume of debug logs which enable troubleshooting of the fault. Hence, the need to reproduce the failure condition is reduced.
- Fast First Level Monitor (FFLM)
  The agent maintains PID files based on search patterns in order to expedite the monitoring process.
- Service Monitoring
  If the service name of the application is provided in the respective attribute, then the service will be monitored apart from the regular monitoring.

- Support for external user-specified monitor utilities
  In addition to the monitoring logic that gets built-in, user-specified monitor utilities can be plugged in. This facility enables the administrators to completely customize the monitoring of their applications that are specific to their setup.
- Delay entry point
  For slow initializing applications, the agent intelligently figures out the best way to delay the first monitor after online.

Supported software

Agent Builder version 6.0.0.0 supports the following software:

Veritas Cluster Server (VCS)	6.0
ACC Library	6.0
Operating Systems	■ Microsoft Windows 2008 (x64 bit)

Agent Builder version 5.1.0.0 supports the following software:

Veritas Cluster Server (VCS)	5.1
ACC Library	5.1
Operating Systems	■ Microsoft Windows 2003 (32 bit) ■ Microsoft Windows 2003 (x64 bit)

Prerequisites for creating an agent using Agent Builder

Ensure that the following prerequisites are met before you create the agent.

- The application for which an agent is developed must lend itself to being controlled by the agent and must be able to operate in a clustered environment. The following criteria describe an application that can successfully operate in a clustered environment:
  - The application should have a well-defined start program.
  - The application should have a well-defined stop program.
  - The application should have at least one well-defined monitoring method. For example, the application's process pattern should be

known, or the application should maintain PID files, or the application should be fully controlled through the service.

- The application that is to be made highly available is successfully installed and configured on the system.
- The Agent Builder tool is installed on the system.
  See "Installing Agent Builder in a VCS environment" on page 10.
- All cluster nodes where the agent will be deployed must contain the following Windows utilities:
  - "%*SYSTEMDRIVE*%\WINDOWS\System32\Wbem\wmic.exe"
  - "%*SYSTEMDRIVE*%\WINDOWS\System32\sc.exe"
- The latest version of the ACC library is installed on the system.
  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.
  See "About the ACC library" on page 8.
- Since Agent Builder depends on a set of Perl modules, ensure that VRTSPerl is installed and is available in your PATH.

About the ACC library

The operations for the agent created using Agent Builder depend on a set of Perl modules known as the ACC library. The ACC Library contains common, reusable functions which perform tasks, such as process identification, logging, and system calls.

The library must be installed on each system in the cluster that will run the agent.

For more information on ACC Library, see the *Veritas™ ACC Library Installation Guide* (Windows).

Installing the ACC library

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

To install the ACC library

1. Log in to any node in the cluster as a user with domain administrative privileges.
2. Open Windows Explorer and change to the following directory:

For Windows 2003 (x32)
***cd1*\windows\w2k3\vcs\application\acc_library*acclibversion*_library\Pkgs**

For Windows 2003 (x64)
***cd1*\windows\w2k3x64\vcs\application\acc_library\5.1\5.1.0.0_library\Acclib_lib.5.1.0.0-GA_w2k3X64\Pkgs**

For Windows 2008 (x64)
cd1\windows\w2k8x64\vcs\application\acc_library\6.0\6.0.0.0_library\Pkgs

3 Double-click **VRTSacclib.msi**.

4 Follow the instructions given in the install program window to complete the installation of the ACC library.

Installing Agent Builder in a VCS environment

Perform the following steps to install Agent Builder.

