

# Oracle® Solaris Cluster Data Service for PostgreSQL Guide

Copyright © 2000, 2013, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

---

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf disposition de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, breveter, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l'accompagne, est concédé sous licence au Gouvernement des Etats-Unis, ou à toute entité qui délivre la licence de ce logiciel ou l'utilise pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique:

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer des dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour ce type d'applications.

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée d'The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation.

# Contents

---

<b>Preface</b> .....	7
<b>1 Installing and Configuring HA for PostgreSQL</b> .....	11
HA for PostgreSQL Overview .....	11
Overview of Installing and Configuring HA for PostgreSQL .....	12
Planning the HA for PostgreSQL Installation and Configuration .....	12
PostgreSQL and Oracle Solaris Zones .....	12
PostgreSQL WAL Shipping .....	13
Configuration Restrictions .....	13
Configuration Requirements .....	15
Installing and Configuring PostgreSQL .....	16
▼ How to Enable a PostgreSQL Database to Run in the Global Zone .....	17
▼ How to Install and Configure PostgreSQL in the Global Zone .....	18
▼ How to Enable a Non-Global Zone to Run PostgreSQL in an HA for Solaris Zones Configuration .....	19
▼ How to Install and Configure PostgreSQL in an HA for Solaris Zones Configuration .....	21
Verifying the Installation and Configuration of PostgreSQL .....	22
▼ How to Verify the Installation and Configuration of PostgreSQL .....	23
Installing the HA for PostgreSQL Package .....	24
▼ How to Install the HA for PostgreSQL Package .....	24
Registering and Configuring HA for PostgreSQL .....	24
Specifying Configuration Parameters for the PostgreSQL Resource .....	25
Specifying the Parameters for the Rolechanger Resource .....	32
Specifying Configuration Files for WAL File Shipping Without Shared Storage .....	36
Preparing Your PostgreSQL Installation for Cluster Control .....	42
Creating and Enabling Resources for PostgreSQL .....	45
Verifying the HA for PostgreSQL Installation and Configuration .....	47
▼ How to Verify the HA for PostgreSQL Installation and Configuration .....	48

▼ How to Verify the Oracle Solaris Cluster HA for PostgreSQL WAL File Shipping Installation and Configuration .....	48
Tuning the HA for PostgreSQL Fault Monitor .....	49
Operation of the HA for PostgreSQL Parameter File .....	50
Operation of the Fault Monitor for HA for PostgreSQL .....	51
Debugging HA for PostgreSQL .....	52
▼ How to Activate Debugging for HA for PostgreSQL .....	52
<b>A Files for Configuring Oracle Solaris Cluster HA for PostgreSQL Resources .....</b>	<b>55</b>
Listing of pgs_config .....	55
Listing of rolechg_config .....	58
<b>B Deployment Example: Installing PostgreSQL in the Global Zone or a Zone Cluster .....</b>	<b>61</b>
Target Cluster Configuration .....	61
Software Configuration .....	61
Assumptions .....	62
Installing and Configuring PostgreSQL on Shared Storage in the Global Zone or a Zone Cluster .....	62
▼ Example: Preparing the Cluster for PostgreSQL .....	62
▼ Example: Configuring Cluster Resources for PostgreSQL .....	63
▼ Example: Modifying the PostgreSQL Configuration File .....	63
▼ Example: Building and Installing the PostgreSQL Software on Shared Storage .....	64
▼ Example: Enabling the PostgreSQL Software to Run in the Cluster .....	65
Installing the PostgreSQL Binaries in the Default Directory (Alternative Installation) .....	66
▼ Example: Building and Installing the PostgreSQL Software in the Default Directory in the Global Zone .....	67
<b>C Deployment Example: Installing PostgreSQL in a Non-Global Zone With HA for Solaris Zones .....</b>	<b>69</b>
Target Cluster Configuration .....	69
Software Configuration .....	69
Assumptions .....	70
Installing and Configuring PostgreSQL on Shared Storage in a Non-Global Zone with HA for Solaris Zones .....	70
▼ Example: Preparing the Cluster for PostgreSQL .....	70

