

Cluster Server Agents for PostgreSQL Database and Replication Installation and Configuration Guide

AIX, Linux, Solaris SPARC

7.0

VERITAS™

Veritas InfoScale™ Availability Agents

Last updated: 2021-01-08

Legal Notice

Copyright © 2021 Veritas Technologies LLC. All rights reserved.

Veritas and the Veritas Logo are trademarks or registered trademarks of Veritas Technologies LLC or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

This product may contain third-party software for which Veritas is required to provide attribution to the third-party ("Third-Party Programs"). Some of the Third-Party Programs are available under open source or free software licenses. The License Agreement accompanying the Software does not alter any rights or obligations you may have under those open source or free software licenses. Refer to the third-party legal notices document accompanying this Veritas product or available at:

<https://www.veritas.com/about/legal/license-agreements>

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Veritas Technologies LLC and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. VERITAS TECHNOLOGIES LLC SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, et seq. "Commercial Computer Software and Commercial Computer Software Documentation," as applicable, and any successor regulations, whether delivered by Veritas as on premises or hosted services. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

Veritas Technologies LLC
2625 Augustine Drive
Santa Clara, CA 95054
<http://www.veritas.com>

Technical Support

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

<https://www.veritas.com/support>

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

<https://my.veritas.com>

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

Worldwide (except Japan)

CustomerCare@veritas.com

Japan

CustomerCare_Japan@veritas.com

Documentation

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

<https://sort.veritas.com/documents>

Documentation feedback

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

infoscaledocs@veritas.com

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

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

Veritas Services and Operations Readiness Tools (SORT)

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

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

Contents

Chapter 1	Introducing the agents for PostgreSQL Database and Replication	7
	About the InfoScale Availability agents for PostgreSQL Database and Replication	7
	Supported software	9
	Supported configurations for PostgreSQL Streaming Replication	9
	Features of the agent	11
	How the agent supports intelligent resource monitoring	12
	PostgreSQL Database agent functions	12
	Online	12
	Offline	13
	Monitor	14
	Clean	15
	Action entry points	15
	PostgreSQL Replication agent functions	16
Chapter 2	Installing, upgrading, and removing the agents for PostgreSQL Database and Replication	19
	Before you install the Cluster Server agent for PostgreSQL	19
	About the ACC library	20
	Installing the ACC library	20
	Installing the ACC library IPS package on Oracle Solaris 11 systems	21
	Installing the ACC library package on Solaris brand non-global zones	22
	Installing the agent in a VCS environment	23
	Installing the agent IPS package on Oracle Solaris 11 systems	24
	Installing agent packages on Solaris brand non-global zones	25
	Installing the agent in a Solaris 10 brand zone	26
	Uninstalling the agent in a VCS environment	26
	Removing the ACC library	27

	Upgrading the PostgreSQL and the PgSQLRep agents	28
Chapter 3	Configuring the agents for PostgreSQL Database and Replication	32
	About configuring the Cluster Server agent for PostgreSQL	32
	Importing the agent types files in a VCS environment	33
	PostgreSQL agent attributes	34
	Executing a customized monitoring program	45
	Setting up detail monitoring for the VCS agent for PostgreSQL	45
Chapter 4	Enabling the PostgreSQL and the PgSQLRep agents to support IMF	48
	About Intelligent Monitoring Framework	48
	Benefits of IMF	49
	Agent functions for the IMF functionality	49
	imf_init	49
	imf_getnotification	49
	imf_register	49
	Attributes that enable IMF	50
	IMF	50
	IMFRegList	51
	Before you enable the agent to support IMF	51
	Enabling the agent to support IMF	52
	If VCS is in a running state	52
	Restarting the agent	54
	If VCS is not in a running state	54
	Disabling intelligent resource monitoring	55
	Sample IMF configurations	55
Chapter 5	Configuring the service groups for PostgreSQL using the CLI	57
	About configuring service groups for PostgreSQL	57
	Before configuring the service groups for PostgreSQL	58
	PostgreSQL entities in a clustered environment	58
	Virtualizing PostgreSQL	59
	Creating service groups for PostgreSQL under Solaris non-global zones	59
	Configuring service groups for PostgreSQL Replication	60
	Configuring PostgreSQL nofailover trigger	63

Chapter 6	Troubleshooting the agent for PostgreSQL	65
	Using the correct software and operating system versions	65
	Meeting prerequisites	65
	Verifying virtualization	66
	Starting the PostgreSQL server outside a cluster	66
	Reviewing error log files	67
	Using trace level logging	67
	Troubleshooting the configuration for IMF	69
	Known issues	70
Appendix A	Sample Configurations	71
	About sample configurations for the agents for PostgreSQL	71
	Sample agent type definition for PostgreSQL	71
	Sample configuration files	73
	Sample service group configurations for PostgreSQL	77

Introducing the agents for PostgreSQL Database and Replication

This chapter includes the following topics:

- [About the InfoScale Availability agents for PostgreSQL Database and Replication](#)
- [Supported software](#)
- [Supported configurations for PostgreSQL Streaming Replication](#)
- [Features of the agent](#)
- [How the agent supports intelligent resource monitoring](#)
- [PostgreSQL Database agent functions](#)
- [PostgreSQL Replication agent functions](#)

About the InfoScale Availability agents for PostgreSQL Database and Replication

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

The InfoScale Availability agents for PostgreSQL manage and provide high availability for PostgreSQL servers and EDB Postgres Advanced Servers in a clustered environment.

The agent for PostgreSQL Database can bring a specific PostgreSQL server instance online and monitor its state. The agent can also detect failures and can turn off the instance in case of a failure.

The agent for PostgreSQL Replication provides high availability for the Synchronous Streaming Replication feature of PostgreSQL. PostgreSQL itself uses this feature to replicate data to standby PostgreSQL server instances and thereby provide fault tolerance and disaster recovery to achieve high availability.

Note: The agent for PostgreSQL Replication (PgSQLRep) is available for the Linux platform only.

The PgSQLRep agent supports the following use cases only:

- A two-node failover cluster
- Two n-node global clusters with the Master instance in one cluster and the Slave instance in the other cluster
- Two n-node global clusters with Veritas Cluster File System (CFS) as the underlying storage management component

See [“Supported configurations for PostgreSQL Streaming Replication”](#) on page 9.

The agents for PostgreSQL Database and Replication support the following operations for Synchronous Streaming Replication:

- Monitoring the replication link on the master node
- Either manual or automatic takeover on the standby node by using the PromoteSlaveAction entry point on the standby PostgreSQL resource
- Either manual or automatic re-registration of the old master node in case of service group switchover using the RewindAction entry point
- Failover and switch of a PostgreSQL Master server in a cluster
- Failover and switch of a PostgreSQL Standby server in a cluster
- Promotion of the standby node to primary in case of DBMS failure on the master node
- Promotion of the standby node to primary in case of master node failure

If the master node fails, the PostgreSQL Replication agent promotes the standby node.

If the master node is not shut down gracefully before it tries to re-register as a slave, the rewind operation fails.

When the RestartdbToRewind attribute is set to 1, the database instance is started and stopped gracefully before the rewind operation.

To set up Streaming Replication and to efficiently perform takeover and failover, provide appropriate values for the following settings in these configuration files:

postgresql.conf

- archive_mode
- archive_command
- synchronous_standby_names
- listen_addresses
- wal_level
- max_wal_senders
- max_replication_slots
- wal_log_hints
- synchronous_commit

For PostgreSQL 12 and later:
postgresql.auto.conf, or any other configuration file that the agent should copy to the data directory on the standby node.

- primary_conninfo
- restore_command

For versions earlier than PostgreSQL 12: recovery.conf

- standby_mode
- primary_conninfo
- restore_command
- recovery_target_timeline
- primary_slot_name

Supported software

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

Supported configurations for PostgreSQL Streaming Replication

InfoScale supports the following types of high availability configurations using the VCS agent for PostgreSQL Streaming Replication.

Figure 1-1 depicts a two-server Primary-Standby configuration. Both the nodes, N1 and N2, are a part of the same cluster, Pg_cluster. N1 is the Primary server and N2 is the Standby server. Replication is active from N1 to N2.

Figure 1-1 Two-node failover cluster

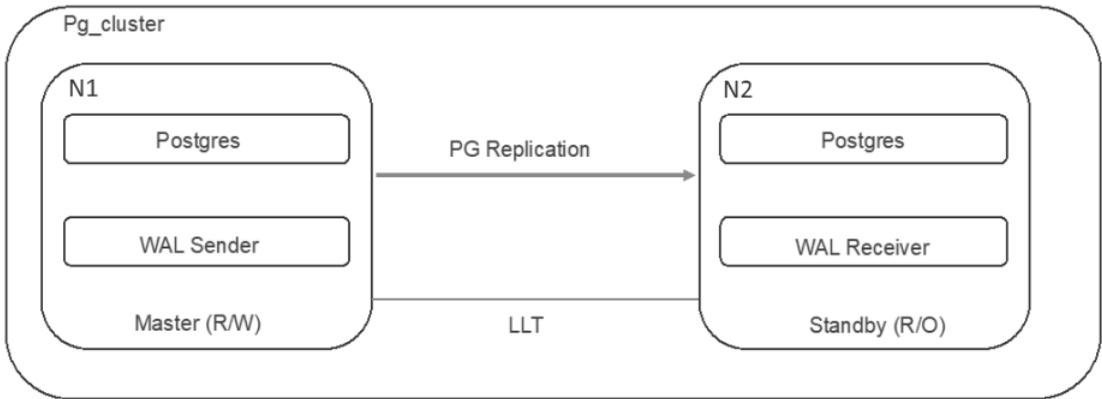


Figure 1-2 depicts a Primary-Standby configuration, where the cluster can have two or more nodes. There are two clusters, PG_clus1 and Pg_Clus2. Nodes N1 and N2 are a part of PG_clus1, and nodes N3 and N4 are a part of PG_clus2. PG_clus1 is the Primary cluster and PG_clus2 is the Standby cluster. Replication is active from a single node of PG_clus1 to a single node of PG_clus2. The storage is imported on only one node in a cluster at any given time.

