Veritas™ Cluster Server Agent Developer's Guide

Linux

5.0 Maintenance Pack 3

Veritas Cluster Server Agent Developer's Guide

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Product version: 5.0 MP3

Documentation version: 5.0 MP3.0

Legal Notice

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

Technical Support

Symantec 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 Symantec to answer your questions in a timely fashion. For example, the Technical Support group works with Product Engineering and Symantec Security Response to provide alerting services and virus definition updates.

Symantec's maintenance 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 Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers automatic software upgrade protection
- Global support that is available 24 hours a day, 7 days a week
- Advanced features, including Account Management Services

For information about Symantec's Maintenance Programs, you can visit our Web site at the following URL:

www.symantec.com/techsupp

Contacting Technical Support

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

www.symantec.com/techsupp/

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 Symantec
- Recent software configuration changes and network changes

Licensing and registration

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

www.symantec.com/techsupp

Customer service

Customer service information is available at the following URL:

www.symantec.com/techsupp

Customer Service is available to assist with 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 maintenance contracts
- Information about the Symantec Buying Programs
- Advice about Symantec's technical support options
- Nontechnical presales questions
- Issues that are related to CD-ROMs or manuals

Documentation feedback

Your feedback on product documentation is important to us. Send suggestions for improvements and reports on errors or omissions to clustering_docs@symantec.com.

Include the title and document version (located on the second page), and chapter and section titles of the text on which you

are reporting.

Maintenance agreement resources

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

Asia-Pacific and Japan	contractsadmin@symantec.com
Europe, Middle-East, and Africa	semea@symantec.com
North America and Latin America	supportsolutions@symantec.com

Additional enterprise services

Symantec offers a comprehensive set of services that allow you to maximize your investment in Symantec products and to develop your knowledge, expertise, and global insight, which enable you to manage your business risks proactively.

Enterprise services that are available include the following:

Symantec Early Warning Solutions	These solutions provide early warning of cyber attacks, comprehensive threat analysis, and countermeasures to prevent attacks before they occur.
Managed Security Services	These services remove the burden of managing and monitoring security devices and events, ensuring rapid response to real threats.
Consulting Services	Symantec Consulting Services provide on-site technical expertise from Symantec and its trusted partners. Symantec Consulting Services offer a variety of prepackaged and customizable options that include assessment, design, implementation, monitoring, and management capabilities. Each is focused on establishing and maintaining the integrity and availability of your IT resources.
Educational Services	Educational Services provide a full array of technical training, security education, security certification, and awareness communication programs.

To access more information about Enterprise services, please visit our Web site at the following URL:

www.symantec.com

Select your country or language from the site index.

Contents

Chapter 1 Introduction

About VCS agents 14
How agents work 15
About intentional offline of applications 15
About the agent framework 15
Resource type definitions 16
Agent functions (entry points) 16
About on-off, on-only, and persistent resources 16
About attributes 18
About developing an agent 22
Considerations for the application 22
High-level overview of the agent development process 23

Chapter 2 Agent entry point overview

About agent entry points 26
Supported entry points 26
How the agent framework interacts with entry points 26
Agent entry points described 27
About the monitor entry point 27
About the info entry point 27
About the online entry point 30
About the offline entry point 31
About the clean entry point 32
About the action entry point 34
About the attr_changed entry point 35
About the open entry point 35
About the close entry point 36
About the shutdown entry point 36
Return values for entry points 37
Considerations for using C++ or script entry points 38
About the VCSAgStartup routine 38
About the agent information file 40
Example agent information file (UNIX) 40
Implementing the agent XML information file 43
About the ArgList Attribute 43

ArgList attribute for agents registered as V50 and later 44
ArgList Attribute for agents registered as V40 and earlier 44

Chapter 3 Creating entry points in C++

About creating entry points in C++ 48
Entry point examples in this chapter 49
Data Structures 50
Syntax for C++ entry points 52
Syntax for C++ VCSAgStartup 52
Syntax for C++ monitor 53
Syntax for C++ info 54
Syntax for C++ online 60
Syntax for C++ offline 61
Syntax for C++ clean 62
Syntax for C++ action 63
Syntax for C++ attr_changed 65
Syntax for C++ open 67
Syntax for C++ close 68
Syntax for C++ shutdown 69
Agent framework primitives 70
VCSAgRegisterEPStruct 70
VCSAgSetCookie 70
VCSAgSetCookie2 71
VCSAgRegister 73
VCSAgUnregister 74
VCSAgGetCookie 75
VCSAgStrlcpy 76
VCSAgStrlcat 76
VCSAgSnprintf 76
VCSAgCloseFile 76
VCSAgDelString 76
VCSAgExec 77
VCSAgExecWithTimeout 78
VCSAgGenSnmpTrap 80
VCSAgSendTrap 80
VCSAgLockFile 80
VCSAgInitEntryPointStruct 81
VCSAgSetStackSize 81
VCSAgUnlockFile 81
VCSAgDisableCancellation 82
VCSAgRestoreCancellation 82
VCSAgSetEntryPoint 83
VCSAgSetLogCategory 83