- ▼ Example: Configuring Cluster Resources for PostgreSQL ..... 71
- ▼ Example: Configuring HA for Solaris Zones ..... 71
- ▼ Example: Modifying the PostgreSQL Configuration File ..... 73
- ▼ Example: Building and Installing the PostgreSQL Software on Shared Storage in a  
Non-Global Zone With HA for Solaris Zones ..... 74
- ▼ Example: Enabling the PostgreSQL Software to Run in the Cluster ..... 75
- Installing the PostgreSQL Binaries in the Default Directory in a Non-Global Zone With HA for  
Solaris Zones (Alternative Installation) ..... 76
- ▼ Example: Building and Installing the PostgreSQL Software in the Default Directory in a  
Non-Global Zone With HA for Solaris Zones ..... 76
  
- D Deployment Example: Installing PostgreSQL in the Global Zone Using WAL File Shipping ... 79**
- Target Cluster Configuration ..... 79
- Software Configuration ..... 79
- Assumptions ..... 80
- Installing and Configuring PostgreSQL on Shared Storage in the Global Zone ..... 80
- ▼ Example: Preparing the Cluster for PostgreSQL ..... 80
- ▼ Example: Configuring Cluster Resources for PostgreSQL ..... 81
- ▼ Example: Modifying the PostgreSQL Configuration File ..... 82
- ▼ Example: Building and Installing the PostgreSQL Software on Shared Storage ..... 84
- ▼ Example: Enabling the PostgreSQL Software to Run in the Cluster ..... 85
  
- Index ..... 91**



# Preface

---

*Oracle Solaris Cluster Data Service for PostgreSQL Guide* explains how to install and configure Oracle Solaris Cluster data services.

---

**Note** – This Oracle Solaris Cluster release supports systems that use the SPARC and x86 families of processor architectures. In this document, “x86” refers to the larger family of x86 compatible products. Information in this document pertains to all platforms unless otherwise specified.

---

This document is intended for system administrators with extensive knowledge of Oracle software and hardware. Do not use this document as a planning or presales guide. Before reading this document, you should have already determined your system requirements and purchased the appropriate equipment and software.

The instructions in this book assume knowledge of the Oracle Solaris Operating System and expertise with the volume-manager software that is used with Oracle Solaris Cluster software.

Bash is the default shell for Oracle Solaris 11. Machine names shown with the Bash shell prompt are displayed for clarity.

## Using UNIX Commands

This document contains information about commands that are specific to installing and configuring Oracle Solaris Cluster data services. The document does *not* contain comprehensive information about basic UNIX commands and procedures, such as shutting down the system, booting the system, and configuring devices. Information about basic UNIX commands and procedures is available from the following sources:

- Online documentation for the Oracle Solaris Operating System
- Oracle Solaris Operating System man pages
- Other software documentation that you received with your system

## Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P-1 Typographic Conventions

Typeface	Description	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name% you have mail.</code>
<b>AaBbCc123</b>	What you type, contrasted with onscreen computer output	<code>machine_name% su</code> Password:
<i>aabbcc123</i>	Placeholder: replace with a real name or value	The command to remove a file is <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . <i>A cache</i> is a copy that is stored locally. Do <i>not</i> save the file. <b>Note:</b> Some emphasized items appear bold online.

## Shell Prompts in Command Examples

The following table shows UNIX system prompts and superuser prompts for shells that are included in the Oracle Solaris OS. In command examples, the shell prompt indicates whether the command should be executed by a regular user or a user with privileges.

TABLE P-2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	\$
Bash shell, Korn shell, and Bourne shell for superuser	#
C shell	machine_name%
C shell for superuser	machine_name#



## Related Documentation

Information about related Oracle Solaris Cluster topics is available in the documentation that is listed in the following table. All Oracle Solaris Cluster documentation is available at <http://www.oracle.com/technetwork/indexes/documentation/index.html>.

Topic	Documentation
Hardware installation and administration	<i>Oracle Solaris Cluster 4.1 Hardware Administration Manual</i> Individual hardware administration guides
Concepts	<i>Oracle Solaris Cluster Concepts Guide</i>
Software installation	<i>Oracle Solaris Cluster Software Installation Guide</i>
Data service installation and administration	<i>Oracle Solaris Cluster Data Services Planning and Administration Guide</i> and individual data service guides
Data service development	<i>Oracle Solaris Cluster Data Services Developer's Guide</i>
System administration	<i>Oracle Solaris Cluster System Administration Guide</i> <i>Oracle Solaris Cluster Quick Reference</i>
Software upgrade	<i>Oracle Solaris Cluster Upgrade Guide</i>
Error messages	<i>Oracle Solaris Cluster Error Messages Guide</i>
Command and function references	<i>Oracle Solaris Cluster Reference Manual</i> <i>Oracle Solaris Cluster Data Services Reference Manual</i> <i>Oracle Solaris Cluster Geographic Edition Reference Manual</i> <i>Oracle Solaris Cluster Quorum Server Reference Manual</i>
Compatible software	Oracle Solaris Cluster Compatibility Guide available at the Oracle Solaris Cluster Technical Resources page

## Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

## Getting Help

If you have problems installing or using Oracle Solaris Cluster, contact your service provider and provide the following information.