Figure 1-2 Two n-node global clusters with Master instance in one cluster and Slave instance in the other

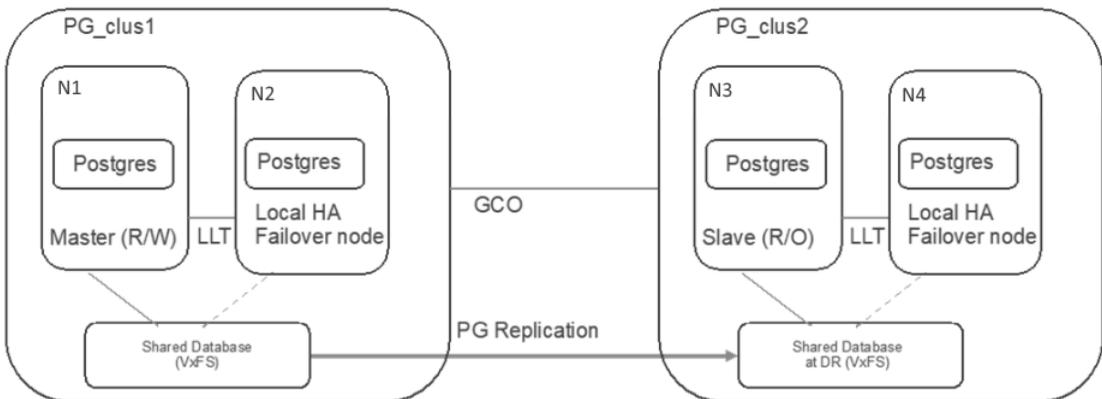
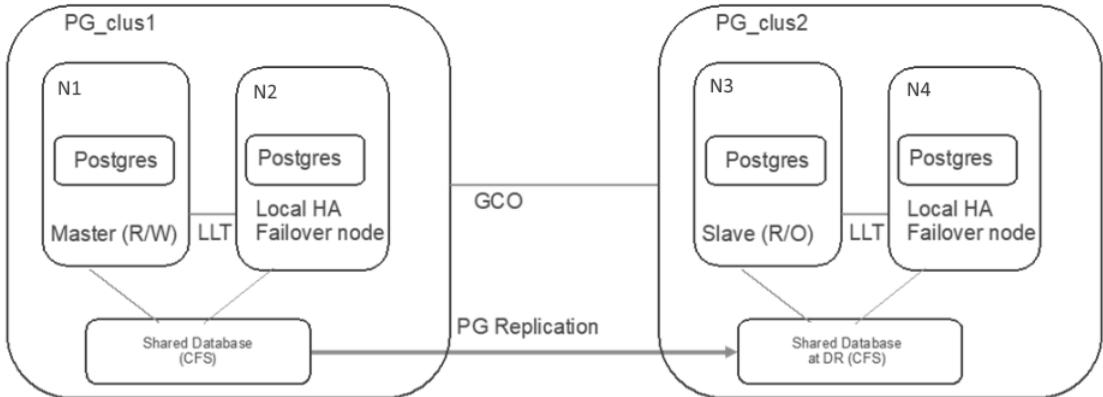


Figure 1-3 depicts a Primary-Standby configuration, where a cluster can have two or more nodes. There are two clusters, PG_clus1 and Pg_Clus2. Nodes N1 and N2 are a part of PG_clus1, and nodes N3 and N4 are a part of PG_clus2. PG_clus1 is the Primary cluster and PG_clus2 is the Standby cluster. Replication is active from a single node of PG_clus1 to a single node of PG_clus2. The storage is

imported on all the nodes of a cluster at any given time. The Veritas Cluster File System (CFS) component is used to support this type of configuration.

Figure 1-3 Two n-node global clusters with Veritas Cluster File System (CFS) as the underlying storage management component



For details on the service group dependencies for each of these supported configurations: See [“Configuring service groups for PostgreSQL Replication”](#) on page 60.

Features of the agent

The following are the features of the Cluster Server agent for PostgreSQL:

- Support for validation of attributes that are based on the agent functions
The agent can validate attributes in each agent function before the actual data processing starts.
- Support for First Failure Data Capture (FFDC)
In case of a fault, the agent generates a huge volume of the debug logs that enable troubleshooting of the fault.
- Support for Fast First Level Monitor (FFLM)
The agent maintains PID files based on search patterns to expedite the monitoring process.
- Support for external user-supplied monitor utilities
The agent enables user-specified monitor utilities to be plugged in, in addition to the built-in monitoring logic. This enables administrators to completely customize the monitoring of the application.
- Support for intelligent resource monitoring and poll-based monitoring

The agent supports the Cluster Server Intelligent Monitoring Framework (IMF) feature. IMF allows the agent to register the resources to be monitored with the IMF notification module so as to receive immediate notification of resource state changes without having to periodically poll the resources.

See [“About Intelligent Monitoring Framework”](#) on page 48.

- Delayed agent function
The agent manages the first monitor after online for slow initializing applications.

How the agent supports intelligent resource monitoring

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

When an IMF-enabled agent starts up, the agent initializes the Asynchronous Monitoring Framework (AMF) kernel driver. After the resource is in a steady state, the agent registers with the AMF kernel driver, the details of the resource that are required to monitor the resource.

For example, the PostgreSQL agent registers the PIDs of the PostgreSQL processes and the PgSQLRep agent registers the `postgres walsender` process with the AMF kernel driver.

The `imf_getnotification` function of the agent waits for any resource state changes. When the AMF kernel driver module notifies the `imf_getnotification` function about a resource state change, the agent framework runs the monitor agent function to ascertain the state of that resource. The agent notifies the state change to VCS, which then takes appropriate action.

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

PostgreSQL Database agent functions

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

Online

The online function performs the following tasks:

- Verifies that the required attributes are set correctly.

- Verifies that the PostgreSQL server instance is not already online. If the instance is online, the online operation exits immediately.
- Kills any remaining PostgreSQL processes by using the user name associated with the specific resource.
- Configures this instance to connect as standby before startup if a master server is already running.
- Starts and stops the database instance gracefully if the Rewind operation fails and the RestartdbToRewind attribute is set to 1, and then attempts the Rewind operation again.
- Attempts to start the PostgreSQL server instance with the command:

```
$ baseDirectory/pg_ctl start -w -D dataDirectory
-o "-h hostName -p portNumber" startOptions
```

If MonitorReplication is enabled, the command used is:

```
$ baseDirectory/pg_ctl start -w -D dataDirectory
-o "-p portNumber" startOptions
```

The command always gets executed in the context of a PostgreSQL user, who has the privileges to start and stop the postgres (postmaster) process. If systemd is supported for the platform, and the UseSystemD attribute is enabled, the `systemctl start serviceName` command is used.

Note: In this case, service file must be updated appropriately with the proper start and stop commands.

- Checks if the server has started up completely.
- Gives the control back to VCS High Availability Daemon (HAD).

Offline

The offline function performs the following tasks:

- Verifies that the required attributes are set correctly.
- Verifies that the PostgreSQL server instance is not offline.
- If the instance is already offline, the operation verifies if any processes belonging to this PostgreSQL resource exist.
- Attempts to stop the PostgreSQL server instance with the command:

```
$ baseDirectory/pg_ctl stop -w -D dataDirectory stopOptions
```

The `pg_ctl` command uses the option specified in the `StopOpts` attribute to shut down the PostgreSQL database server. If no option is specified in the `StopOpts` attribute, the agent stops the database server using the default `-m smart` shutdown option.

The command always gets executed in the context of a PostgreSQL user, who has the privileges to start and stop the `postgres` (`postmaster`) process.

If `systemd` is supported for the platform, and the `UseSystemD` attribute is enabled, the `systemctl stop serviceName` command is used.

- Gives the control back to HAD.

Monitor

The monitor function monitors the states of the PostgreSQL servers on all nodes within the cluster. The operation performs the following tasks:

- The monitor function conducts a first-level check to determine that the PostgreSQL server processes are running on the system in the cluster. If the first-level check does not find these processes running on the node, the check exits immediately and reports the instance as OFFLINE.

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

See [“How the agent supports intelligent resource monitoring”](#) on page 12.

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

See [“MonitorFreq”](#) on page 50.

- If the `SecondLevelMonitor` attribute is set to a value greater than 0, the monitor operation conducts a second level check.

During Second Level Monitoring, the agent uses the monitor command to verify that the PostgreSQL server is up.

```
$ baseDirectory/pg_ctl status -D dataDirectory
```

The command is executed in the context of a PostgreSQL user, who has the privilege to monitor the `postgres` (`postmaster`) process.

Note: The attribute used to configure the second-level check and its frequency depends on the software versions of VCS and PostgreSQL agent you have installed: For VCS 5.1 SP1 or later with PostgreSQL agent version 5.1.1.0, use the `LevelTwoMonitorFreq` attribute. For VCS 5.1 or earlier with PostgreSQL agent 5.1.0.0 or earlier, use the `SecondLevelMonitor` attribute.

- Depending on the value of the MonitorProgram attribute, the monitor operation can perform a customized check using a user-supplied monitoring utility. See [“PostgreSQL agent attributes”](#) on page 34.
- If the MonitorReplication attribute is set to 1, the replication link is monitored by using the following command on the master node:

```
$ baseDirectory/psql -A -t -h hostName -p portNumber  
-c "select * from pg_stat_replication"
```

Clean

In case of a failure or after an unsuccessful attempt to bring a PostgreSQL server instance online or take a PostgreSQL server instance offline, the clean operation performs the following tasks:

- Attempts to gracefully shut down the PostgreSQL server instance with the command:

```
$ baseDirectory/pg_ctl stop -w -D dataDirectory
```

The command always gets executed in the context of a PostgreSQL user, who has the privileges to start and stop the postgres (postmaster) process.
- The clean operation kills the parent PostgreSQL process and its remaining child processes, if any, pertaining to this PostgreSQL instance.
- Gives the control back to HAD.

Note: For information about the additional functions of the agent for PostgreSQL when IMF is enabled: See [“Agent functions for the IMF functionality”](#) on page 49.

Action entry points

The PostgreSQL provides the following action entry points for Streaming Replication.

PromoteSlaveAction

This action entry point:

- Runs on the standby node.
- Takes the Global group on the old master node offline, if it is online.
- Stops the old master node.
- Promotes the standby to master by using the command:

```
$ baseDirectory/pg_ctl -D dataDirectory promote
```

- Issues a checkpoint by using the command:

```
$ baseDirectory/psql -A -t "-h $sHostName -p $iPortNumber  
-c \"checkpoint\""
```

To allow sufficient time for the promote slave operation to complete, the agent sets the timeout value of this action entry point to twice the value of the OfflineTimeout attribute.

Example: # hares -action pgl PromoteSlaveAction -sys vmrac019

RewindAction

This action entry point runs on the old master node after it shuts down gracefully and executes the following command:

```
$ baseDirectory/pg_rewind -D dataDirectory  
--source-server="sourceConnectionString"
```

Example: # hares -action pgl RewindAction -sys vmrac017

The timeout value for this action is set to the OnlineTimeout value.

BackupAction

This action entry point runs on a secondary node, and it performs the backup from the primary instance by using the command that is specified in the BackupCmd attribute.

Example: # hares -action pgl PromoteSlaveAction -sys vmrac019

The timeout value for this action is set to the OnlineTimeout value.

PostgreSQL Replication agent functions

The following table lists the operations or functions that the VCS agent for PostgreSQL Replication can perform.

