# Application Note: Using AgentBuilder to Create New Agents

AIX, Linux, Solaris

6.2

January 2016

# Application Note: Using AgentBuilder to Create New Agents

Product version: 6.2

Document version: 6.2 Rev 0

## Legal Notice

Veritas Technologies LLC
500 E Middlefield Road
Mountain View, CA 94043

http://www.veritas.com

# Technical Support

Technical Support maintains support centers globally. Technical Support's primary role is to respond to specific queries about product features and functionality. The Technical Support group also creates content for our online Knowledge Base. The Technical Support group works collaboratively with the other functional areas within the company to answer your questions in a timely fashion.

Our support offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and/or Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers software upgrades
- Global support purchased on a regional business hours or 24 hours a day, 7 days a week basis
- Premium service offerings that include Account Management Services

For information about our support offerings, you can visit our website at the following URL:

www.veritas.com/support

All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policy.

## Contacting Technical Support

Customers with a current support agreement may access Technical Support information at the following URL:

www.veritas.com/support

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level
- Hardware information
- Available memory, disk space, and NIC information

- Operating system
- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:
  - Error messages and log files
  - Troubleshooting that was performed before contacting Technical Support
  - Recent software configuration changes and network changes

## Licensing and registration

If your product requires registration or a license key, access our technical support Web page at the following URL:

www.veritas.com/support

## Customer service

Customer service information is available at the following URL:

www.veritas.com/support

Customer Service is available to assist with non-technical questions, such as the following types of issues:

- Questions regarding product licensing or serialization
- Product registration updates, such as address or name changes
- General product information (features, language availability, local dealers)
- Latest information about product updates and upgrades
- Information about upgrade assurance and support contracts
- Advice about technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs, DVDs, or manuals

## Support agreement resources

If you want to contact us regarding an existing support agreement, please contact the support agreement administration team for your region as follows:

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

# Using AgentBuilder to Create New Agents

This document includes the following topics:

# Introduction

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

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

# About AgentBuilder

AgentBuilder 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.

AgentBuilder contains a core agent from which the agentbuilder utility generates code for a new agent type. The core agent is similar in functionality to any other application agent. The agent has well-defined attributes, entry points, and design patterns for clustering enterprise applications.

## Key features

The agent created using AgentBuilder is fully feature compatible with the existing application agent. In addition, the agent has the following enhanced features:

- Based on ACCLib and portable across Cluster Server and Veritas Cluster Server One (VCS One) on UNIX platforms.
- Regular expression matching for process patterns.
- Configurable monitoring sequence.
- First Failure Data Capture for faster troubleshooting and Root Cause Analysis (RCA).
- Support for Solaris zones.
- Can be used to cluster any application that has well-defined start and stop programs. Due to its enhanced and flexible monitoring sequence, the agent can be used for applications which do not have a fixed command line but a complex command pattern.

# Prerequisites for creating an agent using AgentBuilder

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.

    Example: The application's process pattern should be known or the application should maintain PID files.
- The application that is to be made highly available is successfully installed and configured on the system.
- The AgentBuilder tool is installed on the system.
- 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.

# About ACC library

The operations for the agent created using AgentBuilder depend on a set of Perl modules known as the ACC library. The ACC Library contains common, reusable functions that 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 *Cluster Server ACC Library Installation Guide*.

# Installing 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. Download ACC library.

   You can download either the complete Agent Pack tarball or the individual ACCLib tarball from the Veritas Services and Operations Readiness Tools (SORT) site (https://sort.veritas.com/agents).

2. Uncompress the file to a temporary location, say /tmp.

3. If you downloaded the complete Agent Pack tarball, navigate to the following pkgs directory (for AIX and Solaris), or rpms directory (for Linux).

   | | |
   |---|---|
   | AIX | `cd1/aix/vcs/application/acc_library/version_library/pkgs` |
   | Linux | `cd1/linux/generic/vcs/application/acc_library/version_library/rpms` |
   | Solaris | `cd1/solaris/dist_arch/vcs/application/acc_library/version_library/pkgs` |

   where dist_arch is sol_sparc.