- Your name and email address (if available)
- Your company name, address, and phone number
- The model number and serial number of your systems
- The release number of the operating environment (for example, Oracle Solaris 11)
- The release number of Oracle Solaris Cluster (for example, Oracle Solaris Cluster 4.1)

Use the following commands to gather information about your system for your service provider.

Command	Function
<code>prtconf -v</code>	Displays the size of the system memory and reports information about peripheral devices
<code>psrinfo -v</code>	Displays information about processors
<code>pkg list</code>	Reports which packages are installed
<code>prtdiag -v</code>	Displays system diagnostic information
<code>/usr/cluster/bin/clnode show-rev -v</code>	Displays Oracle Solaris Cluster release and package version information for each node

Also have available the contents of the `/var/adm/messages` file.

# Installing and Configuring HA for PostgreSQL

---

This chapter explains how to install and configure HA for PostgreSQL and contains the following sections:

- “HA for PostgreSQL Overview” on page 11
- “Overview of Installing and Configuring HA for PostgreSQL” on page 12
- “Planning the HA for PostgreSQL Installation and Configuration” on page 12
- “Installing and Configuring PostgreSQL” on page 16
- “Verifying the Installation and Configuration of PostgreSQL” on page 22
- “Installing the HA for PostgreSQL Package” on page 24
- “Registering and Configuring HA for PostgreSQL” on page 24
- “Verifying the HA for PostgreSQL Installation and Configuration” on page 47
- “Tuning the HA for PostgreSQL Fault Monitor” on page 49
- “Debugging HA for PostgreSQL” on page 52

## HA for PostgreSQL Overview

HA for PostgreSQL enables the Oracle Solaris Cluster software to manage PostgreSQL by providing components to perform the orderly startup, shutdown, and fault monitoring of PostgreSQL.

You can configure HA for PostgreSQL as a failover service. You *cannot* configure HA for PostgreSQL as a multiple-masters service or as a scalable service.

When a PostgreSQL database cluster is managed by the HA for PostgreSQL data service, the PostgreSQL instance becomes a failover PostgreSQL resource across the Oracle Solaris Cluster nodes. The failover is managed by the HA for PostgreSQL data service, which runs within the global zone, zone clusters, and HA for Solaris Zones.

For conceptual information about failover data services, multiple-masters data services, and scalable data services, see *Oracle Solaris Cluster Concepts Guide*.

## Overview of Installing and Configuring HA for PostgreSQL

The following table summarizes the tasks for installing and configuring HA for PostgreSQL and provides cross-references to detailed instructions for performing these tasks. Perform the tasks in the order that they are listed in the table.

TABLE 1-1 Tasks for Installing and Configuring HA for PostgreSQL

Task	Instructions
Plan the installation	<a href="#">“Planning the HA for PostgreSQL Installation and Configuration” on page 12</a>
Install and configure the PostgreSQL software	<a href="#">“Installing and Configuring PostgreSQL” on page 16</a>
Verify the installation and configuration	<a href="#">“How to Verify the Installation and Configuration of PostgreSQL” on page 23</a>
Install HA for PostgreSQL packages	<a href="#">“Installing the HA for PostgreSQL Package” on page 24</a>
Register and configure HA for PostgreSQL resources	<a href="#">“Registering and Configuring HA for PostgreSQL” on page 24</a>
Verify the HA for PostgreSQL installation and configuration	<a href="#">“Verifying the HA for PostgreSQL Installation and Configuration” on page 47</a>
Tune the HA for PostgreSQL fault monitor	<a href="#">“Tuning the HA for PostgreSQL Fault Monitor” on page 49</a>
Debug HA for PostgreSQL	<a href="#">“Debugging HA for PostgreSQL” on page 52</a>

## Planning the HA for PostgreSQL Installation and Configuration

This section contains the information you need to plan your HA for PostgreSQL installation and configuration.