Table 1-1 PostgreSQL Replication agent functions

Function	Description
Open	Deletes the lock file if the underlying PostgreSQL Database is in the standby mode or its service group is in the OFFLINE state.

Table 1-1 PostgreSQL Replication agent functions (*continued*)

Function	Description
Online	<ul style="list-style-type: none"> ■ Verifies that the required attributes are set correctly. ■ Returns online if the underlying database is already in the primary mode and if replication is configured. ■ Promotes the underlying database to the primary mode, based on the values of the SplitTakeOver and the AutoTakeOver attributes, if the database is in the standby mode. The following command is used to promote the database to the primary mode: <pre># baseDir/pg_ctl promote -w -D 'dataDir' -o "-p 'portNum'"</pre> ■ Creates a lock file for this resource. ■ Re-registers the old primary as standby based on the value of the RegistrationOfStandby attribute. Uses the RegisterStandby action entry point for re-registration. ■ Checks whether the server has started properly. ■ Gives the control back to the VCS High Availability Daemon (HAD).
Monitor	<p>Verifies whether the lock file for the PgSQLRep resource is present, and:</p> <ul style="list-style-type: none"> ■ returns Online if the lock file is present. ■ returns Offline if the lock file is not present. <p>If the SwitchMode attribute is set:</p> <ul style="list-style-type: none"> ■ changes the replication mode from synchronous to asynchronous when replication link is broken. ■ switches the replication mode back to synchronous when the replication link is re-established.
Offline	<ul style="list-style-type: none"> ■ Deletes the lock file. ■ Gives the control back to HAD.
Clean	<ul style="list-style-type: none"> ■ Deletes the lock file. ■ Gives the control back to HAD.

RegisterStandby action entry point

The agent provides this action entry point, which runs on the old-primary node and performs the following tasks:

- Takes the PostgreSQL group offline on the old master node if it is online.
- If the RegistrationOfStandby attribute is set to 1:
 - Clears the status of the PostgreSQL group on the old master node if it is FAULTED.
 - Brings the PostgreSQL group online in the standby mode if the RegistrationOfStandby attribute is set to 1.

To allow sufficient time for the registration of the standby to complete, the agent sets the timeout value of this action entry point to the sum of the values of the ActionTimeout, OnlineTimeout, and OfflineTimeout attributes.

A sample action execution command used is as follows:

```
# hares -action resourceName RegisterStandby -actionargs  
valueOfRegistrationOfStandby -sys systemName -clus clusterName
```

Installing, upgrading, and removing the agents for PostgreSQL Database and Replication

This chapter includes the following topics:

- [Before you install the Cluster Server agent for PostgreSQL](#)
- [About the ACC library](#)
- [Installing the ACC library](#)
- [Installing the agent in a VCS environment](#)
- [Uninstalling the agent in a VCS environment](#)
- [Removing the ACC library](#)
- [Upgrading the PostgreSQL and the PgSQLRep agents](#)

Before you install the Cluster Server agent for PostgreSQL

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

Before you install the agent for PostgreSQL, ensure that the following prerequisites are met.

- Install and configure Cluster Server.
For more information on installing and configuring Cluster Server, refer to the Cluster Server installation and configuration guides.
- Install the latest version of ACC Library.
To install or update the ACC Library package, locate the library and related documentation in the Agent Pack tarball.
See “[About the ACC library](#)” on page 20.

About the ACC library

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

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

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

Installing the ACC library

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

To install the ACC library

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

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

- 3 If you downloaded the complete Agent Pack tar file, navigate to the directory containing the package for the platform running in your environment.

AIX `cd1/aix/vcs/application/acc_library/version_library/pkg`s

Linux `cd1/linux/generic/vcs/application/acc_library/version_library/rpm`s

Solaris `cd1/solaris/dist_arch/vcs/application/acc_library/version_library/pkg`s

- 4 If you downloaded the individual ACCLib tar file, navigate to the `pkg`s directory (for AIX and Solaris), or `rpm`s 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 \
VRTSacclib-VersionNumber-GA_GENERIC.noarch.rpm`

Solaris `# pkgadd -d VRTSacclib.pkg`

See [“Installing the ACC library IPS package on Oracle Solaris 11 systems”](#) on page 21.

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

Installing the ACC library IPS package on Oracle Solaris 11 systems

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

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

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

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

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

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

4 Install the package.

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

5 Remove the publisher from the system.

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

6 Enable the publishers that were disabled earlier.

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

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

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

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

1 Ensure that the SMF services,

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

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

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

2 Log on to the non-global zone as a superuser.**3** Ensure that the SMF service

```
svc:/application/pkg/zones-proxy-client:default is online inside the  
non-global zone:
```

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

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

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

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

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

7 Install the package.

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

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

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

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

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

- 10 Enable the publishers that were disabled earlier.

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

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

Installing the agent in a VCS environment

Install the agent for PostgreSQL on each node in the cluster.

To install the agent in a VCS environment

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

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

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

```
AIX          cd1/aix/vcs/application/postgresql_agent/  
             vcs_version/version_agent/pkg
```

```
Linux       cd1/linux/generic/vcs/application/postgresql_agent/  
            vcs_version/version_agent/rpms
```

```
Solaris     cd1/solaris/dist_arch/vcs/application/postgresql_agent/  
            vcs_version/version_agent/pkg
```

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

4 Log in as a superuser.

5 Install the package.

```
AIX          # installp -ac -d
              VRTSpgsql.rte.bff VRTSpgsql.rte

Linux        # rpm -ihv \
              VRTSpgsql-AgentVersion-GA_GENERIC.noarch.rpm

Solaris      # pkgadd -d . VRTSpgsql
```

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

See [“Importing the agent types files in a VCS environment”](#) on page 33.

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 `VRTSpgsql.p5p` package from the `pkgs` directory to the system in the `/tmp/install` directory.

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

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

where the publisher name is obtained using the `pkg publisher` command.

3 Add a file-based repository in the system.

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

4 Install the package.

```
# pkg install --accept VRTSpgsql
```

5 Remove the publisher from the system.

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

6 Enable the publishers that were disabled earlier.

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

Installing agent packages on Solaris brand non-global zones

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

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

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

```
# svcs svc:/application/pkg/zones-proxy-client:default
```
- 4 Copy the `VRTSpgsql.p5p` package from the `pkgs` directory to the non-global zone (for example, at the `/tmp/install` directory).
- 5 Disable the publishers that are not reachable, as package install may fail, if any of the already added repositories are unreachable.

```
# pkg set-publisher --disable <publisher name>
```
- 6 Add a file-based repository in the non-global zone.

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

```
# pkg install --accept VRTSpgsql
```
- 8 Remove the publisher on the non-global zone.

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

```
# svcadm clear svc:/application/pkg/system-repository:default
```
- 10 Enable the publishers that were disabled earlier.

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

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

Installing the agent in a Solaris 10 brand zone

To install the PostgreSQL agent in a Solaris 10 brand zone:

- Ensure that the ACC library package, VRTSacclib, is installed in the non-global zone.

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

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

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

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

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

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

```
## Executing postinstall script.  
  
ln: cannot create  
/opt/VRTSagents/ha/bin/PostgreSQL/imf_getnotification: File exists  
  
ln: cannot create /opt/VRTSagents/ha/bin/PostgreSQL/imf_register:  
File exists  
  
or ## Executing postinstall script.  
  
ln: cannot create  
/opt/VRTSagents/ha/bin/PostgreSQL/imf_getnotification: No such file  
or directory  
  
ln: cannot create /opt/VRTSagents/ha/bin/PostgreSQL/imf_register: No  
such file or directory
```

Uninstalling the agent in a VCS environment

You must uninstall the agent for PostgreSQL from a cluster while the cluster is active.

To uninstall the agent in a VCS environment

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

```
# haconf -makerw
```

- 3 Remove all PostgreSQL resources from the cluster. Run the following command to verify that all resources have been removed:

```
# hares -list Type=PostgreSQL
# hares -list Type=PgSQLRep
```

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

```
# hatype -delete PostgreSQL
# hatype -delete PgSQLRep
```

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

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

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

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

Run the following command to uninstall the agent:

```
AIX          # installp -u VRTSpgsql.rte
```

```
Linux       # rpm -e VRTSpgsql
```

```
Solaris    # pkgrm VRTSpgsql
```

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

```
# pkg uninstall VRTSpgsql
```

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
```

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

```
# pkg uninstall VRTSacclib
```

Upgrading the PostgreSQL and the PgSQLRep agents

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

To upgrade the PostgreSQL and the PgSQLRep agents

- 1 Verify the agent version.

```
Linux # rpm -qi VRTSpgsql | grep Version
```

```
Solaris # pkginfo -l VRTSpgsql | grep VERSION
```

The output resembles:

```
Version : 7.0.0.0
```

- 2 Save the VCS configuration.

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

3 Identify the appropriate resource.

```
# hatype -resources PostgreSQL
# hatype -resources PgSQLRep
```

The output resembles:

```
pgsql
pgsqlrep
```

Identify the appropriate service group.

```
# hares -display pgsql | grep Group
# hares -display pgsqlrep | grep Group
```

The output resembles:

```
pgsql          Group      global      PostgreSQL_grp
pgsqlrep       Group      global      PgSQLRep_grp
```

4 Identify the current agent version from the type-level attribute, Version.

```
# hatype -display PostgreSQL | grep Version
# hatype -display PgSQLRep | grep Version
```

5 Freeze the service group.

```
# hagrps -freeze PostgreSQL_grp
# hagrps -freeze PgSQLRep_grp
```

6 Identify whether the PostgreSQL or the PgSQLRep agent is running.

```
# haagent -display PostgreSQL | grep Running
# haagent -display PgSQLRep | grep Running
```

The output resembles:

```
PostgreSQL      Running      Yes
PgSQLRep        Running      Yes
```

7 If the agent is running, stop the agent.

```
# haagent -stop PostgreSQL -force -sys hostname
# haagent -stop PgSQLRep -force -sys hostname
```

8 Verify the status of the agent.

```
# haagent -display PostgreSQL | grep Running
# haagent -display PgSQLRep | grep Running
```

The output resembles:

```
PostgreSQL          Running          No
PgSQLRep            Running          No
```

9 Uninstall the agent by running the following command:

```
AIX          # installp -u VRTSpgsql.rte
Linux        # rpm -e VRTSpgsql
Solaris      # pkgrm VRTSpgsql
```

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

```
# pkg uninstall VRTSpgsql
```

10 Install the latest agent.

See [“Installing the agent in a VCS environment”](#) on page 23.

11 Update the agent type definition.