To install Agent Builder

1. Log in to any node in the cluster as a user with domain administrative privileges.
2. Download the Agent Pack from the Symantec Operations Readiness Tools (SORT) site: https://sort.symantec.com/agents.
   You can download the complete Agent Pack tar file or the individual Agent Builder tar file.
3. Uncompress the file to a temporary location.
4. If you downloaded the complete Agent Pack tar file, navigate to the directory containing the package for the platform running in your environment
   ***cd1*\windows\w2k3\vcs\application\agentbuilder*vcs_version**version*_agent\pkgs**

   Windows 2003 (x64)
   ***cd1*\windows\w2k3x64\vcs\application\agentbuilder*vcs_version**version*_agent\AgentBuilder_agt.*version*-GA_w2k3X64\Pkgs**

   Windows 2008 (x64)
   ***cd1*\windows\w2k8x64\vcs\application\agentbuilder*vcs_version**version*_agent\Pkgs**
5. Double-click **VRTSagentbuilder.msi**.
6. Follow the instructions given in the install program window to complete the installation of Agent Builder.

Building an agent using Agent Builder

You can build an agent in either of the following two ways:

- Building an agent using the CLI
- Building an agent using the GUI

After you finish building the agent, perform the steps given in the section "Post-build steps" on page 12.

Building an agent using the CLI

Perform the following steps to build an agent using the Agent Builder CLI (command line interface).

To build an agent using the Agent Builder CLI

1 Open a command prompt, and change to the `%VCS_HOME%\bin\AgentBuilder\bin` directory:
`cd %VCS_HOME%\bin\AgentBuilder\bin`

2 Run the following command:
`agentBuilder.bat -a agent [-c] [-o outdir] [-A author] [-z <tar|zip|rar>] [-v]`
where:

agent	is the name of the agent.
`-c`	is the console mode
outdir	is the complete location of the output directory in which the agent will be generated. If not specified, the agent will be generated in the `%VCS_HOME%\bin\AgentBuilder\Agents` directory.
author	is the author name in whose name the agent will be generated. If not provided, the current login name will be used as the author name.
`tar|zip|rar`	is the option used for compressing the generated agent file. The supported commands are tar, zip, and rar. To use this option, the archive utility should be installed on the system and its location should be part of the PATH environment variable. If not provided, the agent folder will not be compressed.
`-v`	is the option for getting verbose output.

Note: You can use the **`-v`** option for getting the Agent Builder version.

Agent Builder builds the agent and type as specified in the arguments, and creates the following two directories inside the *`outdir`* directory:

`agent`: contains the agent-specific files
`sample_agent`: contains the agent types file and the `agent.html` file.

Refer to `agent.html` for the agent-specific documentation

3 Perform the steps given in the section "Post-build steps" on page 12.

Building an agent using the GUI

Perform the following steps to build an agent using the Agent Builder GUI (graphical user interface).

Note: Before using the Agent Builder GUI, make sure you install Java 1.4 or later.

To build an agent using the Agent Builder GUI

1. Run the agentBuilder.bat file located in the following directory:
   `%VCS_HOME%\bin\AgentBuilder\bin`
   The Agent Builder Tool wizard displays.
2. Click **Next**.
3. In the Agent Credentials screen, enter the agent details. Only the **Agent Name** field is mandatory; the other fields are optional.
4. Click **Next**.
5. In the Agent Summary screen, review the agent information, and then click **Next**.
   If the agent is created successfully, the Final Output screen displays the message: `Agent [Agent Name] created successfully` and the log file displays in a separate window.
6. Click **Finish**.
7. Perform the steps given in the section "Post-build steps" on page 12.

Post-build steps

After you build an agent using either the CLI or the GUI, perform the following steps.

1. If the environment variable VCS_HOME is set, then Agent Builder will generate `main.cmd` for adding the type. Copy the file `default50Agent.dll` from *%VCS_HOME%*`\bin` directory to the agent directory.
   However, if VCS_HOME is not set, then copy the file `default50Agent.dll` from the `bin` directory of the VCS installation to the agent directory by the name `agent.dll`.
2. Copy the *`outdir`*`/agent` directory to *%VCS_HOME%*`\bin` and *`sample_agent`* directory to *%VCS_HOME%*`\conf` on the systems where you want to deploy the agent.

3 Add the agent type to the cluster either by using `main.cmd` or by using Veritas Cluster Manager.

If the `sample_agent` directory contains `main.cmd`, then double-click it and wait while the type is added and the execution of `main.cmd` completes.