4. If you downloaded the individual ACCLib tarball, 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.

   | | |
   |---|---|
   | AIX | `# installp -ac -d VRTSacclib.bff VRTSacclib` |
   | Linux | `# rpm -i\`<br>`VRTSacclib-VersionNumber-GA_GENERIC.noarch.rpm` |
   | Solaris | `# pkgadd -d VRTSacclib.pkg` |

## Installing the ACC library IPS package on Oracle Solaris 11 systems

**To install the ACC library IPS package on an Oracle Solaris 11 system**

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 Symantec
   ```

4. Install the package.

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

5. Remove the publisher from the system.

   ```
   # pkg unset-publisher Symantec
   ```

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 service `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 VRTSacclib.p5p package from the pkgs directory to the non-global zone (for example at `/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 Symantec
   ```

7. Install the package.

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

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

   ```
   # pkg unset-publisher Symantec
   ```

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 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 AgentBuilder in a VCS environment

Perform the following steps to install AgentBuilder.

**To install AgentBuilder**

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 tarball or an individual agent tarball.

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

| | |
|---|---|
| AIX | cd1/aix/vcs/application/agentbuilder/vcs_*version*/*version*_agent/pkgs |
| Linux | cd1/linux/generic/vcs/application/agentbuilder/vcs_*version*/*version*_agent/rpms |
| Solaris | cd1/solaris/dist_arch/vcs/application/agentbuilder/vcs_*version*/*version*_agent/pkgs |

   where, dist_arch is sol_sparc.

4. If you downloaded the individual AgentBuilder tarball, navigate to the pkgs directory (for AIX and Solaris), or rpms directory (Linux).
5. Install the package:

| | |
|---|---|
| AIX | `# installp -ac -d VRTSappab.rte.bff VRTSappab.rte` |
| Linux | `# rpm -ihv\`<br>`VRTSappab-AgentVersion-GA_GENERIC.noarch.rpm` |
| Solaris | `# pkgadd -d . VRTSappab` |

# 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 VRTSappab.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/VRTSappab.p5p Symantec
   ```
4. Install the package

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

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

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

# Installing agent packages on Solaris brand non-global zones

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

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

1. Ensure that the SMF service `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 VRTSappab.p5p package from the `pkgs` directory to the non-global zone (for example at `/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/VRTSappab.p5p Symantec
   ```

7. Install the package.

   ```
   # pkg install --accept VRTSappab
   ```

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

   ```
   # pkg unset-publisher Symantec
   ```

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 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 AgentBuilder in a VCS One environment

You must install AgentBuilder on the Policy Master node in VCS One. Using the –system option of the agentbuilder utility, you can deploy the newly created agents on other systems.

**To install AgentBuilder in a VCS One environment**

1. Download the complete Agent Pack tarball from FileConnect site, on the Policy Master system:

   https://fileconnect.symantec.com/

   Alternatively,

   Download the individual agent tarball from the Veritas Services and Operations Readiness Tools (SORT) site:

   https://sort.veritas.com/home

2. Uncompress the file to a temporary location, say `/tmp`.

3. If you downloaded the complete Agent Pack tarball, navigate to the following directory containing the installer for the VCS One agents, for the platform running in your environment:

| | |
|---|---|
| AIX | `# cd aix/vcsone/vcsone_version` |
| Linux | `# cd linux/dist_arch/ vcsone/vcsone_version`<br>Where dist is the Linux distribution and arch is the architecture. |
| Solaris | `# cd solaris/dist_arch/ vcsone/vcsone_version`<br>where dist_arch is sol_sparc. |

4. Enter the command to start the AgentBuilder installation.

```
# ./installagpack -appab
```

# Building an agent using AgentBuilder

**To build an agent using AgentBuilder:**

1. Run the following commands.

| | |
|---|---|
| VCS 4.x | `# cd/opt/VRTSvcs/bin/AgentBuilder` |
| VCS 5.0 and later, VCS One 2.0, 5.0 | `# cd /opt/VRTSagents/ha/bin/AgentBuilder` |

```
./agentbuilder <agent> [-help | -h] [-base <base_product>] \
[-platform <platform>] [-ssh|-rsh] [-verbose] [-zones]\ [-system
System1] [-system System2] [...] [-system SystemN] [-upgrade]
```