VCSAgGetProductName 83
Agent Framework primitives for XRM support 84
VCSAgIsContainerAvailable 84
VCSAgExecInContainerWithTmo 84
VCSAgGetUID 85
VCSAgIsPidInContainer 85
VCSAgIsProcInContainer 85
VCSAgGetContainerID2 85
VCSAgGetContainerName2 86

Chapter 4 Creating entry points in scripts

About creating entry points in scripts 88
Rules for using script entry points 88
Parameters and values for script entry points 88
ArgList attributes 89
Examples 89
Syntax for script entry points 90
Syntax for the monitor script 90
Syntax for the online script 90
Syntax for the offline script 90
Syntax for the clean script 91
Syntax for the action script 91
Syntax for the attr_changed script 91
Syntax for the info script 91
Syntax for the open script 92
Syntax for the close script 92
Syntax for the shutdown script 92
Example script entry points 93

Chapter 5 Logging agent messages

About logging agent messages 98
Logging in C++ and script-based entry points 98
Agent messages: format 99
C++ agent logging APIs 100
Agent application logging macros for C++ entry points 101
Agent debug logging macros for C++ entry points 102
Severity arguments for C++ macros 103
Initializing function_name using VCSAG_LOG_INIT 104
Log category 105
Examples of logging APIs used in a C++ agent 106
Script entry point logging functions 109
VCSAG_SET_ENVS 110

VCSAG_LOG_MSG 112
VCSAG_LOGDBG_MSG 113
Using the functions in scripts 114
Example of logging functions used in a script agent 115

Chapter 6 Building a custom agent

Files for use in agent development 118
Creating the type definition file for a custom agent 119
Example: FileOnOffTypes.cf 119
Example: Type definition for a custom agent that supports intentional offline 119
Requirements for creating the agentTypes.cf file 119
Adding the custom type definition to the configuration 119
Building a custom agent on UNIX 120
Implementing entry points using scripts 120
Example: Using script entry points on UNIX 121
Example: Using VCSAgStartup() and script entry points on UNIX 122
Implementing entry points using C++ 123
Example: Using C++ entry points on UNIX 123
Example: Using C++ and script entry points on UNIX 126
Installing the custom agent 129
Defining resources for the custom resource type 130
Sample resource definition 130
Deploying custom agents on virtual machines running Linux 132

Chapter 7 Testing agents

About testing agents 136
Using debug messages 136
Using the engine process to test agents 137
Test commands 137
Using the AgentServer utility to test agents 138

Chapter 8 Static type attributes

About static attributes 142
Overriding static type attributes 142
Static type attribute definitions 143
ActionTimeout 143
AgentClass 143
AgentFailedOn 143
AgentPriority 143
AgentReplyTimeout 144
AgentStartTimeout 144

ArgList 144
AttrChangedTimeout 145
CleanTimeout 145
CloseTimeout 145
ContainerType 145
ConfInterval 147
FaultOnMonitorTimeouts 148
FireDrill 148
InfoInterval 148
InfoTimeout 149
IntentionalOffline 149
LogDbg 149
LogFileSize 150
ManageFaults 151
MonitorInterval 152
MonitorStatsParam 152
MonitorTimeout 152
NumThreads 153
OfflineMonitorInterval 153
OfflineTimeout 153
OnlineRetryLimit 154
OnlineTimeout 154
OnlineWaitLimit 154
OpenTimeout 154
Operations 155
RegList 155
RestartLimit 156
ScriptClass 156
ScriptPriority 156
SupportedActions 157
ToleranceLimit 157
Scheduling class and priority configuration support 158
Priority ranges 158
Default scheduling classes and priorities 159
Initializing attributes in the configuration file 159
Setting attributes dynamically from the command line 160

Chapter 9 State transition diagrams

State transitions 162
State transitions with respect to ManageFaults attribute 171

Chapter 10 Internationalized messages

Creating SMC files 180
SMC format 180
Example SMC file 180
Formatting SMC files 181
Naming SMC files, BMC files 181
Message examples 182
Using format specifiers 182
Converting SMC files to BMC files 183
Storing BMC files 183
Displaying the contents of BMC files 183
Using BMC Map Files 184
Location of BMC Map Files 184
Creating BMC Map Files 184
Example BMC Map File 184
Updating BMC Files 185