If `main.cmd` is not present in the `sample_agent` directory, then import the type by performing the following steps:

- **a** Start Veritas Cluster Manager and connect to the cluster on which the agent is installed.
- **b** Click **File** > **Import Types**.
- **c** In the Import Types dialog box, select the *`sample_agent`*`\agentTypes.cf` file.
- **d** Click **Import**.
- **e** Save the VCS configuration.

Configuring the agent

After the agent and the agent type are created, you can configure the resources using the newly created agent type.

Before you configure the resources, you must first define the attributes and the agent functions or entry points.

Attribute definitions

The agent created by Agent Builder has the following attributes:

Table 2-1 Agent attributes

Attribute	Definition
CleanProgram String	The complete path to a user-specified utility that is used to forcibly stop the application. Command line arguments are supported. This is an optional attribute. If CleanProgram contains an argument, enclose the CleanProgram between quotes ("") as shown in example 1. If the argument contains a path, then put the path between quotes ("") as shown in example 2. Example 1: `"\"C:\\Program Files\\IBMIHS\\bin\\httpd.exe\" stop"` Example 2: `"\"C:\\Program Files\\IBMIHS\\bin\\httpd.exe\" -f \"C:\\Program Files\\IBMIHS\\conf\\httpd.conf\" -k stop"` Example 3: `"C:\Program Files\IBMIHS\bin\apachectlean.bat"` Default Value: `""`
Domain String	Specifies the Windows domain name to which the specified user belongs. If the attribute value for User does not belong to a Windows domain, use VCS localization settings to specify the local computer name for each system. Example: ABDomain Default Value: `""`

Table 2-1 Agent attributes

Attribute	Definition
EnvFile String	The complete path to the file that the agent for Agent Builder sources to set the environment variables. This is an optional attribute. Example: `C:\\Program Files\\IBM\\WebSphere\\envfile` Default Value: `""`
ListenAddressPort String	A composite string containing the virtual hostname/IPv4 address and the listen port, and delimited by a colon (:). This attribute is used to do a connect during monitoring, to determine if the application is listening to the port on the specified host. This is an optional attribute. Example: `adminsol.veritas.com:9191` Default Value: `""`
MonitorProcessPatterns Vector String	A list of processes to be monitored and cleaned. The process pattern can be a regular expression or a process name with full command line arguments. This is an optional attribute. `Example: C:\\program files\\IBM\\WebSphere\\AppSrv\\profiles\\AppSrv01\\java\\bin\\java` Default Value: `""`

Table 2-1 Agent attributes

Attribute	Definition
MonitorProgram String	Contains the complete path name and command-line arguments for an externally provided monitor program. The monitor entry point executes this program to perform a user-defined state check. Monitor entry point executes the MonitorProgram according to the given MonitorProgramFrequency. This program is not supplied with Agent Builder and is externally developed by the user to satisfy unique requirements. Symantec recommends storing the external monitor utility on the shared disk directory to ensure that the file is always available on the online system. The exit code of the program is interpreted by the monitor entry point as follows: ■ 110 or 0 - Agent Builder is ONLINE. ■ 100 or 1 - Agent Builder is OFFLINE. ■ 99 - Agent Builder state is UNKNOWN. ■ Any other value- The Agent Builder state is UNKNOWN. This is an optional attribute. If MonitorProgram contains an argument, enclose the MonitorProgram between quotes ("") as shown in example 1. If the argument contains a path, then put the path between quotes ("") as shown in example 2. Example 1: `"\"C:\\Program Files\\IBM\\WebSphere\\myMonitor.bat\" -u user1"` Example 2: "\"C:\\Program Files\\IBM\\WebSphere\\bin\\myMonitor.bat\" -f \"C:\\Program files\\conf.cf\"" Example 3: `"C:\\Program Files\\IBM\\WebSphere\\myMonitor.bat"` Default Value: `""`