### PostgreSQL and Oracle Solaris Zones

HA for PostgreSQL is supported in a cluster of Oracle Solaris zones, known as an Oracle Solaris Cluster zone cluster, and in the Oracle Solaris Cluster data service for Oracle Solaris Zones (HA for Solaris Zones). HA for Solaris Zones are managed by the Oracle Solaris Zones agent, and are represented by a resource of a resource group.

## PostgreSQL WAL Shipping

The PostgreSQL agent offers three options for a cluster configuration. In these three options, two options leverage the Write Ahead Log (WAL) file shipping features and require the installation of PostgreSQL `pg_standby` utility. The various options for cluster configuration are the following:

- Traditional HA configuration with shared storage. In this configuration, you have a cluster with an active PostgreSQL resource, where the database directories reside on a cluster file system or on a highly available local file system, also called a failover file system.
- WAL file shipping between two PostgreSQL failover resources. In this configuration, you have two independent PostgreSQL resources in a cluster or in different clusters. One of the resources acts as a primary server and obtains the client requests. The other resource acts as a standby server applying the PostgreSQL WAL files shipped from the primary server.
- WAL file shipping without shared storage. This configuration does not require shared storage. The PostgreSQL WAL file shipping replaces the shared storage. This configuration consists of three resource groups. In two single-node resource groups, one resource group contains the designated primary database resource. The other resource group contains the designated standby database resource. The third resource group contains a logical host and a Rolechanger resource. This Rolechanger resource is responsible for transforming the designated standby into an acting primary on a node outage of the designated primary.

## Configuration Restrictions

The configuration restrictions in the subsections that follow apply only to HA for PostgreSQL.



**Caution** – Your data service configuration might not be supported if you do not observe these restrictions.

### Restriction for the Location of the Database Cluster

The PostgreSQL database cluster is where the database files and the configuration files are stored. The database cluster, represented by the configuration variable `PGDATA`, needs to be placed on the shared storage.

### Restriction for the Listening Policy of the PostgreSQL Database Server

HA for PostgreSQL requires that the PostgreSQL listens at the `localhost`. Otherwise the monitoring of your data service will not work. For more information, see [“Preparing Your PostgreSQL Installation for Cluster Control”](#) on page 42.

## Restriction for the PostgreSQL `postgresql.conf` File

The `postgresql.conf` file is one of the central configuration files for a specific PostgreSQL database cluster.

The `postgresql.conf` file must be stored in the PGDATA path. You cannot register HA for PostgreSQL if the file `postgresql.conf` is not in the directory referenced in the PGDATA variable. The other configuration files can be kept elsewhere. For more information about registration, see “[Registering and Configuring HA for PostgreSQL](#)” on page 24.

## Restriction for the Password Policy for the HA for PostgreSQL Monitoring Database

HA for PostgreSQL requires a database to which it can connect and where it can manipulate a table for monitoring purposes. The password policy of this database for access from the `localhost` must be either `trust` or `password`. All other password policies can be whatever is applicable. For more information about setting the password policy, see “[Registering and Configuring HA for PostgreSQL](#)” on page 24. For more information about the password policy, go to <http://www.postgresql.org>.

## Restriction for the PostgreSQL `smf` Service Name in an HA for Solaris Zones Configuration

The PostgreSQL configuration in an HA for Solaris Zones configuration uses the `smf` component of Oracle Solaris Cluster HA for Solaris Zones. The registration of the HA for PostgreSQL data service in an HA for Solaris Zones configuration defines an `smf` service to control the PostgreSQL database. The name of this `smf` service is generated in this naming scheme: `svc:/application/sczone-agents:resource-name`. No other `smf` service with exactly this name can exist.

The associated `smf` manifest is automatically created during the registration process in this location and naming scheme:

`/var/svc/manifest/application/sczone-agents/resource-name.xml`. No other manifest can coexist with this name.

## Restriction for the PostgreSQL WAL File Shipping Without Shared Storage

The `pg_standby` utility must be configured with a trigger file after a failover from the primary to the standby triggering a role conversion. An automatic failback cannot occur because the old primary is now out of synchronization. To invoke an actual copy, the PostgreSQL user needs to copy, customize, and execute the two example scripts: `resilver-step1` and `resilver-step2`.

To minimize the data loss on a planned failover, you should switch the PostgreSQL transaction logs before you perform the failover. For information about switching transaction logs, see <http://www.postgresql.org>.