where,

- base_product option specifies the base product, which can take a value from the following: {vcs4, vcs5}. Use the vcs4 vlaue for VCS 4.0 and use the value vcs5 for VCS 5.0. By default, the tool generates the agent for VCS 5.1 and later.
- platform specifies the supported platforms, which can take a value from the following: {solaris | solaris/sparc | solaris/x86 | linux | linux/x86 | aix | aix/rs6000}. By default, the agent will be generated for the platform it is run on. For other platforms, specify this option explicitly.
- ssh is the secure shell to deploy the agent to remote systems. By default, the agent is generated on the platform the agent is run from. For other platforms, specify this option explicitly.

  system option specifies the nodes on which you want to deploy the agent.
- rsh is the remote shell to deploy the agent to remote systems.
- `-verbose` displays verbose information
- zones option, if specified, enables Solaris zones support for VCS 4.0/4.1. Zone support is available by default for VCS One and VCS 5.0 and later. Deploys new agent in all zones on the specified systems.
- system option specifies the nodes in the cluster on which you want to deploy the agent.
- `-upgrade` option allows the user to generate a new version of an existing VCS agent that was created using an earlier version of AgentBuilder.

2. The AgentBuilder builds the agent and type as specified in the arguments and copies it to all the nodes as defined by the argument.

   For VCS 4.1, the AgentBuilder deploys the new agent in the directory `/opt/VRTSvcs/bin` on the remote system specified by `-system <system name>`.

   For VCS 5.0 and later, VCS One 2.0, and VCS One 5.0, the AgentBuilder deploys the new agent in the directory `/opt/VRTSagents/ha/bin` on the remote system specified by `-system <system name>`.

---

**Note:** For zone support, the newly created agent must be deployed to the local zone where the application is hosted as well as on all the nodes on which the zone can fail over.

---

For examples of building an agent using AgentBuilder: See “Sample Configurations” on page 27.

# 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 AgentBuilder has the following attributes:

**Table 1-1** Agent attributes

| Attribute | Description |
|---|---|
| CleanProgram<br>String | 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.<br>Example: `/IBMIHS/bin/myclean.sh`<br>Default: No default value |
| EnvFile<br>String | Complete path to the file that the agent for AgentBuilder sources to set the environment variables. This is an optional attribute.<br>Example: `/IBMIHS/bin/setEnv.sh`<br>Default: No default value |
| ListenAddressPort<br>String | A composite string containing the virtual hostname/IPv4 address and listen port delimited by a colon (":"). This 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.<br>Example: `adminsol.veritas.com:9191`<br>Default: No default value |

**Table 1-1** Agent attributes *(continued)*

| Attribute | Description |
|---|---|
| MonitorProcessPatterns<br>Vector String | A list of processes to be monitored and cleaned. Process pattern can be a regular expression or a process name with full command-line arguments displayed by the ps command. This is an optional attribute.<br>Example: `{"/IBMIHS/bin/httpd.*-f.*/IBMIHS/conf/httd.conf"}`<br>Default: No default value |
| MonitorProgram<br>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. Based on the UNIX user defined in the User attribute, this MonitorProgram runs in this user-defined shell. Monitor entry point executes the MonitorProgram according to the given MonitorProgramFrequency.<br>This program is not supplied with the AgentBuilder and is externally developed by the user to satisfy unique requirements.<br>The exit code of the program is interpreted by the monitor entry point as follows:<br>■ 110 or 0 - AgentBuilder is ONLINE.<br>■ 100 or 1 - AgentBuilder is OFFLINE.<br>■ 99 - AgentBuilder state is UNKNOWN.<br>■ Any other value- AgentBuilder state is UNKNOWN.<br>This is an optional attribute.<br>Example 1: `/ibm/myMonitor.sh`<br>Example 2: `/ibm/myMonitor.sh arg1 arg2`<br>Default: No default value |