Table 2-1 Agent attributes

Attribute	Definition
MonitorProgramFrequency Integer	Used to enable second-level or external monitoring and specify how often it is run. During second level monitoring the monitor program will execute the utility provided by MonitorProgram attribute. The numeric value of MonitorProgramFrequency specifies how often the MonitorProgram must run. ■ 0- indicates never run the MonitorProgram. ■ 1- indicates run MonitorProgram every monitor interval. ■ 2- indicates run MonitorProgram every second monitor interval, and so on. This is an optional attribute. Example: 1 Default Value: 0 **Note:** ■ Exercise caution while setting MonitorProgramFrequency to large numbers. For example, if the MonitorInterval is set to 60 seconds and the MonitorProgramFrequency is set to 100, then the MonitorProgram is executed every 100 minutes, which may not be as often as intended. For maximum flexibility, no upper limit is defined for MonitorProgramFrequency. Thus, you can cause the MonitorProgram check to occur once a month, if that is what you desire. ■ If the MonitorSequence attribute contains only MonitorProgram, then the value of the MonitorProgramFrequency attribute must be set to 1.

Table 2-1 Agent attributes

Attribute	Definition
MonitorSequence String	Used to define the sequence in which monitoring methods will be used for monitoring the resource. MonitorSequence is a combination of one or more of the following monitoring methods separated by a space: ■ MonitorProgram ■ ListenAddressPort ■ MonitorProcessPatterns ■ PidFilesPatterns ■ ServiceName Each method corresponds to an attribute. This is a mandatory attribute. Example: ■ ListenAddressPort MonitorProgram ■ PidFilesPatterns ListenAddressPort MonitorProgram ■ MonitorProcessPatterns MonitorProgram ■ ServiceName MonitorProcessPatterns Default Value: "MonitorProcessPatterns ListenAddressPort PidFilesPatterns MonitorProgramServiceName" **Note:** ■ At least one of the attributes specified for MonitorSequence should have a non-null value. ■ If MonitorProgram is the first method defined, ensure that MonitorProgram is available locally on each configured node.
Passwd String	Password for the user. Use the vcsencrypt -agent command to encrypt the password. If you are using the VCS GUI, the GUI automatically encrypts the password. Refer to the VCS documentation for more information about VCSEncrypt. Example: FRGrHRiRJ Default Value: ""

Table 2-1 Agent attributes

Attribute	Definition
PidFilesPatterns Vector String	A list of PID files that contain the process IDs to be monitored and cleaned. These files are application-generated files. Each PID file should contain one PID which will be monitored. Specify the complete path of each PID file in the list. Process pattern can be associated with each PID file, if required. Process pattern can be a regular expression or a process name with full command-line arguments. This is an optional attribute. Example: `C:\\program files\\apache\\server1\\logs\\httpd.pid'',` `httpd.*-f \\IBMIHS\\conf\\httd.conf''` Default Value: "" **Note:** The process ID can change when the process restarts. If the application takes time to update the PID file, then the agent's monitor script may return an incorrect result. If this occurs, increase the ToleranceLimit in the resource definition. The Patterns can contain a regular expression.
ResLogLevel String	Specifies the logging detail performed by the agent for the resource. Valid values are: ■ ERROR - Only logs error messages. ■ WARN - Logs error and warning messages. ■ INFO - Logs error, warning, and informational messages. ■ TRACE - Logs error, warning, informational, and trace messages. This option is very verbose and should only be used during initial configuration or for troubleshooting and diagnostic operations. Example: INFO Default Value: INFO

Table 2-1 Agent attributes

Attribute	Definition
ServiceName String	The service name of the application that has to be started and stopped as a part of the process of starting and stopping the application. If ServiceName contains a space, enclose ServiceName between quotes ("") as shown in example 2. Example 1: Apache2.2 Example 2: “IBMWAS6Service - CellManager01” Default Value: `""`
StartProgram String	The complete path to the user-supplied external utility that is used to start the application. Command line arguments are supported. Either ServiceName or StartProgram should be provided for starting the Agent Builder instance. If StartProgram contains an argument, enclose StartProgram between quotes ("") as shown in example 1. If the argument contains a path, then put the path also between quotes ("") as shown in example 2. Example 1: `"\"C:\\Program Files\\IBMIHS\\bin\\httpd.exe\" start"` Example 2: `"\"C:\\Program Files\\IBMIHS\\bin\\httpd.exe\" -f \"C:\\Program Files\\IBMIHS\\conf\\httpd.conf\" -k start" Files\\IBMIHS\\conf\\httpd.conf\`` Example 3: `"C:\Program Files\IBMIHS\bin\apachestart.bat"` Default Value: `""`