---

**Note** – The PostgreSQL WAL file Shipping without shared storage configuration cannot be deployed with non-global zones managed by the HA for Solaris Zones agent.

---

## Configuration Requirements

The configuration requirements in this section apply only to HA for PostgreSQL.




---

**Caution** – If your data service configuration does not conform to these requirements, the data service configuration might not be supported.

---

## Dependencies Between HA for PostgreSQL Components

The dependencies between the HA for PostgreSQL components are described in the following table.

TABLE 1-2 Dependencies Between HA for PostgreSQL Components

Component	Dependency
PostgreSQL resource in the global zone or in a zone cluster	SUNW.HAStoragePlus – This dependency is required only if the configuration uses a highly available local file system, or failover file system. It is a resource_offline_restart_dependency.  SUNW.LogicalHostname
PostgreSQL resource in an HA for Solaris Zones configuration	Oracle Solaris Cluster HA for Solaris Zones boot resource.  SUNW.HAStoragePlus – This dependency is a resource_offline_restart_dependency.  SUNW.LogicalHostname – This dependency is required only if the zones boot resource does not manage the zone's IP address.

You set these dependencies, when you register and configure HA for PostgreSQL. For more information, see “[Registering and Configuring HA for PostgreSQL](#)” on page 24.

If more elaborate dependencies are required, see the [r\\_properties\(5\)](#) and [rg\\_properties\(5\)](#) man pages for further dependencies and affinities settings.

## Parameter File for HA for PostgreSQL

HA for PostgreSQL requires a parameter file to pass configuration information to the data service. You must create a directory for this file. Because the directory must be available on each node that is to host the PostgreSQL database, place the directory on the shared storage. If HA for PostgreSQL is configured for an HA for Solaris Zones configuration, this file must be

available in this zone. The parameter file is created automatically when the resource is registered. Although it is possible to store the password for database monitoring in clear text in the parameter file, you must not do it due to security risks. Instead of storing the password in clear text in the parameter file, encrypt the password using the register script.

## Configuration Requirements for the WAL File Shipping Without Shared Storage Configuration

For the WAL file shipping without shared storage configuration, the `rsync` utility is required. As an additional requirement, you need to link some PostgreSQL configuration files outside the `PGDATA` directory. Otherwise these files are destroyed during the resilvering of the primary database. Information about how to perform these steps is available in the comments of the `resilver1` script. The PostgreSQL users on both nodes require a nonpassword login on each node.

# Installing and Configuring PostgreSQL

This section explains only the special requirements for installing PostgreSQL for use with HA for PostgreSQL. For complete information about installing and configuring PostgreSQL, see <http://www.postgresql.org>. For complete information about installing and configuring an Oracle Solaris non-global zone, see *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

Determine if the Oracle-supplied PostgreSQL software is already installed and if the installed version fulfills your requirements. To do this you need to check whether the following minimum packages are installed on your system.

```
database/postgres-84
database/postgres-84/contrib
database/postgres-84/library
database/postgres-84/tests
database/postgres-common
service/database/postgres-84
```

To determine which PostgreSQL version is installed, submit the following commands.

```
# su - non-root-user
$ postmaster --version
```

If the minimum set of packages is not installed, you must decide whether to install PostgreSQL from the Oracle Solaris media or to build PostgreSQL on your own. If the version does not fulfill your needs, you must build PostgreSQL on your own.

For each PostgreSQL database that you are installing and configuring choose the following tasks according to your zone type.



Determine whether you have to configure HA for PostgreSQL to run in the global zone or in an HA for Solaris Zones configuration.

- The global zone procedure is applicable if you install PostgreSQL in the global zone or in a zone cluster.
- The HA for Solaris Zones procedure is applicable if you install PostgreSQL in a non-global zone that is configured with HA for Solaris Zones.