**Table 1-1** Agent attributes *(continued)*

| Attribute | Description |
|---|---|
| MonitorProgramFrequency<br>Integer | Used to enable MonitorProgram. The numeric value of MonitorProgramFrequency specifies how often the MonitorProgram must run.<br>■ 0- indicates never run the MonitorProgram.<br>1- indicates run MonitorProgram every monitor interval.<br>2- indicates run MonitorProgram every second monitor interval, and so on.<br>This is an optional attribute.<br>Example: 1<br>Default: 0<br>**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.<br>If MonitorSequence attribute contains only MonitorProgram, then the value of MonitorProgramFrequency attribute must be set to 1. |

**Table 1-1** Agent attributes *(continued)*

| Attribute | Description |
|---|---|
| MonitorSequence<br>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:<br>■ MonitorProgram<br>■ ListenAddressPort<br>■ MonitorProcessPatterns<br>■ PidFilesPatterns<br>Each method corresponds to an attribute. This is a mandatory attribute.<br>Example:<br>■ ListenAddressPort MonitorProgram<br>■ PidFilesPatterns ListenAddressPort MonitorProgram<br>■ MonitorProcessPatterns MonitorProgram<br>Default: "MonitorProcessPatterns PidFilesPatterns ListenAddressPort MonitorProgram"<br>**Note:** At least one of the attributes specified for MonitorSequence should have a non-null value.<br>If MonitorProgram is the first method defined, ensure that MonitorProgram is available locally on each configured node. |
| PidFilesPatterns<br>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 displayed by the ps command. This is an optional attribute.<br>`{"/var/logs/httpd.pid","httpd.*-f /IBMIHS/conf/httd.conf"}` |

**Table 1-1** Agent attributes *(continued)*

| Attribute | Description |
|---|---|
| ResLogLevel<br>String | Specifies the logging detail performed by the agent for the resource. Valid values are:<br>■ ERROR - Only logs error messages.<br>■ WARN - Logs above plus warning messages.<br>■ INFO - Logs above plus informational messages.<br>■ TRACE - Logs above plus trace messages. This is very verbose and should only be used during initial configuration or for troubleshooting and diagnostic operations.<br>Example: INFO<br>Default: INFO<br>**Note:** The use of the ResLogLevel attribute is deprecated from VCS version 6.2 onwards. 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. The agent captures the first failure data of the unexpected events and automatically logs debug messages in their respective agent log files. |
| StartProgram<br>String | Complete path to the user-supplied external utility that is used to start up the application. Command-line arguments are supported. This is a mandatory attribute.<br>Example: `/IBMIHS/bin/apachectl start`<br>Default: No default value. |
| StopProgram<br>String | Complete path to the user-supplied external utility that is used to stop the application. Command-line arguments are supported. This is a mandatory attribute.<br>Example: `/IBMIHS/bin/apachectl stop`<br>Default: No default value. |

**Table 1-1** Agent attributes *(continued)*

| Attribute | Description |
|---|---|
| User<br>String | UNIX user name used to run the 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. In other words, the user name must resolve to the same UID and have the same default shell on each system in the cluster. The agent entry points use the getpwnam function system call to obtain UNIX user attributes. As a result, the user can be defined locally or can be defined in a common repository (for example, NIS, NIS+, or LDAP). The processes specified in the MonitorProcesses and PidFilesPatterns list must run in the context of the specified user.<br>This is an optional attribute.<br>Example: daemon Default<br>Value: "root" |

The following resource type level attributes are useful while configuring the agent created using AgentBuilder.

**Table 1-2** Resource type level 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.<br>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 AgentBuilder resources in the cluster. If debug logs must be enabled for a specific AgentBuilder resource, override the LogDbg attribute.<br>Type and dimension: string-keylist<br>Default: {} (none)<br>For more information on how to use the LogDbg attribute, refer to the *Cluster Server Administrator's Guide*. |

**Table 1-2** Resource type level attributes *(continued)*

| Attribute | Description |
|---|---|
| OnlineWaitLimit | Number of monitor intervals to wait after completing the online procedure, and before the resource becomes online.<br>Use this attribute to allow sufficient time for slow initializing applications to start.<br>Default: 2 |
| OfflineWaitLimit | Number of monitor intervals to wait after completing the offline procedure, and before the resource becomes offline.<br>This attribute is available only for VCS versions 5.0 and later.<br>Use this attribute to allow sufficient time for applications that take a longer time to stop.<br>Default: 0 |

# 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 (entry points)

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

## Online

The agent performs the following tasks in an online operation:

- Performs the following check to ensure that the resource is fully offline.
  - If defined, checks the MonitorProcessPatterns 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.