```
# haconf -makerw

# hatype -modify PostgreSQL SupportedActions PromoteSlaveAction
RewindAction BackupAction

# hatype -modify PostgreSQL ArgList ResLogLevel, State, IState,
PostgreSQLUser, BaseDir, DataDir, EnvFile, HostName, Port,
StartOpts, StopOpts, DBUser, DBName, Table, UseSystemD,
ServiceName, SecondLevelMonitor, MonitorProgram,
MonitorReplication, ClientAddr, SourceConnStr, BackupCmd,
RecoveryFile, RestartdbToRewind, RegistrationOfStandby,
LinkMonitor, AutoTakeOver, SplitTakeOver, DetailedMonitoring,
SwitchMode

# haconf -dump -makero
```

12 Start the database.

```
# haagent -start PostgreSQL -sys hostname  
# haagent -start PgSQLRep -sys hostname
```

The output resembles:

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

13 Verify the status of the agent.

```
# haagent -display PostgreSQL | grep Running  
# haagent -display PgSQLRep | grep Running
```

The output resembles:

```
PostgreSQL      Running      Yes  
PgSQLRep        Running      Yes
```

14 Unfreeze the service group.

```
# hagr -unfreeze PostgreSQL_grp  
# hagr -unfreeze PgSQLRep_grp
```

Configuring the agents for PostgreSQL Database and Replication

This chapter includes the following topics:

- [About configuring the Cluster Server agent for PostgreSQL](#)
- [Importing the agent types files in a VCS environment](#)
- [PostgreSQL agent attributes](#)
- [Executing a customized monitoring program](#)
- [Setting up detail monitoring for the VCS agent for PostgreSQL](#)

About configuring the Cluster Server agent for PostgreSQL

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

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

See [“About sample configurations for the agents for PostgreSQL”](#) on page 71.

Importing the agent types files in a VCS environment

To use the agents for PostgreSQL Database (PostgreSQL) or Replication (PgSQLRep), you must import the agent types file into the cluster. You can import the agent types file using the VCS Java GUI or using the CLI.

To import the PostgreSQL agent types file using the VCS Java GUI

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

VCS 5.x or later	AIX	<code>/etc/VRTSagents/ha/conf/PostgreSQL/</code>
	Linux	<code>PostgreSQLTypes.cf</code>
VCS 5.0	Solaris SPARC	<code>/etc/VRTSagents/ha/conf/PostgreSQL/</code>
		<code>PostgreSQLTypes50.cf</code>
VCS 5.1 or later	Solaris SPARC	<code>/etc/VRTSagents/ha/conf/PostgreSQL/</code>
		<code>PostgreSQLTypes51.cf</code>

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

The PostgreSQL agent type is now imported to the VCS engine. You can proceed to create PostgreSQL Database resources.

Note: In case of a PostgreSQL Replication setup, repeat this procedure with the `/etc/VRTSagents/ha/conf/PgSQLRep/PgSQLRepTypes.cf` file. You can then proceed to create the PgSQLRep resources.

For additional information about using the VCS Java GUI, refer to the *Cluster Server Administrator's Guide*.

To import the PostgreSQL agent types file using the CLI

- 1 If VCS is running, execute
`/etc/VRTSagents/ha/conf/PostgreSQL/PostgreSQLTypes.cmd.`
- 2 If VCS is not running, perform the following steps sequentially:

- Copy the agent types file from the `/etc/VRTSagents/ha/conf/agentTypesFile` directory to the `/etc/VRTSvcs/conf/config` directory.
 The value of `agentTypesFile` depends on the product version and the supported operating systems.

3	VCS 4.x	AIX	<code>/etc/VRTSvcs/conf/sample_PostgreSQL/</code>
		Linux	<code>PostgreSQLTypes.cf</code>
		Solaris	
	VCS 5.x or later	AIX	<code>/etc/VRTSagents/ha/conf/PostgreSQL/</code>
		Linux	<code>PostgreSQLTypes.cf</code>
	VCS 5.0	Solaris SPARC	<code>/etc/VRTSagents/ha/conf/PostgreSQL/</code> <code>PostgreSQLTypes50.cf</code>
	VCS 5.1 or later	Solaris SPARC	<code>/etc/VRTSagents/ha/conf/PostgreSQL/</code> <code>PostgreSQLTypes51.cf</code>

- 4 Include the agent types file in the `main.cf` file.