Table 2-1 Agent attributes

Attribute	Definition
StopProgram String	The complete path to the user-supplied external utility that is used to stop the application. Command line arguments are supported. If StopProgram contains an argument, enclose StopProgram between quotes ("") as shown in example 1. If the argument contains a path, then put the path also between quotes ("") as shown in example 2. Example 1: `"\"C:\\Program Files\\IBMIHS\\bin\\httpd.exe\" stop"` Example 2: `"\"C:\\Program Files\\IBMIHS\\bin\\httpd.exe\" -f \"C:\\Program Files\\IBMIHS\\conf\\httpd.conf\" -k stop"` Example 3: `"C:\Program Files\IBMIHS\bin\apacheStop.bat"` Default Value: `""`
User String	The Windows user name used to run StartProgram, StopProgram, MonitorProgram, and CleanProgram. If the MonitorProgram attribute is specified, the agent uses this user's credentials to run the defined program. The user name must be synchronized across the systems within the cluster. The processes specified in the MonitorProcesses and PidFilesPatterns list must run in the context of the specified user. This is an optional attribute. Example: User1 Default Value: Administrator

State definitions

- ONLINE: Indicates the service being monitored is online.
- OFFLINE: Indicates the service being monitored is offline.
- UNKNOWN: Indicates the service operation is in a pending state, or that the agent could not determine the state of the resource.

Agent functions

The agent brings services online, takes them offline, and monitors their status.

Online

The agent performs the following tasks in an online operation:

- Verifies that the required attributes are set correctly.
- Performs the following check to ensure that the resource is fully offline.
  - If defined, checks the status of the service.
  - If defined, checks the MonitorProcessPatterns attribute to see if any process defined in the pattern is running.
  - If defined, checks the PidFilesPatterns attribute to see if any PIDs defined in the attribute are running.
- If the resource is deemed to be fully online, then returns immediately.
- If the resource is deemed to be fully offline, then executes the start routine by calling the user-defined StartProgram or by starting the service, if defined, or by performing both actions if both are defined.
- If the resource is deemed to be partially online, then cleans up the partial online state by killing the processes and calling CleanProgram to clean up the partially online resource.
- After cleaning up a partially online resource, proceeds with the online operation by executing the start routine.

Offline

The agent performs the following tasks in an offline operation:

- Verifies that the required attributes are set correctly.
- Verifies that the Agent Builder server instance is not offline.
- If the instance is already offline, the operation verifies if any processes belonging to this Agent Builder resource exist.
- Attempts to stop the application by stopping the service, if defined, or by executing the StopProgram if defined or performing both actions if both are defined.
- Then, the offline operation kills any existing processes that belong to this Agent Builder server instance if MonitorProcessPatterns or PidFilesPatterns are defined.

Monitor

The agent performs the following tasks in the monitor operation:

- The monitor operation conducts a service-level check to determine whether the service is running or not. If the service-level check does not find the service running on the node, the monitor process will report the instance as offline. The service level monitoring will be done only when the ServiceName attribute is provided and the MonitorSequence attribute value string contains ServiceName.
- Executes the different monitoring methods according to the user-defined MonitorSequence. The different monitoring methods are as follows:
  - PidFilesPatterns
  - MonitorProcessPatterns
  - ListenAddressPort
  - MonitorProgram

  For a description of the monitoring methods, refer to "Attribute definitions" on page 14.
- If any of the above method returns offline, then monitor returns offline.
- If any of the above method returns online, then it continues with the next method in the sequence defined by the user until all the monitoring methods have been executed. If all of the methods return online, only then is the resource considered to be online.
- The agent runs MonitorProgram at a periodic interval determined by MonitorProgramFrequency. The agent runs all other monitoring methods on every monitoring cycle.