- If the resource is deemed to be partially online, then cleans up the partial online state by killing the processes and calling CleanProgram to perform cleanup of the partially online resource.
- After cleaning up a partially online resource proceeds with the online by executing the start routine.

## Offline

Executes the stop routine by calling the user-defined StopProgram.

## Monitor

The agent performs the following tasks in the monitor operation:

- Executes the different monitoring methods according to the user-defined MonitorSequence. The different monitoring methods are as follows:
  - PidFilesPatterns
  - MonitorProcessPatterns
  - ListenAddressPort
  - MonitorProgram

  For more information about the monitoring methods: See “Attribute definitions” on page 17.
- If any of the above method returns offline, then monitor returns offline.
- If any of the above method returns online, 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 the MonitorProgram at a periodic interval determined by the MonitorProgramFrequency. The agent runs all other monitoring methods on every monitoring cycle.

## Clean

The agent performs the following tasks in the clean operation:

- Calls the user-defined CleanProgram to cleanup any application state, such as stale ipc resources.
- If any of the PIDs or patterns defined in the PidFilesPatterns are found, then clean those.
- If any of the patterns defined in MonitorProcessPatterns are found then clean those.

# Resource type definition

AgentBuilder creates new resource type using the resource type definition, which depends on the base product option.

## VCS 5.1 and later

The resource type definition specific to VCS 5.1 and later is as follows.

```
type AgentBuilder (
static boolean AEPTimeout = 1
static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
static str AgentDirectory = "/opt/VRTSagents/ha/bin/AgentBuilder"
static str ArgList[] = { ResLogLevel, State, IState,
StartProgram, StopProgram, CleanProgram,
MonitorProgramFrequency, MonitorProgram,
User, EnvFile, MonitorSequence,
ListenAddressPort, PidFilesPatterns,
MonitorProcessPatterns }
str ResLogLevel = "INFO"
str StopProgram
str MonitorProgram
str CleanProgram
str User = "root"
str EnvFile
str ListenAddressPort
str StartProgram
str MonitorProcessPatterns[]
int MonitorProgramFrequency = 0
str PidFilesPatterns{}
str MonitorSequence = "MonitorProcessPatterns PidFilesPatterns
ListenAddressPort MonitorProgram"
)
```

## VCS 5.0

The resource type definition specific to VCS 5.0 is as follows.

```
type AgentBuilder (
static boolean AEPTimeout = 1
static str ContainerType = Zone
str ContainerName
static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
static str AgentDirectory =
"/opt/VRTSagents/ha/bin/AgentBuilder"
static str ArgList[] = { ResLogLevel, State, IState,
StartProgram, StopProgram, CleanProgram, MonitorProgramFrequency,
MonitorProgram, User, EnvFile, MonitorSequence, ListenAddressPort,
PidFilesPatterns, MonitorProcessPatterns }
str ResLogLevel = "INFO"
str StopProgram
str MonitorProgram
str CleanProgram
str User = "root"
str EnvFile
str ListenAddressPort
str StartProgram
str MonitorProcessPatterns[]
int MonitorProgramFrequency = 0
str PidFilesPatterns{}
str MonitorSequence = "MonitorProcessPatterns PidFilesPatterns
ListenAddressPort MonitorProgram"
)
```

## VCS 4.1

The resource type definition specific to VCS 4.1 is as follows.

```
type AgentBuilder (
static str ArgList[] = { ResLogLevel, State, IState,
StartProgram, StopProgram, CleanProgram, MonitorProgramFrequency,
MonitorProgram, User, EnvFile, MonitorSequence, ListenAddressPort,
PidFilesPatterns, MonitorProcessPatterns }
str ResLogLevel = "INFO"
str StopProgram
str MonitorProgram
str CleanProgram
str User = "root"
str EnvFile
str ListenAddressPort
str StartProgram
str MonitorProcessPatterns[]
int MonitorProgramFrequency = 0
str PidFilesPatterns{}
str MonitorSequence = "MonitorProcessPatterns PidFilesPatterns
ListenAddressPort MonitorProgram"
)
```

# Sample Configurations

Following are examples of creating agents using AgentBuilder and their sample configurations.