```
# echo 'include "PostgreSQLTypes.cf"' > main.cf
```

- 5 Start HAD.

The PostgreSQL agent type is now imported to the VCS engine. You can proceed to create PostgreSQL Database resources.

Note: In case of a PostgreSQL Replication setup, repeat this procedure with the `/etc/VRTSagents/ha/conf/PgSQLRep/PgSQLRepTypes.cf` file. You can then proceed to create the PgSQLRep resources.

For additional information about using the VCS CLI, refer to the *Cluster Server Administrator's Guide*.

PostgreSQL agent attributes

Refer to the required and the optional attributes when you configure the agent for PostgreSQL.

Table 3-1 Required attributes for PostgreSQL agent

Attribute	Description
<p>Name: PostgreSQLUser Type: String Dimension: Scalar</p>	<p>The dedicated OS user name that is created when the PostgreSQL server is installed. This user performs all the database server operations, such as start, stop, and monitor. This user name must be identical on all failover nodes.</p> <p>Note: For EnterpriseDB Postgres Advanced Server, the default value of this attribute is <code>enterprisedb</code>. Modify the value of this attribute as required.</p> <p>Default: <code>postgres</code> Example: postgres</p>
<p>Name: DataDir Type: String Dimension: Scalar</p>	<p>Absolute path of the directory that contains the database that this instance of the PostgreSQL server manages. Veritas recommends that you configure this directory on shared storage so that the same copy is available on the failover node.</p> <p>The PostgreSQL user must be the owner of this database directory.</p> <p>Default: No default value Example: /opt/postgres/data</p>
<p>Name: BaseDir Type: String Dimension: Scalar</p>	<p>The installation path of the PostgreSQL database server, where the PostgreSQL executables like <code>pg_ctl</code>, <code>postgres</code>, and so on reside.</p> <p>Default: No default value Example: /usr/bin</p>
<p>Name: HostName Type: String Dimension: Scalar</p>	<p>Virtual host name for this PostgreSQL database instance.</p> <p>Default: No default value Example: web1.veritas.com</p>
<p>Name: Port Type: Integer Dimension: Scalar</p>	<p>Dedicated port number for the PostgreSQL server. The database server is started using the and port number provided.</p> <p>Note: For EnterpriseDB Postgres Advanced Server, the default value of this attribute is 5444. Modify the value of this attribute as required.</p> <p>Default: <code>5432</code> Example: 5212</p>

Table 3-1 Required attributes for PostgreSQL agent (*continued*)

Attribute	Description
<p>Name: MonitorReplication</p> <p>Type: Boolean</p> <p>Dimension: Scalar</p>	<p>Indicates that the agent should monitor the replication link to support the PostgreSQL Streaming Replication feature. When this attribute is set, the agent expects that the setup is that of PostgreSQL Replication and that the PgSQLRep agent is configured.</p> <p>The agent takes the following actions:</p> <ul style="list-style-type: none"> ■ Verifies the replication host from the <code>pg_stat_replication</code> table. ■ Promotes the standby to master and stops the old master, in case of a switchover or a failover operation. ■ Restarts the old master as standby if the <code>RegistrationOfStandby</code> attribute is enabled. <p>Default: 0</p> <p>Example: 1</p>
<p>Name: SwitchMode</p> <p>Type: Boolean</p> <p>Dimension: Scalar</p>	<p>Indicates whether the agent should switch the replication mode from Synchronous to Asynchronous when the replication link is broken. If enabled, the agent also switches the replication mode back to Synchronous when the replication link is restored.</p> <p>Default: 0</p> <p>Example: 1</p>

Table 3-2 Optional attributes for PostgreSQL agent

Attribute	Description
Name: ResLogLevel Type: String Dimension: Scalar	Specifies the logging detail that the agent performs for the resource. The valid values are as follows: <ul style="list-style-type: none"> ■ ERROR: Only logs error messages. ■ WARN: Logs error messages and warning messages. ■ INFO: Logs error messages, warning messages, and informational messages. ■ TRACE: Logs error messages, warning messages, informational messages, and trace messages. TRACE is very verbose and should be used only during initial configuration or for troubleshooting and diagnostic operations. <p>Note: When ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later, use LogDbg instead of ResLogLevel to enable debug logs for ACCLib-based agents. The agent captures the first failure data of the unexpected events and automatically logs debug messages in their corresponding agent log files.</p> Default: INFO Example: ERROR
Name: LogDbg Type: String Dimension: Keylist	When ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later, use the LogDbg resource-type attribute to enable debug logs for ACCLib-based agents. Set LogDbg to DBG_5 to enable debug logs for ACCLib-based agents. By default, when you set LogDbg to DBG_5 , it enables debug logs for all PostgreSQL resources in the cluster. If you need to enable debug logs for a specific PostgreSQL resource, override the LogDbg attribute. For details on how to use the LogDbg attribute, see the <i>Cluster Server Administrator's Guide</i> . Default: No default value

Table 3-2 Optional attributes for PostgreSQL agent (*continued*)

Attribute	Description
<p>Name: EnvFile</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>Full path of the file name to source to set the environment before the PostgreSQL commands are executed.</p> <p>Veritas recommends storing the file on a shared disk. The <code>ksh</code>, <code>sh</code>, and <code>csh</code> shell environments are supported.</p> <p>This attribute is required only if second-level monitoring is enabled.</p> <p>Default: No default value</p> <p>Example: <code>/postgres/data/pg.env</code></p>
<p>Name: MonitorProgram</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>Absolute path name of an external, user-supplied monitor executable.</p> <p>For details on setting this attribute: See "Executing a customized monitoring program" on page 45.</p> <p>Default: No default value</p> <p>Example 1: <code>ServerRoot/bin/myMonitor.pl</code></p> <p>Example 2: <code>ServerRoot/bin/myMonitor.sh arg1 arg2</code></p>

Table 3-2 Optional attributes for PostgreSQL agent (*continued*)

Attribute	Description
<p>Name: SecondLevelMonitor</p> <p>Type: Integer</p> <p>Dimension: Scalar</p>	<p>Used to enable second-level monitoring and specify how often it is run. Second-level monitoring is a deeper, more thorough state check of the configured PostgreSQL server instance. The numeric value specifies how often the second-level monitoring routines are run.</p> <p>For example, if MonitorInterval is set to 60 seconds and SecondLevelMonitor is set to 100, the second-level check is performed only after every 100 minutes.</p> <p>To provide maximum flexibility, the value set is not checked for an upper limit. You can set the second-level check to occur once a month, if that is appropriate for your environment.</p> <p>Set the value of this attribute to a large number only after careful consideration of these implications.</p> <p>See "Setting up detail monitoring for the VCS agent for PostgreSQL" on page 45.</p> <p>Note: The SecondLevelMonitor attribute is applicable to VCS versions earlier than VCS 5.1 SP1 with PostgreSQL agent versions earlier than 5.1.1.0. From VCS version 5.1 SP1 with PostgreSQL agent version 5.1.1.0 onwards, the SecondLevelMonitor attribute is deprecated. Instead, a resource type level attribute LevelTwoMonitorFreq should be used to specify the frequency of in-depth monitoring.</p> <p>Default: 0</p> <p>Example: 1</p>
<p>Name: UseSystemD</p> <p>Type: Boolean</p> <p>Dimension: Scalar</p>	<p>SystemD is a system and a service manager for Linux operating systems. It helps manage applications across Linux distributions that support the SystemD feature. When UseSystemD is set to 1 on the SystemD-enabled RHEL and SLES platforms, the PostgreSQL resource uses the PostgreSQL service file for the online and the offline operations. The PostgreSQL resource comes online as a service in system.slice. When this attribute is set to 0, typical online and offline functions start and stop the resource in user.slice.</p> <p>Default: 0</p> <p>Example: 1</p>

Table 3-2 Optional attributes for PostgreSQL agent (*continued*)

Attribute	Description
Name: ServiceName Type: String Dimension: Scalar	Used to provide support for SystemD-enabled RHEL and SLES platforms. This attribute defines the name of the service that is used to start and stop the PostgreSQL application. If UseSystemD is set to 1, you must specify a value for ServiceName. Default: No default value Example: postgresql-9.4
Name: LevelTwoMonitorFreq Type: Integer Dimension: Scalar	Specifies the frequency at which the agent for this resource type must perform second-level or detailed monitoring. You can also override the value of this attribute at the resource level. The value indicates the number of monitor cycles after which the agent monitors the PostgreSQL server in detail. For example, the value 5 indicates that the agent monitors the PostgreSQL server in detail after every five online monitor intervals. Note: This attribute is applicable to VCS version 5.1 SP1 or later with PostgreSQL agent version 5.1.1.0 or later. If VCS version is earlier than 5.1 SP1 and PostgreSQL agent version is earlier than 5.1.1.0, you must use the SecondLevelMonitor attribute. Set LevelTwoMonitorFreq to the same value as SecondLevelMonitor, if both the following conditions are true: <ul style="list-style-type: none"> ■ You upgraded the VCS version to VCS 5.1 SP1 or later and the PostgreSQL agent version to 5.1.1.0 (or later). ■ You had enabled detail monitoring in the previous version. Default: 0
Name: StartOpts Type: String Dimension: Scalar	The startup options for the <code>pg_ctl</code> command. Example: -l logfile

Table 3-2 Optional attributes for PostgreSQL agent (*continued*)

Attribute	Description
<p>Name: StopOpts</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>Shutdown options for the PostgreSQL database server. You can use this attribute to specify a shutdown mode, such as <code>-m fast</code>, which does not wait for clients to disconnect.</p> <p>If this attribute is not specified, the agent stops the database server with the default <code>-m smart</code> shutdown mode.</p> <p>If the database is configured for replication, you must specify the <code>-t TIMEOUT_SECS</code> option. You need not specify the <code>-w</code> option, because the relevant value is already passed.</p> <p>For information about shutdown options, see <code>postgres --help</code>. These options are used in the <code>pg_ctl</code> command, which is used when the <code>postgres</code> process is stopped.</p> <p>Default: No default value</p> <p>Example: -m fast</p>
<p>Name: DBUser</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>A valid database user name that is used to run queries on the database during detail monitoring. This user must have privileges to run queries on or to update the table that is created for detail monitoring.</p> <p>Default: No default value</p> <p>Example: postgres</p>
<p>Name: DBName</p> <p>Type:</p> <p>Dimension:</p>	<p>A valid database name in which the table is created for detail monitoring.</p> <p>Default: No default value</p> <p>Example: postgres</p>
<p>Name: Table</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>A valid database table in the <code>\$DBUser</code> schema on which the query is executed during detail monitoring. The table should contain a single field <code>TSTAMP</code> with datatype <code>DATE</code>.</p> <p>Default: No default value</p> <p>Example: vcsslm</p>
<p>Name: SourceConnStr</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>Connection string that the <code>RewindAction</code> action entry point uses to support the PostgreSQL Streaming Replication feature.</p> <p>Default: No default value</p> <p>Example: host=10.209.69.168 port=5432 user=postgres dbname=postgres</p>

Table 3-2 Optional attributes for PostgreSQL agent (*continued*)

Attribute	Description
<p>Name: BackupCmd</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>Complete base backup command that is used to backup the database instance. The agent uses this value to support the PostgreSQL Streaming Replication feature.</p> <p>Note: If third-party tools are used for PostgreSQL backup, specify the full path of the backup command, including the command options to be used.</p> <p>Default: No default value</p> <p>Example: <code>/usr/pgsql-10/bin/pg_basebackup -R -X stream -D /var/lib/pgsql/data -U postgres -h 10.209.69.168</code></p>
<p>Name: RecoveryFile</p> <p>Type: String</p> <p>Dimension: Scalar</p>	<p>Full path of the <code>recovery.conf</code> file. This value is applicable in the context of the replication support for PostgreSQL. The agent copies this file to the data directory when it configures a standby instance.</p> <p>For PostgreSQL 12 and later, specify the full path of the <code>postgresql.auto.conf</code> file or any other configuration file that the agent should copy to the data directory on standby node.</p> <p>Default: No default value</p> <p>Example 1: <code>/var/lib/pgsql/backup/recovery.conf</code></p> <p>Example 2: <code>/var/lib/pgsql/backup/postgresql.auto.conf</code></p>
<p>Name: RestartdbToRewind</p> <p>Type: Boolean</p> <p>Dimension: Scalar</p>	<p>Indicates whether to restart the database if the <code>pg_rewind</code> command fails. The agent uses this value to support the PostgreSQL Streaming Replication feature. If the database is not gracefully shut down and if the value of this attribute is set to 1, the agent starts and stops the database before it runs the <code>pg_rewind</code> command again.</p> <p>Note: This attribute must be set to 1 for the synchronous mode of replication.</p> <p>Default: 0</p> <p>Example: 1</p>

Table 3-2 Optional attributes for PostgreSQL agent (*continued*)

Attribute	Description
Name: RegistrationOfStandby Type: Boolean Dimension: Scalar	Indicates whether to re-register the old master as the standby. The agent uses this value to support the PostgreSQL Streaming Replication feature. In case of a failover or a switchover operation, the PgSQLRep resource refers to this attribute. If its value is set to 1, the agent re-registers the old master as standby. Default: 0 Example: 1
Name: LinkMonitor Type: Boolean Dimension: Scalar	Specifies whether the agent should check the state of the replication while the resource comes online on the master node. If set to 1, the agent checks the state of the replication and brings the resource online only if the state is UP. This attribute is used only on the node where the database is already in the master mode. Default: 0 Example: 1
Name: AutoTakeOver Type: Boolean Dimension: Scalar	Indicates whether the agent should bring the standby database replication resource online when the primary database is not available. If set to 0, the agent does not bring the resource online when the primary database is not available. Default: 0 Example: 1
Name: SplitTakeOver Type: Boolean Dimension: Scalar	Indicates whether the agent should bring the standby database replication resource online when the replication link is not in the healthy state. If set to 0, the agent does not bring the resource online when the replication is not in the healthy state. Default: 0 Example: 1
Name: DetailedMonitoring Type: Boolean Dimension: Scalar	Indicates whether the agent should log the monitoring details during each monitor cycle. Default: 0 Example: 1

Table 3-3 Required attributes for PgSQLRep agent

Attribute	Description
Name: ResLogLevel Type: String Dimension: Scalar	Specifies the level of logging detail that the agent provides for the resource. The valid values are as follows: <ul style="list-style-type: none"> ■ ERROR: Only logs error messages. ■ WARN: Logs error messages and warning messages. ■ INFO: Logs error messages, warning messages, and informational messages. ■ TRACE: Logs error messages, warning messages, informational messages, and trace messages. TRACE is very verbose and should be used only during initial configuration or for troubleshooting and diagnostic operations. <p>Note: When ACCLib version is 6.2.0.0 or later and VCS version is 6.2 or later, use LogDbg instead of ResLogLevel to enable debug logs for ACCLib-based agents. The agent captures the first failure data of the unexpected events and automatically logs debug messages in their corresponding agent log files.</p> Default: INFO Example: ERROR
Name: PgSQLResource Type: String Dimension: Scalar	Specifies the name of the PostgreSQL Database resource. The PostgreSQL attribute values are fetched from this resource. Default: No default value Example: pg_res

Table 3-4 Internal attribute for PgSQLRep agent

Attribute	Description
Name: WalSenderPid Type: Integer Dimension: Scalar	Used to store the process ID of the Wal Sender process. Do not edit this value.

Note: For information about the additional attributes of the agent for PostgreSQL when IMF is enabled:

See [“Attributes that enable IMF”](#) on page 50.

Executing a customized monitoring program

You can configure the monitor function to execute MonitorProgram. MonitorProgram is a custom monitor utility to perform a user-defined PostgreSQL server state check.

The utility is executed in the context of the UNIX user that is defined in the PostgreSQLUser attribute.

The monitor operation executes MonitorProgram if:

- The MonitorProgram attribute value is set to a valid executable utility.
- The first-level process check indicates that the PostgreSQL server instance is online.
- The LevelTwoMonitorFreq attribute is set to 1 and the second-level check returns the server state as ONLINE.

Or

- The LevelTwoMonitorFreq attribute is set to greater than 1, but the second-level check is deferred for this monitoring cycle.

The monitor operation interprets the program exit code as follows:

110 or 0	PostgreSQL server is online
100 or 1	PostgreSQL server is offline
Any other value	PostgreSQL server state is unknown

Setting up detail monitoring for the VCS agent for PostgreSQL

The Cluster Server agent for PostgreSQL provides the following two levels of application monitoring:

- Primary (basic monitoring)
In the basic monitoring mode, the agent monitors the PostgreSQL processes to verify that they are continuously active.
- Secondary (detail monitoring)
In the detail monitoring mode, the agent executes the psql SELECT statement to monitor the health of the database.
You can use the agent's detail monitoring capability to monitor the status of a database and listener and to increase the confidence in their availability.

Note: Disable detail monitoring before undertaking any database maintenance that involves disabling database access to external users.

Detail monitoring for a PostgreSQL resource verifies whether a database is ready for transactions by performing a SELECT transaction against a table within the database. This SELECT statement fetches the time-stamp from the table created for detail monitoring.

Ensure the following before you set up and enable detail monitoring:

- the agent is running satisfactorily at the basic level of monitoring.
- you have created a test table (with a timestamp) in the PostgreSQL database.

The example to set up detail monitoring shows how to create and test a table for use by detail monitoring, and how to enable detail monitoring.

To set up detail monitoring for PostgreSQL

- 1 Make the VCS configuration writable.