Clean

In the clean operation, the agent tries to shut down the Agent Builder resource in a graceful way if the clean reason is not offline timeout or offline ineffective (since in these two cases graceful shutdown has already occurred). After trying the graceful shutdown, the agent kills the remaining Agent Builder processes.

The agent performs the following tasks in the clean operation:

- Calls the user-defined CleanProgram/StopProgram to cleanup any application state.
- If any of the PIDs or patterns defined in the PidFilesPatterns are found, then cleans those.
- If any of the patterns defined in MonitorProcessPatterns are found then clean those.

Resource type definition

The Agent Builder creates a new resource type using the resource type definition.

The resource type definition is as follows.

```
type AgentBuilder (
    static boolean AEPTimeout = 1
    static str ArgList[] = { ResLogLevel, State, IState, Domain,
Passwd, User, EnvFile, StartProgram, StopProgram, CleanProgram,
ServiceName, ListenAddressPort, MonitorProgramFrequency,
MonitorSequence, MonitorProgram ,PidFilesPatterns,
MonitorProcessPatterns }
    str ResLogLevel = INFO
    str Domain
    str Passwd
    str User = Administrator
    str EnvFile
    str StartProgram
    str StopProgram
    str CleanProgram
    str ServiceName
    str ListenAddressPort
    int MonitorProgramFrequency
    str MonitorSequence = "MonitorProcessPatterns PidFilesPatterns
ListenAddressPort MonitorProgram ServiceName"
    str MonitorProgram
    str PidFilesPatterns{}
    str MonitorProcessPatterns[]
   )
```

Sample configurations

The following example demonstrate how to create an agent using Agent Builder along with their sample configurations.

In this example, a new agent AgentApache is created by author Symantec, and it is deployed on two Windows systems using the following procedure.

1. The agent author runs the following command:

   C:\Program Files\Veritas\cluster server\bin\AgentBuilder\bin>agentBuilder.bat -a AgentApache -c -A "Symantec"

   Agent [AgentApache] created successfully!

2. The directory `C:\Program Files\Veritas\cluster server\bin\AgentBuilder\agents\AgentApache_agent_file\AgentApache` is copied to the directory `C:\Program Files\Veritas\cluster server\bin` on both the systems.

3 The directory `C:\Program Files\Veritas\cluster server\bin\AgentBuilder\agents\AgentApache_agent_file\Sample_AgentApache` is copied to the directory `C:\Program Files\Veritas\cluster server\conf` on both the systems.

In the following sample configuration, an Apache instance has been clustered in two different ways by creating two different resources of the ApacheAgent resource. Either of the two configurations can be used for the actual configuration.

Sample configuration 1

```
group AB_grp (
    SystemList = { SYSTEM1 = 0 , SYSTEM2 = 1 }
    )
    AgentApache Apache_res (
        Enabled = 0
        Critical = 0
        Domain = DOMAIN_AP
                Passwd = IUJuKUlOJoKOlOMoN
        ServiceName = "Apache2.2"
        ListenAddressPort = "vcsapche:8000"
        MonitorProgramFrequency = 0
        MonitorSequence = "ListenAddressPort ServiceName"
        )
    AgentApache Apache_res_1 (
        Enabled = 0
        Critical = 0
        Domain = ISV-DOMAIN
        Passwd = bphNepE
        StartProgram =
"\"\\\"D:\\\\rajeev\\\\httpd-2.2.11-win64\\\\httpd-2.2-x64\\\\bin\\
\\httpd.exe\\\" -k start\""
        StopProgram =
"\"\\\"D:\\\\rajeev\\\\httpd-2.2.11-win64\\\\httpd-2.2-x64\\\\bin\\
\\httpd.exe\\\" -k stop\""
        CleanProgram =
"\"\\\"D:\\\\rajeev\\\\httpd-2.2.11-win64\\\\httpd-2.2-x64\\\\bin\\
\\httpd.exe\\\" -k stop\""
        ListenAddressPort @SYSTEM1 = "SYSTEM1:180"
                ListenAddressPort @SYSTEM2 = "SYSTEM2:180"
        MonitorProgram =
"\"D:\\\\rajeev\\\\httpd-2.2.11-win64\\\\httpd-2.2-x64\\\\logs\\\\m
onitor.bat\""
        PidFilesPatterns = {

"\"D:\\\\rajeev\\\\httpd-2.2.11-win64\\\\httpd-2.2-x64\\\\logs\\\\h
ttpd.pid\"" =
"D:\\\\rajeev\\\\httpd-2.2.11-win64\\\\httpd-2.2-x64\\\\bin\\\\http
d.exe" }
        MonitorProcessPatterns = {
```

```
"D:\\\\rajeev\\\\httpd-2.2.11-win64\\\\httpd-2.2-x64\\\\bin\\\\http
d.exe" }
        MonitorSequence = "MonitorProcessPatterns PidFilesPatterns
ListenAddressPort MonitorProgram"
        )
```