Example 1

In this example, a new agent ABSendmail is created for base product VCS 4.x and it is deployed on system Linux-1. # ./agentbuilder ABSendmail -base vcs4 -platform linux -system Linux-1.

```
# ./agentbuilder ABSendmail -base vcs4 -platform linux -system
Linux-1
Agent [ABSendmail] created successfully!
Using [ssh] as default to deploy agent on [Linux-1]
root@Linux-1's password:
Successfully deployed agent on system : [Linux-1]
```

In the following sample configuration, you configure the executable sendmail as StartProgram, StopProgram, and MonitorProgram, with start, stop, and status specified as command line arguments respectively.

Configure the agent for monitoring using following methods: a process specified by sendmail in MonitorProcessPatterns attribute, and PID stored in the PidFilesPatterns attribute and MonitorProgram. As no user is specified, the agent uses the root user.

```
ABSendmail sendmail_res (
 StartProgram = "/etc/init.d/sendmail start"
 StopProgram = "/etc/init.d/sendmail stop"
 MonitorProgram = "/etc/init.d/sendmail status"
 MonitorProcessPatterns = { sendmail }
 MonitorProgramFrequency = 1
 PidFilesPatterns = { "/var/run/sendmail.pid" = "" }
 ).
```

Example 2

In this example, a new agent ABSamba is created for base product VCS 5.0 and it is deployed on systems HP-1 and HP-2.

```
# ./agentbuilder ABSamba -base vcs5 -platform hpux -system HP-1
-system HP-2
Agent [ABSamba] created successfully!
Using [ssh] as default to deploy agent on [HP-1]
root@HP-1's password:
Successfully deployed agent on system : [HP-1]
```

```
Using [ssh] as default to deploy agent on [HP-2]
root@HP-2's password:
Successfully deployed agent on system : [HP-2]
```

In the following sample configuration, you configure the executable samba as StartProgram, StopProgram, and CleanProgram, with start, stop, and force stop specified as command-line arguments respectively.

Configure the agent for monitoring using following methods:

- processes having pattern "smbd" and "nmbd" in MonitorProcessPatterns attribute
- PID stored in “smbd.pid” file and corresponding pattern for the process “smbd” specified in PidFilesPatterns attribute
- An executable, sambaMonitor, monitors the application and uses “all” as its command line argument. Monitor program executes according to MonitorProgramFrequency specified, which is 2 monitor cycles in this example.

```
ABSamba samba_app (
 StartProgram = "/usr/sbin/samba start"
 StopProgram = "/usr/sbin/samba stop"
 CleanProgram = "/usr/sbin/samba force stop"
 MonitorProgram = "/usr/local/bin/sambaMonitor all"
 MonitorProcessPatterns = { "smbd", "nmbd" }
 MonitorProgramFrequency = 2
 PidFilesPatterns = { "/var/lock/samba/smbd.pid" =
 "smbd" }
 )
```

### Example 3

In this example, a new agent ABApache_zones is created for the base product VCS 5.1 and it is deployed on system node-1, node-2 and shared-zone. The new agent supports Solaris zones. As the verbose option is used, detailed messages are displayed while creating and deploying the agent.

Note that the newly created agent must be deployed to the local zone where the application is hosted as well as on all the nodes where the zone can fail over.

Here the shared-zone can fail over between two systems node-1 and node-2. Using –system option the agent ABApache_zones is deployed to node-1 and node-2.