To install and configure PostgreSQL in the global zone, complete the following tasks:

- [“How to Enable a PostgreSQL Database to Run in the Global Zone”](#) on page 17
- [“How to Install and Configure PostgreSQL in the Global Zone”](#) on page 18

To install and configure PostgreSQL in an HA for Solaris Zones configuration, complete the following tasks:

- [“How to Enable a Non-Global Zone to Run PostgreSQL in an HA for Solaris Zones Configuration”](#) on page 19
- [“How to Install and Configure PostgreSQL in an HA for Solaris Zones Configuration”](#) on page 21

## ▼ How to Enable a PostgreSQL Database to Run in the Global Zone

For a complete example of deploying in the global zone, see [Appendix B, “Deployment Example: Installing PostgreSQL in the Global Zone or a Zone Cluster.”](#)

- 1 As superuser, register the `SUNW.HASStoragePlus` and `SUNW.gds` resource types.

```
# clresourcetype register SUNW.HASStoragePlus SUNW.gds
```

- 2 Create a failover resource group.

```
# clresourcegroup create PostgreSQL-resource-group
```

- 3 Create a resource for PostgreSQL's disk storage.

```
# clresource create -t SUNW.HASStoragePlus \  
-p FileSystemMountPoints=PostgreSQL-instance-mount-points \  
PostgreSQL-has-resource
```

- 4 (Optional) If you plan to access the database from a logical host, choose the following tasks according to your zone type.

```
# clreslogicalhostname create -g PostgreSQL-resource-group \  
PostgreSQL-logical-hostname-resource-name
```

- 5 Enable the failover resource group.

```
# clresourcegroup online -M PostgreSQL-resource-group
```

## 6 Create a directory for the HA for PostgreSQL parameter file.

```
# mkdir PostgreSQL-instance-mount-points/parameter-dir
```

# ▼ How to Install and Configure PostgreSQL in the Global Zone

---

**Note** – For complete information about installing PostgreSQL, go to <http://www.postgresql.org>.

---

For a complete example of deployment in the global zone, see [Appendix B, “Deployment Example: Installing PostgreSQL in the Global Zone or a Zone Cluster.”](#)

**Before You Begin** Determine the following requirements for the deployment of PostgreSQL with Oracle Solaris Cluster software:

- See whether the PostgreSQL version that you need is already installed on each cluster node by searching the most probable root paths to find `bin/postmaster`:

<code>/usr</code>	Root path for PostgreSQL shipped with Oracle Solaris software.
<code>/usr/local/psql</code>	Root path for the PostgreSQL build without a prefix.
<code>/your-path</code>	Fully customized root path for PostgreSQL. This is where to place the binaries on the shared storage. A known convention is <code>/path/postgresql-x.y.z</code> .

- Determine the number of PostgreSQL resources to deploy.
- Determine which cluster file systems will be used by each PostgreSQL resource.
- Make sure, that a C compiler, `make`, and the `readline` package are installed. These packages are needed to build PostgreSQL from the source code downloads from <http://www.postgresql.org>.

The following assumptions are made:

- The compiler `gcc` and the `gmake` package are installed in `/usr/soft`.
- The `readline` package is installed under `/usr/local`.
- The PostgreSQL database software will be installed on the shared storage in the directory `version` in the failover file system `/global/postgres`.
- The PostgreSQL database cluster will be installed in the same file system as the database software, in the directory `/global/postgres/data`.
- The home directory of the `postgres` user is `/global/postgres`.

- The PostgreSQL build directory is in `/tmp/postgres/version`, and the software is already downloaded and extracted in this place.

**1 As superuser create the home directory for the PostgreSQL user on one node.**

```
# mkdir /global/postgres
```

**2 Add a group for PostgreSQL on every node.**

```
# groupadd -g 1000 postgres
```

**3 Add a user who owns the PostgreSQL installation on every node.**

```
# useradd -u 1000 -g postgres -d /global/postgres -s /usr/bin/ksh postgres
# chown -R postgres:postgres /global/postgres
```

**4 Switch to the PostgreSQL user.**

```
# su - postgres
```

**5 Set your PATH variable.**

```
$ PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
$ export PATH
```

**6 Set your LD\_LIBRARY\_PATH variable.**

```
$ LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:
$ export LD_LIBRARY_PATH
```

**7 Switch to your build directory.**

```
$ cd /tmp/postgres/version
```

**8 Configure the PostgreSQL build.**

```
$ ./configure --prefix=/global/postgres/version
```

**9 Complete, verify and install the build.**

```
$ make
$ make check
$ make install
```

## ▼ How to Enable a Non-Global Zone to Run PostgreSQL in an HA for Solaris Zones Configuration

For a complete example of deploying in an HA for Solaris Zones configuration, see [Appendix C](#), “Deployment Example: Installing PostgreSQL in a Non-Global Zone With HA for Solaris Zones.”