Sample configuration 2

```
group AB_GRP_1 (
    SystemList = { SYSTEM1 = 0, SYSTEM2 = 1 }
    )
    Apache test_apache (
        Domain = DOMAIN_AP
        Passwd = IUJuKUlOJoKOlOMoN
        StartProgram = "'\\\"C:\\\\Program Files\\\\Apache Software
Foundation\\\\Apache2.2\\\\bin\\\\httpd.exe\\\" -k start\""
        StopProgram = "'\\\"C:\\\\Program Files\\\\Apache Software
Foundation\\\\Apache2.2\\\\bin\\\\httpd.exe\\\" -k stop\""
        CleanProgram = "'\\\"C:\\\\Program Files\\\\Apache Software
Foundation\\\\Apache2.2\\\\bin\\\\httpd.exe\\\" -k stop\""
        MonitorProcessPatterns = {
             "C:\\\\Program Files\\\\Apache Software
Foundation\\\\Apache2.2\\\\bin\\\\httpd.exe" }
        )

    IP test_ip (
        Address = "212.12.125.93"
        SubNetMask = "255.255.255.0"
        MACAddress @SYSTEM1 = "02-14-32-4D-VB-Z2"
        MACAddress @SYSTEM2 = "02-21-19-93-BV-39"
        )

    NIC test_nic (
        MACAddress @SYSTEM1 = "02-14-32-4D-VB-Z2"
        MACAddress @SYSTEM2 = "02-21-19-93-BV-39"
        )

    test_ip requires test_nic
    test_apache requires test_ip

    // resource dependency tree
    //
    //  group AB_GRP1
    //  {
    //  Apache test_apache
    //      {
    //      IP test_ip
    //          {
    //          NIC test_nic
    //          }
    //      }
    //  }
```

Removing Agent Builder in a VCS environment

You can choose to uninstall Agent Builder after you have finished creating the agents. Perform the following steps to remove Agent Builder.

To remove Agent Builder

1. Log in, as a user with domain administrative privileges, to the node in the cluster from which you want to uninstall Agent Builder.
2. Click **Start** > **Settings** > **Control Panel**. The Control Panel window opens.
3. Double-click **Add/Remove Programs**. The Add or Remove Programs window opens.
4. From the list of programs, select **VRTSagentbuilder**.
5. Click **Change/Remove**.
6. Follow the instructions given in the uninstall program window to complete the uninstallation of Agent Builder.

Removing the agent created by Agent Builder

Perform the following steps to remove the agent created using Agent Builder.

To remove the agent created by Agent Builder

1. Log in to any node in the cluster as a user with domain administrative privileges.
2. Remove all the resources of the resource type to be deleted.
3. Delete the resource type that was created using Agent Builder.
4. Remove the agent directory from all nodes in which the agent is deployed.
5. Go to *%VCS_HOME%*\bin folder and remove the agent folder.