```
# haconf -makerw
```

- 2 Freeze the service group to avoid automated actions by VCS caused by an incomplete reconfiguration.

```
# hagr -freeze serviceName
```

- 3 Log on as a PostgreSQL user.

```
$ su - ownerName
```

- 4 Start the psql utility, and as the database administrator, run the following command to set up a database table:

```
$ psql -h hostName -p portNumber -U adminUserName  
-d databaseName
```

Enter the password when prompted.

- 5 As the database administrator, issue the following statements at the psql prompt to create the test table:

```
CREATE USER <USER> WITH PASSWORD '<PASSWORD>';  
CREATE table <TABLE>(tstamp timestamp);  
GRANT SELECT,UPDATE ON <TABLE> TO <USER>;  
INSERT INTO <TABLE> VALUES (CURRENT_TIMESTAMP);  
\q
```

- 6** If the `pg_hba.conf` file is configured for password-based authentication methods, such as `md5` or `password`, create a `.pgpass` file in the format:

```
hostname:port:database:username:password.
```

Here, use the username and password that you created while creating the test table in step 5.

- 7** Make sure that the file permissions for the `.pgpass` file are 0600 or less.
- 8** Place the `.pgpass` file in the shared file system or on the local file systems. The `.pgpass` file must be in the same location on all nodes in the cluster.
- 9** Create a `.env` file and enter the following text in it:

```
export PGPASSFILE=pgpassFilePath
```

Place the `.env` file in the shared disk and provide the path of this `.env` file in the `EnvFile` agent attribute.

- 10** To test the database table for use, run the following command:

```
$ psql -A -t -h hostName -p portNumber -U userName
-d databaseName -c "select tstamp from TableName;"
```

- 11** Enable detail monitoring for the PostgreSQL resource using the following VCS commands:

```
# hares -modify PostgreSQLResource DBUser User
# hares -modify PostgreSQLResource DBName DBName
# hares -modify PostgreSQLResource Table Table
# hares -modify PostgreSQLResource SecondLevelMonitor 1
# haconf -dump -makero
# hagr -unfreeze service_group
```

You can also use Cluster Manager (Java Console) or Veritas Infoscale Operations Manager to set these attributes.

Enabling the PostgreSQL and the PgSQLRep agents to support IMF

This chapter includes the following topics:

- [About Intelligent Monitoring Framework](#)
- [Agent functions for the IMF functionality](#)
- [Attributes that enable IMF](#)
- [Before you enable the agent to support IMF](#)
- [Enabling the agent to support IMF](#)
- [Disabling intelligent resource monitoring](#)
- [Sample IMF configurations](#)

About Intelligent Monitoring Framework

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

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

You can enable the PostgreSQL agent for IMF, provided the following software versions are installed:

- Cluster Server (VCS) 5.1 SP1 or later
- Cluster Server agent for PostgreSQL version 5.1.0.0 or later

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

Benefits of IMF

IMF offers the following benefits:

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

Agent functions for the IMF functionality

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

imf_init

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

imf_getnotification

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

imf_register

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

Attributes that enable IMF

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

IMF

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

This attribute includes the following keys:

Mode

Define this attribute to enable or disable intelligent resource monitoring.

The valid values are as follows:

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

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

Type and dimension: integer-association

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

MonitorFreq

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

Default: 1

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

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

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

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

RegisterRetryLimit

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

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

Default: 3.

IMFRegList

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

Enable IMF for the PgSQLRep agent only when the `SwitchMode` attribute of the PostgreSQL agent is set to 1. Then, set the value of this attribute to `WalSenderPid`.

Type and dimension: string-vector

Default: No default value

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

Before you enable the agent to support IMF

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

Enabling the agent to support IMF

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

- **AgentFile**: Set the AgentFile attribute to **Script51Agent** or **Script60Agent** as appropriate for your agent version
- **IMF Mode**: Set the IMF Mode attribute to **2**
- **IMFRegList**: Update the IMFRegList attribute

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

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

```
# cd /opt/VRTSagents/ha/bin/<resourceType>
# ln -s /opt/VRTSamf/imf/imf_getnotification imf_getnotification
# ln -s /opt/VRTSagents/ha/bin/<resourceType>/monitor imf_register
```

If VCS is in a running state

To enable the resource for IMF when VCS is in a running state

- 1 Make the VCS configuration writable.

```
# haconf -makerw
```

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

```
# hatype -modify <resourceType> AgentFile\
/opt/VRTSvc/bin/Script51Agent
```

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

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

# haattr -add -static <resourceType> IMFRegList -string -vector
```

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

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

```
# hatype -modify <resourceType> IMF Mode num MonitorFreq num
RegisterRetryLimit num
```

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

```
# hatype -modify <resourceType> IMF Mode 2 MonitorFreq 5 \
RegisterRetryLimit 3
```

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

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

```
# hatype -modify PostgreSQL IMFRegList BaseDir DataDir
PostgreSQLUser

# hatype -modify PostgreSQLRep IMFRegList BaseDir DataDir
PostgreSQLUser
```

- 6 Save the VCS configuration.

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

- 7 If the agent is running, restart the agent.

For information on the commands you can use to restart the agent, see [Restarting the agent](#).

Restarting the agent

To restart the agent:

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

```
# haagent -stop <resourceType> -force -sys <systemName>
```

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

- 2 Run the following command to start the agent:

```
# haagent -start <resourceType> -sys <systemName>
```

If VCS is not in a running state

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

- 1 Update the AgentFile attribute.

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

- 2 Update the IMF attribute.

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

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

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

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

- 3 Update the IMFRegList attribute.

For PostgreSQL:

```
static str IMFRegList[] = { BaseDir, DataDir,  
PostgreSQLUser }
```

For PgSQLRep:

```
static str IMFRegList[] = { WalSenderPid }
```

Disabling intelligent resource monitoring

To disable intelligent resource monitoring

- 1 Make the VCS configuration writable.

```
# haconf -makerw
```

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

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

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

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

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

- 4 Save the VCS configuration.

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

Sample IMF configurations

An example of a type definition file for a PostgreSQL agent that is IMF-enabled is as follows. In this example, the IMF-related attributes are set to the following values:

```
AgentFile /opt/VRTSvcs/bin/Script51Agent
IMF{ } { Mode=2, MonitorFreq=5, RegisterRetryLimit=3 }
IMFRegList[] { BaseDir DataDir PostgreSQLUser }
LevelTwoMonitorFreq 25

type PostgreSQL (
    static str AgentDirectory = "/opt/VRTSagents/ha/bin/PostgreSQL"
    static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"
    static str ArgList[] = { ResLogLevel, State, IState, PostgreSQLUser,
        BaseDir, DataDir, EnvFile, HostName, Port, StartOpts, StopOpts,
        DBUser, DBName, Table, UseSystemD, ServiceName, SecondLevelMonitor,
        MonitorProgram, MonitorReplication, ClientAddr, SourceConnStr,
        BackupCmd, RecoveryFile, RestartdbToRewind, RegistrationOfStandby }
    static boolean AEPTIMEOUT = 1
    str ResLogLevel = INFO
    str PostgreSQLUser = postgres
    str HostName
    str EnvFile
    int Port = 5432
```

```
str BaseDir
str DataDir
str StartOpts
str DBUser
str DBName
str Table
boolean UseSystemD = 0
str ServiceName
int SecondLevelMonitor
str MonitorProgram
boolean MonitorReplication = 0
str ClientAddr
str SourceConnStr
str BackupCmd
str RecoveryFile
boolean RestartdbToRewind = 0
boolean RegistrationOfStandby = 0
boolean SwitchMode = 0
)
```

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

```
PostgreSQL pg-sg (
  ResLogLevel = TRACE
  HostName = localhost
  EnvFile = "/server/pg.env"
  BaseDir = "/usr/local/pgsql/bin"
  DataDir = "/server"
  StartOpts = "-l /server/logfile"
  DBUser = vcs_user
  DBName = vcs_slm
  Table = vcs_slm
  UseSystemD = 1
  ServiceName = "postgresql-9.4"
)
```

Configuring the service groups for PostgreSQL using the CLI

This chapter includes the following topics:

- [About configuring service groups for PostgreSQL](#)
- [Before configuring the service groups for PostgreSQL](#)
- [PostgreSQL entities in a clustered environment](#)
- [Virtualizing PostgreSQL](#)
- [Creating service groups for PostgreSQL under Solaris non-global zones](#)
- [Configuring service groups for PostgreSQL Replication](#)
- [Configuring PostgreSQL nofailover trigger](#)

About configuring service groups for PostgreSQL

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

You can configure the service groups using one of the following:

- The Cluster Manager (Java console)
- Veritas Infoscale Operations Manager
- The command line

Before configuring the service groups for PostgreSQL

Before you configure the PostgreSQL service group, you must:

- Verify that the Cluster Server components are installed and configured on all nodes in the cluster where you will configure the service group.
For more information on installing the components, refer to the *InfoScale Availability Installation Guide*.
- Verify that the Cluster Server agent for PostgreSQL is installed on all nodes in the cluster.
See “[Installing the agent in a VCS environment](#)” on page 23.

PostgreSQL entities in a clustered environment

A service group is a logical setup containing all resources that can support a PostgreSQL instance in a clustered environment.

The required resources are as follows.

Disk group	<p>Contains a volume and a file system, which is a mount resource containing the PostgreSQL installation files.</p> <p>Use the DiskGroup resource type to create this resource. Create the disk group from the shared disk so that you can import the group into any system in the cluster.</p>
Mount	<p>Mounts, monitors, and unmounts the file system that is dedicated to the PostgreSQL installation files.</p> <p>Use the Mount resource type to create this resource.</p>
Network interface	<p>Monitors the network interface card through which the PostgreSQL instance communicates with other services.</p> <p>Use the NIC resource type to create this resource.</p>
Virtual IP	<p>Configures the virtual IP address dedicated to the PostgreSQL instance. The external services, programs, and clients use this address to communicate with this instance.</p> <p>Use the IP resource type to create this resource.</p>
PostgreSQL server	<p>Starts, stops, and monitors the PostgreSQL server instance.</p> <p>Use the PostgreSQL server resource type to create this resource.</p>

Virtualizing PostgreSQL

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

Review the following basic notes for virtualization:

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

Creating service groups for PostgreSQL under Solaris non-global zones

To configure zones on each cluster node

- 1 Set up the non-global zone configuration.

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

For example:

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

- 2 Verify the non-global zone configuration.

```
hazoneverify servicegroup_name
```

- 3 Whenever you make a change that affects the zone configuration, run the `hazonesetup` command to reconfigure the zones in VCS.
- 4 Make sure that the zone configuration files are consistent on all nodes at all times. The file is located at `/etc/zones/zone_name.xml`.
- 5 Make sure that the application is identical on all nodes. If you update the application configuration on one node, apply the same updates to all nodes.
- 6 Configure the service groups for PostgreSQL.