**1 As superuser, register the SUNW.HASStoragePlus and SUNW.gds resource types.**

```
# clresource_type register SUNW.HASStoragePlus SUNW.gds
```

**2 Create a failover resource group.**

```
# clresourcegroup create PostgreSQL-resource-group
```

**3 Create a resource for the PostgreSQL zone's disk storage.**

```
# clresource create -t SUNW.HASStoragePlus \  
-p FileSystemMountPoints=PostgreSQL-instance-mount-points \  
PostgreSQL-has-resource
```

**4 (Optional) If you want the protection against a total adapter failure for your public network, create a resource for the PostgreSQL's logical hostname.**

```
# clreslogicalhostname create -g PostgreSQL-resource-group \  
PostgreSQL-logical-hostname-resource-name
```

**5 Place the resource group in the managed state.**

```
# clresourcegroup online -M PostgreSQL-resource-group
```

**6 Install the non-global zone.**

Install the zone according to the Oracle Solaris Cluster HA for Solaris Zones agent documentation, assuming that the resource name is *pgsql-zone-rs* and that the zone name is *pgsql-zone*.

**7 Verify the zone's installation.**

```
# zoneadm -z psql-zone boot  
# zoneadm -z psql-zone halt
```

**8 Register the zone's boot component.****a. Copy the zone resource boot component configuration file.**

```
# cp /opt/SUNWsczone/sczbt/util/sczbt_config zones-target-configuration-file
```

**b. Use a plain text editor to set the following variables:**

```
RS=pgsql-zone-rs  
RG=PostgreSQL-resource-group  
PARAMETERDIR=pgsql-zone-parameter-directory  
SC_NETWORK=true|false  
SC_LH=PostgreSQL-logical-hostname-resource-name  
FAILOVER=true|false  
HAS_RS=PostgreSQL-has-resource  
Zonename=pgsql-zone  
Zonebootopt=zone-boot-options  
Milestone=zone-boot-milestone  
Mounts=
```

**c. Create the parameter directory for your zone's resource.**

```
# mkdir psql-zone-parameter-directory
```

**d. Execute the Oracle Solaris Cluster HA for Solaris Zones's registration script.**

```
# /opt/SUNWsczone/sczbt/util/sczbt_register -f zones-target-configuration-file
```

e. Enable the HA for Solaris Zones resource.

```
# clresource enable pgsql-zone-rs
```

9 Enable the resource group.

```
# clresourcegroup online PostgreSQL-resource-group
```

## ▼ How to Install and Configure PostgreSQL in an HA for Solaris Zones Configuration

---

**Note** – For complete information about installing PostgreSQL, go to <http://www.postgresql.org>.

---

**Before You Begin** Determine the following requirements for the deployment of PostgreSQL with Oracle Solaris Cluster software:

- See whether the PostgreSQL version that you need is already installed on each cluster node by searching the most probable root paths where you find `bin/postmaster`:

<code>/usr</code>	Root path for PostgreSQL shipped with the Oracle Solaris software.
<code>/usr/local/pgsql</code>	Root path for the PostgreSQL build without a prefix.
<code>/your-path</code>	Fully customized root path for PostgreSQL. This is where to place the binaries on the shared storage. A known convention is <code>/path/postgresql-x.y.z</code> .

- Determine the number of PostgreSQL resources to deploy.
- Determine which cluster file systems will be used by each PostgreSQL resource.
- Make sure that a C compiler, `make`, and the `readline` package are installed. These packages are needed to build PostgreSQL from the source code downloads from <http://www.postgresql.org>.

The following assumptions are made:

- The zone `postgres-zone` is installed and configured on every node.
- The compiler `gcc` and the `gmake` package are installed in `/usr/sfw`.
- The `readline` package is installed under `/usr/local`.
- The PostgreSQL database software will be installed on the shared storage, in the directory `version` in the failover file system `/postgres`.
- The PostgreSQL database cluster will be installed in the same file system as the database software, in the directory `/postgres/data`.

- The home directory of the *postgres* user is */postgres*.
- The PostgreSQL build directory is in */tmp/postgres/version*, and the software is already downloaded and extracted in this place.