Appendix A Using pre-5.0 VCS agents

Using pre-5.0 VCS agents and registering them as V50 or later 187
Outline of steps to change V40 agents V50 187
Name-value tuple format for agents registered as V50 or later 188
Sourcing ag_i18n_inc modules in script entry points 190
Guidelines for using pre-VCS 4.0 Agents 191
Log messages in pre-VCS 4.0 agents 192
Mapping of log tags (pre-VCS 4.0) to log severities (VCS 4.0) 192
How Pre-VCS 4.0 Messages are Displayed by VCS 4.0 and Later 193
Comparing Pre-VCS 4.0 APIs and VCS 4.0 Logging Macros 193
Pre-VCS 4.0 Message APIs 194
VCSAgLogConsoleMsg 194
VCSAgLogI18NMsg 195
VCSAgLogI18NMsgEx 196
VCSAgLogI18NConsoleMsg 197
VCSAgLogI18NConsoleMsgEx 198

Index 199

Chapter 1

Introduction

- About VCS agents
- How agents work
- About developing an agent

About VCS agents

Agents are programs that manage resources, such as a disk group or an IP address, within a cluster environment. Each type of resource requires an agent. The agent acts as an intermediary between VCS and the resources it manages, typically by bringing it online, monitoring its state, or taking it offline.

Agents packaged with are referred to as *bundled agents*. Examples of bundled agents include the IP (Internet Protocol) and NIC (network interface card) agents. For more information on bundled agents, including their attributes and modes of operation, see the *Bundled Agents Reference Guide*.

Agents packaged separately from the product for use with are referred to as *high availability agents*. They include agents for Oracle, for example. Contact your Symantec sales representative for information on how to purchase these agents for your configuration.

Note: Custom agents, that is, agents developed outside of Symantec, are not supported by Symantec Technical Support.

How agents work

A single agent can manage multiple resources of the same type on one system. For example, the NIC agent manages all NIC resources. The resources to be managed are those defined within the configuration files.

When the VCS engine process, had, comes up on a system, it automatically starts the agents required for the types of resources that are to be managed on the system and provides the agents the specific configuration information for those resources. An agent carries out the commands from to bring resources online, monitor their status, and take them offline, as needed. When an agent crashes or hangs, detects the fault and had restarts the agent.

Note: The VCS engine process is known as "`had`." The acronym stands for "high-availability daemon."

About intentional offline of applications

Certain agents can identify when an application has been intentionally shut down outside of VCS control.

If an administrator intentionally shuts down an application outside of VCS control, VCS does not treat it as a fault. VCS sets the service group state as offline or partial, depending on the state of other resources in the service group.

This feature allows administrators to stop applications without causing failovers. The feature is available for V51 agents.

See "IntentionalOffline" on page 149.

About the agent framework

The agent framework is a set of predefined functions compiled into the agent for each resource type. These functions include the ability to connect to the VCS engine and to understand the common configuration attributes, such as RestartLimit and MonitorInterval. When an agent's code is built in C++, the agent framework is compiled in the agent with an include statement. When an agent is built using script languages, such as shell or Perl, the ScriptAgent on UNIX or the DefaultAgent.dll on Windows provides the agent framework functions. The agent framework handles much of the complexity that need not concern the agent developer.

Resource type definitions

The agent for each type of resource requires a resource type definition that describes the information an agent needs to control resources of that type. The type definition can be considered similar to a header file in a C program. The type definition defines the data types of variables (attributes) to provide error checking, and to provide default values for certain attributes for the entire type.

All resource types need definition. One of the attributes that is defined for the IP resource type is the IPAddress attribute, which stores the actual IP address of a specific IP resource. This attribute is defined as a 'string-scalar' as the use of periods to denote an IP address makes the acceptable values for the attribute a string.

Agent functions (entry points)

An entry point is either a C++ function or a script (shell or Perl, for example) used by the agent to carry out a specific task on a resource. The agent framework supports a specific set of entry points, each of which is expected to do a different task and return. For example, the online entry point brings a resource online.

See “Supported entry points” on page 26.

An agent developer implements the entry points for a resource type that the agent uses to carry out the required tasks on the resources of that type. For example, in the online entry point for the Mount type, the agent developer includes the logic to mount a file system based on the parameters provided to the entry point. These parameters are attributes for a particular resource, for example, mount point, device name, and mount options. In the monitor entry point, the agent developer checks the state of the mount resource and returns a code to indicate whether the mount resource is online or offline.

See “About agent entry points” on page 26.

About on-off, on-only, and persistent resources

Different types of resources require different types of control, requiring implementation of different entry points. Resources can be classified as on-off, on-only, or persistent, depending on the entry points required to control them.