Using the -zones option, AgentBuilder deploys the agent ABApache_zones to shared-zone on both the nodes.

```
# ./agentbuilder ABApache_zones -platform solaris -ssh
-zones -verbose -system node-1 -system node-2 -system
Success: Setting permissions for Agent directory.
```

```
Success: Creating Agent directory structure
Success: Setting permissions for Agent directory.
Success: Generating Agent module
Success: Setting permissions for Agent module
Success: Generating Agent entry point
Success: Setting permissions for Agent entry point
Success: Generating Agent GUI xml
Success: Setting permissions for Agent GUI xml
Generated online, offline, clean entry points
Opening cluster..
Adding resource type [ABApache_zones]
Closing cluster..
Agent [ABApache_zones] created successfully!
Deploying agent on all nodes::
Using [ssh] to deploy agent on [node-1]
Deploying the agent in ZonePath = /zones/shared-zone
Successfully deployed agent on system : [node-1]
Using [ssh] to deploy agent on [node-2]
Password:
Deploying the agent in ZonePath = /zones/shared-zone
Successfully deployed agent on system : [node-2]
```

In the following sample configuration, you configure the executable httpd as StartProgram and StopProgram, with start and stop specified as command line arguments respectively. Configure the agent for monitoring using following methods:

- processes having pattern "/apache/v2.2/bin/httpd" in MonitorProcessPatterns attribute
- a socket listening at host (isv2) and port (9191) specified as part of ListenAddressPort attribute.

In this example, the default monitoring sequence is changed using MonitorSequence attribute. First, a process check is done using MonitorProcessPatterns attribute and then socket connection test is done using ListenAddressPort attribute.

```
ABApache_zones abapache_res (
 StartProgram = "/apache/v2.2/bin/httpd -k start -f
 /apache/v2.2/conf/httpd.conf"
 StopProgram = "/apache/v2.2/bin/httpd -k stop -f
 /apache/v2.2/conf/httpd.conf"
 ListenAddressPort = "isv2:9191"
 MonitorProcessPatterns = { "/apache/v2.2/bin/httpd" }
 MonitorSequence = "MonitorProcessPatterns
 ListenAddressPort PidFilesPatterns MonitorProgram"
 )
```

Example 4

In this example, a new agent ABApache_vcsone is created for base product VCS One and it is deployed on system sun-1.

```
# ./agentbuilder ABApache_vcsone -base vcsone -platform solaris
-system sun-1
Agent [ABApache_vcsone] created successfully!
Using [ssh] as default to deploy agent on [sun-1]
root@sun-1's password:
Successfully deployed agent on system : [sun-1]
```

Example 5

In this example, an agent is created for WebSphereMessageBroker using the AgentBuilder utility, which is further upgraded using the -upgrade option. The upgrade option generates a new version of the existing agent that was created using an earlier version of the AgentBuilder utility.

```
[root@SystemA AgentBuilder]# ./agentbuilder WebSphereMessageBroker
-platform linux -ssh -verbose -system vcslx194
Success: Setting permissions for Agent directory.
Success: Creating Agent directory structure
Success: Setting permissions for Agent directory.
Success: Generating Agent module
Success: Setting permissions for Agent module
Success: Generating Agent entry point
Success: Setting permissions for Agent entry point
Success: Generating Agent GUI xml
Success: Setting permissions for Agent GUI xml
Generated online, offline, clean entry points
Opening cluster..
Adding resource type [WebSphereMessageBroker]
Closing cluster..
Agent [WebSphereMessageBroker] created successfully!
Deploying agent on all nodes::
Using [ssh] to deploy agent on [vcslx194]
Successfully deployed agent on system : [vcslx194]
```

The following example shows how the upgrade flag is used to upgrade an earlier created WebSphereMessageBroker using the -upgrade flag when the AgentBuilder is updated with the additional attributes.

```
[root@SystemA AgentBuilder]# ./agentbuilder WebSphereMessageBroker
-platform linux -ssh -verbose -upgrade -system vcslx194
Success: Setting permissions for Agent directory.
Success: Creating Agent directory structure
Success: Setting permissions for Agent directory.
Success: Generating Agent module
Success: Setting permissions for Agent module
Success: Generating Agent entry point
Success: Setting permissions for Agent entry point
Success: Generating Agent GUI xml
Success: Setting permissions for Agent GUI xml
Generated online, offline, clean entry points
Opening cluster..
Adding resource type [WebSphereMessageBroker]
New attributes will be added to the [WebSphereMessageBroker] agent
[XYZ]
Executing Command:[/opt/VRTSvcs/bin/hatype -modify WebSphereMessageBroker
ArgList ResLogLevel State IState StartProgram StopProgram CleanProgram
MonitorProgramFrequency MonitorProgram User EnvFile MonitorSequence
ListenAddressPort PidFilesPatterns MonitorProcessPatterns XYZ]
Executing Command:[/opt/VRTSvcs/bin/haattr -add WebSphereMessageBroker
XYZ -string]
Updating [AEPTimeout]
Updating [ContainerOpts]
Executing Command:[/opt/VRTSvcs/bin/hatype -modify
WebSphereMessageBroker AEPTimeout 1]
Executing Command:[/opt/VRTSvcs/bin/hatype -modify
WebSphereMessageBroker ContainerOpts RunInContainer 1 PassCInfo 0]
Closing cluster..
Agent [WebSphereMessageBroker] created successfully!
Deploying agent on all nodes::
Using [ssh] to deploy agent on [vcslx194]
Successfully deployed agent on system : [vcslx194]
```