Configuring service groups for PostgreSQL Replication

Note the following considerations before you configure a service group for a PostgreSQL Replication setup:

- The service group for the PgSQLRep resource can only be the parent service group for the PostgreSQL resource.
- The OnlineTimeout attribute value of the PostgreSQL agent type should be large enough to accommodate the replication time taken for either of the following:
 - Starting the PostgreSQL instance
 - Performing the takeover operation

If the replication delays the starting of the PostgreSQL instance and the time exceeds the OnlineTimeout value, you must start the PostgreSQL instance outside VCS.

If the replication delays the takeover operation and the time exceeds the OnlineTimeout value, you must perform the takeover operation manually.

Supported service group configurations

Figure 5-1 Service group dependency for a failover cluster

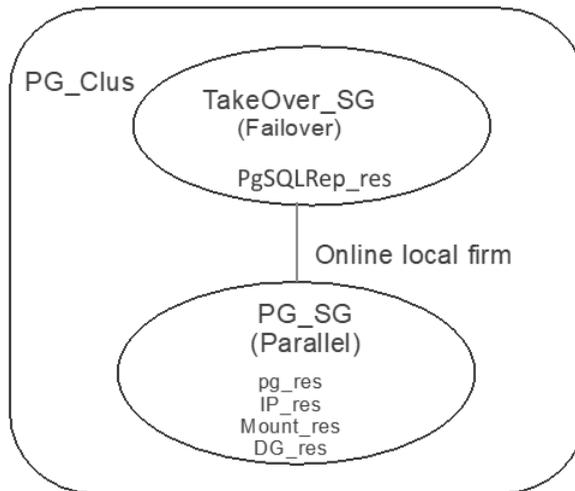


Figure 5-2 Service group dependencies in case of two n-node global clusters with Master instance in one cluster and Slave instance in the other

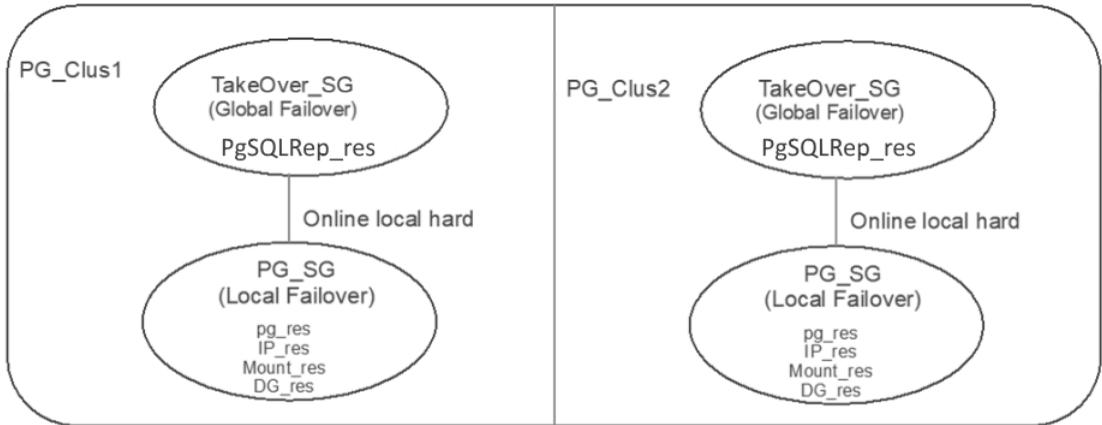
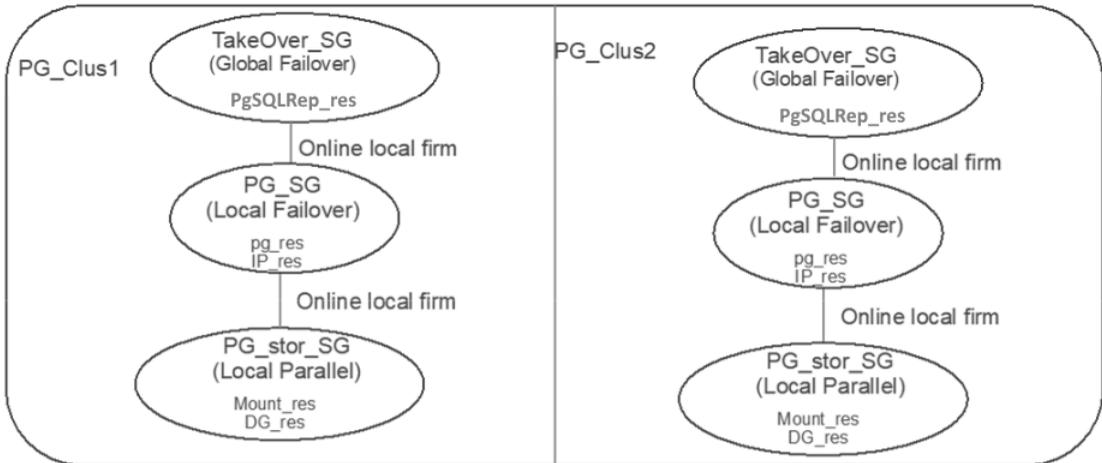


Figure 5-3 Service group dependencies in case of two n-node global clusters with Veritas Cluster File System (CFS) as the underlying storage management component



Perform the following procedure at each site in a global cluster.

Note: For details on configuring global clusters with the underlying storage components, refer to the relevant documents:

- *Cluster Server Configuration and Upgrade Guide*
 - *Storage Foundation and High Availability Configuration and Upgrade Guide*
 - *Storage Foundation Cluster File System High Availability Configuration and Upgrade Guide*
-

To add a failover service group for PostgreSQL on one a site in a global cluster

- 1 Create a service group for PostgreSQL.

For example: `# hagrps -add PG_SG`

For details on creating a service group, refer to the *Cluster Server Administrator's Guide*.

- 2 Add systems to the service group by modifying the SystemList attribute.

For example:

```
# hagrps -modify PG_SG SystemList sys1 0 sys2 1
```

- 3 Create resources for the PostgreSQL instance in the service group.

For example: `# hares -add pg PostgreSQL PG_SG`

- 4 Modify the resource attributes as appropriate for your environment.

See [“PostgreSQL agent attributes”](#) on page 34.

- 5 Create a failover service group for the PgSQLRep resource.

For example: `# hagrps -add TakeOver_SG`

- 6 Add systems to the PgSQLRep service group by modifying the SystemList attribute.

```
# hagrps -modify TakeOver_SG SystemList sys1 0 sys2 1
```

Configure the PgSQLRep service group.

- 7 Create a PgSQLRep resource in the service group.

For example: `# hares -add pgreg PgSQLRep TakeOver_SG`

Note: You must set the RestartLimit attribute for the configured pgreg resource. You can override the attribute values as follows:

```
/opt/VRTSvcs/bin/hares -override pgreg RestartLimit
```

```
/opt/VRTSvcs/bin/hares -modify pgreg RestartLimit 2
```

- 8 Specify the appropriate dependencies between the PG_SG and the TakeOver_SG groups.

For example: `# hagr -link TakeOver_SG PG_SG online local hard`

Verify the group dependencies.

```
# hagr -dep
```

```
# Parent      Child      Relationship
TakeOver_SG   PG_SG      online local hard
```

Configuring PostgreSQL nofailover trigger

You must configure this trigger in case of a setup where two n-node GCO clusters are configured with CFS. In such a setup, when the master PostgreSQL service group fails on all the nodes of a cluster, the nofailover trigger gets executed, which triggers the takeover operation.

Note: The nofailover trigger must be configured for the PostgreSQL service group.

To configure the nofailover trigger

- 1 Copy the `nofailover` script from

```
/etc/VRTSagents/ha/conf/PostgreSQL/nofailover to
```

```
/opt/VRTSvcs/bin/triggers/nofailover.
```

- 2 Run the following command on any node in the cluster:

```
# hagr -modify serviceName TriggersEnabled -add NOFAILOVER
```

For example:

```
# hagr -modify pgsg TriggersEnabled -add NOFAILOVER
```

To disable the nofailover trigger

- ◆ Run the following command on any node in the cluster:

```
# hagrps -modify serviceName TriggersEnabled -delete  
NOFAILOVER
```

For example:

```
# hagrps -modify pgsg TriggersEnabled -delete NOFAILOVER
```

Troubleshooting the agent for PostgreSQL

This chapter includes the following topics:

- [Using the correct software and operating system versions](#)
- [Meeting prerequisites](#)
- [Verifying virtualization](#)
- [Starting the PostgreSQL server outside a cluster](#)
- [Reviewing error log files](#)
- [Troubleshooting the configuration for IMF](#)

Using the correct software and operating system versions

Ensure that you use correct software and operating system versions.

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

Meeting prerequisites

Before installing the agent for PostgreSQL, ensure that the following prerequisites are met.

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

See [“Before you install the Cluster Server agent for PostgreSQL”](#) on page 19.

Verifying virtualization

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

See [“Virtualizing PostgreSQL ”](#) on page 59.

Starting the PostgreSQL server outside a cluster

If you face problems while working with a resource, you must disable the resource within the cluster framework. A disabled resource is not under the control of the cluster framework, and so you can test the PostgreSQL database server independent of the cluster framework. Refer to the cluster documentation for information about disabling a resource.

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

- Starting the PostgreSQL server

To start the PostgreSQL server outside cluster, execute:

```
$ baseDirectory/pg_ctl start -w -D dataDirectory
-o "-p portNumber -h hostName" startOptions
```

- Stopping the PostgreSQL server

To stop the PostgreSQL server outside cluster, execute:

```
$ BaseDir/pg_ctl stop -w -D DataDir
```

- Monitoring the PostgreSQL server

First verify that the PostgreSQL processes are running as PostgreSQLUser. The default value is postgres.

- The agent uses the following monitor command to verify that the PostgreSQL server is up.

```
$ baseDirectory/pg_ctl status -D dataDirectory
```

Try executing this command manually to verify if the PostgreSQL server is up.

Reviewing error log files

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

The common reasons for issues are as follows:

Incorrect port, environment or parameter settings	<p>Verify that ports have been properly configured and declared. Typically, ports from 1 through 1024 are reserved for the superuser.</p> <p>Ensure that parameters to the agent are correctly defined.</p>
Expired licenses	<p>Check the application log files for any error messages related to expired licenses.</p> <p>Ensure that the license keys/files have been placed at the appropriate location, as needed by the application.</p>
Broken symlinks, missing files, and libraries	<p>Verify your installation.</p> <p>Make sure that nothing is broken, and all dependencies for the executables are met.</p>
Insufficient disk space or system parameters	<p>Ensure that the file-system has sufficient space for creation of temporary files that the application might need.</p> <p>Verify that the kernel has been tuned for sufficient IPC resources, file descriptors and meets the hardware requirement. Consult your product documentation for these details.</p>

Consult your application expert if needed.