- On-off resources
  Most resources are on-off, meaning agents start and stop them as required. For example, the engine assigns an IP address to a specified NIC when bringing a resource online and removes the assigned IP address when taking the service group offline. Another example is the DiskGroup resource. The engine imports a disk group when needed and deports it when

it is no longer needed. For agents of on-off resources, all entry points can be implemented.

- On-only resources
  An on-only resource is brought online when required by the engine, but it is not taken offline when the associated service group is taken offline. For example, in the case of the FileOnOnly resource, the engine creates the specified file when required, but does not delete the file if the associated service group is taken offline. For agents of on-only resources, the offline entry point is not needed or invoked.
- Persistent resources
  A persistent resource cannot be brought online or taken offline, yet the resource must be present in the configuration. For example, a NIC resource cannot be started or stopped, but it is required to configure an IP address. The agent monitors persistent resources to ensure their status and operation. An agent for a persistent resource does not require or invoke the online or offline entry points. It uses only the monitor entry points.

About attributes

VCS has the following types of attributes, depending on the object the attribute applies to.

Resource type attributes

Attributes that define the resource types in VCS. These can be further classified as:

- **Type-independent**—Attributes that all agents (or resource types) understand. Examples: RestartLimit and MonitorInterval; these can be set for any resource type.
  Typically, these attributes are set for all resources of a specific type. For example, if you set the MonitorInterval for the IP resource type, the same value applies to all resources of type IP. You can also override the values of these attributes.
- **Type-dependent**—Attributes that apply to a particular resource type. These attributes appear in the type definition file (.cf) for the agent. Examples: The MountPath attribute applies only to the Mount resource type. The Address attribute applies only to the IP resource type.
  Attributes defined in the file types.cf apply to all resources of the resource type. Defining these attributes in the main.cf file overrides the values in the .cf file for a specific resource.
  For example, setting StartVolumes = 1 for the DiskGroup resource type defaults StartVolumes to True for all DiskGroup resources. Setting the value in main.cf overrides the value on a per-resource basis.
- **Static**—These attributes apply for every resource of a particular type. These attributes are prefixed with the term static and are not included in the resource's argument list. You can override some static attributes and assign them resource-specific values.

Attribute data types

VCS supports the following data types for attributes.

String

A string is a sequence of characters enclosed by double quotes. A string may also contain double quotes, but the quotes must be immediately preceded by a backslash. A backslash is represented in a string as \\. Quotes are not required if a string begins with a letter, and contains only letters, numbers, dashes (-), and underscores (_). For example, a string defining a network interface such as hme0 or eth0 does not require quotes as it contains only letters and numbers. However a string defining an IP address contains periods and requires quotes, such as: "192.168.100.1"

Integer — Signed integer constants are a sequence of digits from 0 to 9. They may be preceded by a dash, and are interpreted in base 10. Integers cannot exceed the value of a 32-bit signed integer: 21471183247.

Boolean — A boolean is an integer, the possible values of which are 0 (false) and 1 (true).

Attribute dimensions

VCS attributes have the following dimensions.

Scalar — A scalar has only one value. This is the default dimension.

For example:

```
str MountPoint
```

When values are assigned to a scalar attribute in the main.cf configuration file, it might resemble:

```
MountPoint = "/Backup"
```

Vector — A vector is an ordered list of values. Each value is indexed using a positive integer beginning with zero. Use a comma (,) or a semi-colon (;) to separate values.

A set of brackets ([]) after the attribute name denotes that the dimension is a vector.

For example:

```
str BackupSys[]
```

When values are assigned to a vector attribute in the `main.cf` configuration file, the attribute definition might resemble:

```
BackupSys[] = { sysA, sysB, sysC }
```

For example, an agent's ArgList is defined as:

```
static str ArgList[] = {RVG, DiskGroup, Primary, SRL,
Links}
```

Keylist — A keylist is an unordered list of strings, and each string is unique within the list. Use a comma (,) or a semi-colon (;) to separate values.

For example, to designate the list of systems on which a service group will be started with VCS (usually at system boot):

```
AutoStartList = {SystemA; SystemB; SystemC}
```

For example:

```
keylist BackupVols = {}
```

When values are assigned to a keylist attribute in the `main.cf` file, it might resemble:

```
BackupVols = { vol1, vol2 }
```

Association

An association is an unordered list of name-value pairs. Use a comma (,) or a semi-colon (;) to separate values.

A set of braces ({}) after the attribute name denotes that an attribute is an association.

For example, to designate the list of systems on which the service group is configured to run and the system's priorities:

```
SystemList = {SystemA=1, SystemB=2, SystemC=3}
```

For example:

```
int BackupSysList {}
```

When values are assigned to an association attribute in the `main.cf` file, it might resemble:

```
BackupSysList{} = { sysa=1, sysb=2, sysc=3 }
```