### Example 6

In this example, the SampleAgent is generated using the AgentBuilder utility. This agent is generated for the platform it is being run from for VCS 5.1 and later. In the following example, the agent is generated for Linux.

```
[root@SystemA AgentBuilder]# ./agentbuilder SampleAgent
Agent [SampleAgent] created successfully!"
```

# Removing AgentBuilder in a VCS environment

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

**To remove AgentBuilder**

1. Log in as a superuser.
2. Execute the following command:

| Platform | Command |
|---|---|
| AIX | `# installp -u VRTSappab.rte` |
| Linux | `# rpm -e VRTSappab` |
| Solaris | `# pkgrm VRTSappab`<br>**Note:** To uninstall the agent IPS package on a Solaris 11 system:<br>`# pkg uninstall VRTSappab` |

# Removing AgentBuilder in a VCS One environment

You can remove AgentBuilder using the installagpack program.

**To remove AgentBuilder**

1. Access the temporary location where you downloaded the Agent Pack and navigate to the directory containing the package for the platform running in your environment:

| | |
|---|---|
| AIX | `# cd aix/vcsone/vcsone_version` |
| Linux | `# cd linux/dist_arch/vcsone/\ vcsone_version`<br>Where dist is the Linux distribution and arch is the architecture. |
| Solaris | `# cd solaris/dist_arch/vcsone/vcsone_version`<br>where dist_arch is sol_sparc. |

2. Start the uninstallagpack program.

```
# ./uninstallagpack -appab
```

3. Enter the name of the Policy Master system.
4. Review the output as the program verifies the agent pack that you installed and removes the AgentBuilder package. You can view logs in the `/var/VRTS/install/logs` directory.

# Removing the agent created by AgentBuilder

Perform the following steps to remove the agent created using AgentBuilder.

**To remove the agent created by AgentBuilder**

1. Log in as superuser.
2. Remove all the resources of the resource type to be deleted.
3. Delete the resource type that was created using AgentBuilder.
4. Remove the agent directory from all nodes in which the agent is deployed.

| | |
|---|---|
| VCS 4.1 | `# /opt/VRTSvcs/bin/agent directory` |
| VCS 5.0 and later<br>VCS One 2.0 | `# /opt/VRTSagents/ha/bin/agent directory` |

# 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.

| | |
|---|---|
| AIX | `# installp -u VRTSacclib` |
| Linux | `# rpm -e VRTSacclib` |
| Solaris | `# pkgrm VRTSacclib`<br>**Note:** To uninstall the ACClib IPS package on a Solaris 11 system:<br>`# pkg uninstall VRTSacclib` |

# Known Issues

### For VCS version 4.x on AIX and HP-UX, the agent created by AgentBuilder cannot be deployed to the specified system.

Workarounds

The following two workarounds are available:

- Option 1: Upgrade Perl to version 5.8.2 or later.
- Option 2: Though the newly created agent does not get deployed to the specified system, the agentbuilder utility creates the agentname.tar file. To deploy the newly created agent on each node:
  - Copy the `agentname.tar` file to the / directory.
  - Extract the `agentname.tar` file to get the agent files.

### Warning displayed while creating an agent using AgentBuilder on Linux

While creating a new agent using AgentBuilder with base product as vcs51, the following warning is displayed.

```
VCS WARNING V-16-1-11140 Operation not allowed on ContainerInfo
```

You can safely ignore this warning.