Using trace level logging

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

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

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

To localize ResLogLevel attribute for a resource

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

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

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

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

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

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

- 8 Save the configuration changes.

```
# haconf -dump
```

- 9 Review the contents of the log file.

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

You can also contact Veritas support for more help.

To enable debug logs for all resources of type PostgreSQL

- ◆ Enable the debug log.

```
# hatype -modify PostgreSQL LogDbg DBG_5
```

To override the LogDbg attribute at resource level

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

```
# hares -override PostgreSQL LogDbg
```

```
# hares -modify PostgreSQL LogDbg DBG_5
```

Troubleshooting the configuration for IMF

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

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


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

The `MonitorMethod` attribute specifies the monitoring method that the agent uses to monitor the resource:

 - Traditional—Poll-based resource monitoring
 - IMF—Intelligent resource monitoring
- You can use the `amfstat` command to see a list of registered PIDs for a PostgreSQL resource.

The output of the `ps -ef` command for the PostgreSQL process.

```
$ ps -ef | grep postgres postgres 4883 1 0 Aug16 ?
00:00:00 /usr/local/pgsql/bin/postgres -D /d01/pgsql/data -p 5432 -h
pg-server postgres 4893 4883 0 Aug16 ? 00:00:00 postgres:
logger process postgres 4897 4883 0 Aug16 ? 00:00:01 postgres:
writer process postgres 4898 4883 0 Aug16 ? 00:00:01 postgres:
wal writer process postgres 4899 4883 0 Aug16 ? 00:00:01 postgres:
autovacuum launcher process postgres 4900 4883 0 Aug16 ? 00:00:05
postgres:
stats collector process root 20890 3877 0 11:44 pts/1 00:00:00
grep postgres
```

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

```
# amfstat

AMF Status Report

Registered Reapers (1):
=====
RID      PID      MONITOR  TRIGG    REAPER
0        19219    1        0        PostgreSQL

Process ONLINE Monitors (1):
=====
RID      R_RID    PID      GROUP
1        0        4883     pg-server
```

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

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

For more information about the `ResLogLevel` attribute, See [“PostgreSQL agent attributes”](#) on page 34.

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

```
# amfconfig -p dbglog
```

Known issues

This release of the agent for PostgreSQL has the following known issues:

Problem

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

Description

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

This results in an error message from the engine log.

Workaround

It is safe to ignore this error message.

Sample Configurations

This appendix includes the following topics:

- [About sample configurations for the agents for PostgreSQL](#)
- [Sample agent type definition for PostgreSQL](#)
- [Sample configuration files](#)
- [Sample service group configurations for PostgreSQL](#)

About sample configurations for the agents for PostgreSQL

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

Sample agent type definition for PostgreSQL

VCS 5.1 or later

```
type PostgreSQL (  
    static str AgentDirectory = "/opt/VRTSagents/ha/bin/PostgreSQL"  
    static str AgentFile = "/opt/VRTSvcs/bin/Script50Agent"  
    static keylist SupportedActions = { PromoteSlaveAction,  
        RewindAction, BackupAction }  
    static str ArgList[] = { ResLogLevel, State, IState,  
        PostgreSQLUser, BaseDir, DataDir, EnvFile, HostName, Port,  
        StartOpts, StopOpts, DBUser, DBName, Table, UseSystemD,  
        ServiceName, SecondLevelMonitor, MonitorProgram,
```

```
        MonitorReplication, ClientAddr, SourceConnStr, BackupCmd,
        RecoveryFile, RestartdbToRewind, RegistrationOfStandby }
static boolean AEPTIMEOUT = 1
str ResLogLevel = INFO
str PostgreSQLUser = postgres
str HostName
str EnvFile
int Port = 5432
str BaseDir
str DataDir
str StartOpts
str StopOpts
str DBUser
str DBName
str Table
boolean UseSystemD = 0
str ServiceName
int SecondLevelMonitor
str MonitorProgram
boolean MonitorReplication = 0
str ClientAddr
str SourceConnStr
str BackupCmd
str RecoveryFile
boolean RestartdbToRewind = 0
boolean RegistrationOfStandby = 0
boolean SwitchMode = 0
)

type PgSQLRep (
    static str AgentDirectory = "/opt/VRTSagents/ha/bin/PgSQLRep"
    static str AgentFile = "/opt/VRTSvcs/bin/Script60Agent"
    static keylist SupportedActions = { RegisterStandby }
    static str ArgList[] = { ResLogLevel, State, IState, PostgreSQLResource }
    static boolean AEPTIMEOUT = 1
    str ResLogLevel = INFO
    str PostgreSQLResource
    temp int WalSenderPid
)
```

VCS 5.0 or later

```
type PostgreSQL (
    static boolean AEPTIMEOUT = 1
```

```
static str AgentFile = "/opt/VRTSvc/bin/Script50Agent"
static str AgentDirectory = "/opt/VRTSagents/ha/bin/PostgreSQL"
static str ArgList[] = { ResLogLevel, State, IState,
    PostgreSQLUser, HostName, Port, BaseDir, DataDir, StartOpts,
    StopOpts, DBUser, DBName, Table, SecondLevelMonitor,
    MonitorProgram, MonitorReplication, ClientAddr,
    SourceConnStr, BackupCmd, RecoveryFile, RestartdbToRewind,
    RegistrationOfStandby }
str ResLogLevel = INFO
str PostgreSQLUser = postgres
str HostName
str EnvFile
int Port = 5432
str BaseDir
str DataDir
str StartOpts
str StopOpts
str DBUser
str DBName
str Table
int SecondLevelMonitor = 0
str MonitorProgram
boolean MonitorReplication = 0
str ClientAddr
str SourceConnStr
str BackupCmd
str RecoveryFile
boolean RestartdbToRewind = 0
boolean RegistrationOfStandby = 0
boolean SwitchMode = 0
)
```

Sample configuration files

Snippet of a PostgreSQL resource from a sample configuration file `main.cf`:

```
PostgreSQL pg_server_1 (
    Critical = 1
    ResLogLevel = TRACE
    BaseDir = "/usr/bin"
    DataDir = "/opt/postgres/data"
    HostName = pgserver
    Port = 24321
```

```
StartOpts = "-l /tmp/pglog"
DBUser=dbuser
DBName=dbname
Table=dbtable
UseSystemD = 1
MonitorReplication = 0
ServiceName = "postgresql-9.4"
SecondLevelMonitor=1
EnvFile=/var/lib/pgsql/pg.env
)
PostgreSQL edb_pg_server_1 (
    Critical = 0
    ResLogLevel = TRACE
    PostgreSQLUser = enterprisedb
    HostName = localhost
    EnvFile = "/PostgresPlus/9.1AS/pgplus_env.sh"
    Port = 5444
    BaseDir = "/PostgresPlus/9.1AS/bin"
    DataDir = "/PostgresPlus/9.1AS/data"
    StartOpts = "-l /tmp/pglog"
    StopOpts = "-m fast"
    DBUser=dbuser
    DBName=dbname
    Table=dbtable
    UseSystemD = 1
    MonitorReplication = 0
    ServiceName = "postgresql-9.4"
    SecondLevelMonitor=1
    EnvFile=/var/lib/pgsql/pg.env
)
```

Sample configuration file for a replication setup in a GCO configuration with Cluster File System (CFS):

```
group PostGresDB_SG (
    SystemList = { iar73003 = 0, iar73004 = 1 }
    TriggersEnabled = { NOFAILOVER }
)

IP PostGres_VIP_Res (
    Device = eth0
    Address = "10.209.95.204"
    NetMask = "255.255.252.0"
)
```

```
NIC Postgres_VIP_NIC (
    Device = eth0
)

PostgreSQL PostGres_Res (
    ResLogLevel = TRACE
    HostName = PostGresDB
    EnvFile = "/usr/pgsql-12/MyEnv"
    Port = 5445
    BaseDir = "/usr/pgsql-12/bin"
    DataDir = "/var/lib/pgsql/12/data"
    StopOpts = "-m fast"
    DBUser = vcuser
    DBName = postgres
    Table = vcstable
    MonitorReplication = 1
    ClientAddr = "10.209.69.83"
    SourceConnStr = "host=10.209.69.83 port=5445
        user=postgres dbname=postgres"
    BackupCmd = "/usr/pgsql-12/bin/pg_basebackup -R
        -X stream -D /var/lib/pgsql/12/data -U replica
        --host=10.209.69.83 --port=5445"
    RecoveryFile = "/var/lib/pgsql/12/postgresql.auto.conf"
    RestartdbToRewind = 1
    RegistrationOfStandby = 1
    DetailedMonitoring = 1
    SplitTakeOver = 1
    AutoTakeOver = 1
)

requires group PostGres_Mounts online local firm
PostGres_Res requires PostGres_VIP_Res
PostGres_VIP_Res requires Postgres_VIP_NIC

// resource dependency tree
//
//     group PostGresDB_SG
//     {
//     PostgreSQL PostGres_Res
//     {
//         IP PostGres_VIP_Res
//         {
```

```
//          NIC Postgres_VIP_NIC
//          }
//          }
//          }

group PostGres_Mounts (
    SystemList = { iar73003 = 0, iar73004 = 1 }
    Parallel = 1
)

CFSMount PosGresArchiveMNT_Res2 (
    MountPoint = "/var/lib/pgsql/12/backups/archive"
    BlockDevice = "/dev/vx/dsk/mqdg/PostGres10_Data_Archive_Vol"
    MountOpt = "cluster,crw"
)

CFSMount PosGresDataMNT_Res2 (
    MountPoint = "/var/lib/pgsql/12/data"
    BlockDevice = "/dev/vx/dsk/mqdg/PostGres10_Data_vol"
    MountOpt = "cluster,crw"
)

requires group cvm online local firm

// resource dependency tree
//
//     group PostGres_Mounts
//     {
//     CFSMount PosGresArchiveMNT_Res2
//     CFSMount PosGresDataMNT_Res2
//     }

group PostgresTakeOver_SG (
    SystemList = { iar73003 = 0, iar73004 = 1 }
    Frozen = 1
    Authority = 1
)

PgSQLRep pgrep (
    PgSQLResource = PostGres_Res
)

requires group PostGresDB_SG online local firm
```

```
// resource dependency tree
//
//     group PostgresTakeOver_SG
//     {
//     PgSQLRep pgreg
//     }
```

Note: A similar `main.cf` file is required for the other cluster.

Sample service group configurations for PostgreSQL

The following graphics depicts service groups with PostgreSQL instances running in a VCS environment.

Figure A-1 Sample service group for a PostgreSQL instance