**1 Log in to the non-global zone.**

```
# zlogin postgres-zone
```

**2 Add a group for PostgreSQL.**

```
# groupadd -g 1000 postgres
```

**3 Add a user who owns the PostgreSQL installation on every node.**

```
# useradd -u 1000 -g postgres -d /postgres -m -s /usr/bin/ksh postgres
```

**4 Switch to the PostgreSQL user.**

```
# su - postgres
```

**5 Set your PATH variable.**

```
$ PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
$ export PATH
```

**6 Set your LD\_LIBRARY\_PATH variable.**

```
$ LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:
$ export LD_LIBRARY_PATH
```

**7 Switch to your build directory.**

```
$ cd /tmp/postgres/version
```

**8 Configure the PostgreSQL build.**

```
$ ./configure --prefix=/postgres/version
```

**9 Complete, verify, and install the build.**

```
$ make
$ make check
$ make install
```

## Verifying the Installation and Configuration of PostgreSQL

Before you install the HA for PostgreSQL packages, verify that each PostgreSQL instance that you created is correctly configured to run in a cluster. The instance is the PostgreSQL database cluster together with the associated postmaster processes. This verification does not confirm that the PostgreSQL databases are highly available because the HA for PostgreSQL data service is not yet configured.

## ▼ How to Verify the Installation and Configuration of PostgreSQL

Perform this procedure for each PostgreSQL instance that you created in “[Installing and Configuring PostgreSQL](#)” on page 16. During the verification you will complete the PostgreSQL postinstallation steps.

**Before You Begin** Determine whether you are in a non-global zone or in the global zone. If you are in an HA for Solaris Zones configuration, use */postgres* instead of */global/postgres* for your directory prefix in this procedure.

### 1 Switch to the PostgreSQL user if necessary.

```
# su - postgres
```

### 2 (Optional) Set the PATH and LD\_LIBRARY\_PATH variables.

```
$ PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
$ export PATH
$ LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:
$ export LD_LIBRARY_PATH
```

### 3 Set the PGDATA variable.

The PGDATA variable points to the directory where the PostgreSQL database cluster is installed. The PostgreSQL database cluster is a directory that contains the configuration and the data files for all the databases.

```
$ PGDATA=/global/postgres/data
$ export PGDATA
```

### 4 Create the data directory and the logs directory.

```
$ mkdir /global/postgres/data
$ mkdir /global/postgres/logs
```

### 5 Initialize the PostgreSQL cluster.

```
$ cd ~/postgres-version
$ ./bin/initdb -D $PGDATA
```

### 6 Start the PostgreSQL database server.

```
$ ./bin/pg_ctl -l /global/postgres/logs/firstlog start
```

### 7 Create and delete a test database.

```
$ ./bin/createdb test
$ ./bin/dropdb test
```

### 8 If you are in a non-global zone, leave this zone and return to the target zone.

## Installing the HA for PostgreSQL Package

If you did not install the HA for PostgreSQL package during your initial Oracle Solaris Cluster installation, perform this procedure to install the package.

### ▼ How to Install the HA for PostgreSQL Package

Perform this procedure on each cluster node where you want the HA for PostgreSQL software to run.

**1 On the cluster node where you are installing the data service package, assume the root role.**

**2 Ensure that the solaris and ha-cluster publishers are valid.**

```
# pkg publisher
PUBLISHER          TYPE    STATUS  URI
solaris            origin online  solaris-repository
ha-cluster         origin online  ha-cluster-repository
```

For information about setting the solaris publisher, see [“Set the Publisher Origin to the File Repository URI”](#) in *Copying and Creating Oracle Solaris 11.1 Package Repositories*.

**3 Install the HA for PostgreSQL software package.**

```
# pkg install ha-cluster/data-service/postgresql
```

**4 Verify that the package installed successfully.**

```
$ pkg info ha-cluster/data-service/postgresql
```

Installation is successful if output shows that State is Installed.

**5 Perform any necessary updates to the Oracle Solaris Cluster software.**

For instructions on updating single or multiple packages, see [Chapter 11, “Updating Your Software,”](#) in *Oracle Solaris Cluster System Administration Guide*.

## Registering and Configuring HA for PostgreSQL

Before you perform the procedures in this section, ensure that the HA for PostgreSQL data service packages are installed.

The configuration and registration file in the `/opt/SUNWscPostgreSQL/util` directory exists to register the HA for PostgreSQL resources. This file defines the dependencies that are required between the HA for PostgreSQL component and other resources. For information about these dependencies, see [“Dependencies Between HA for PostgreSQL Components”](#) on page 15

This section covers the following main topics:



- “Specifying Configuration Parameters for the PostgreSQL Resource” on page 25
- “Preparing Your PostgreSQL Installation for Cluster Control” on page 42
- “How to Create and Enable Resources for PostgreSQL” on page 45

## Specifying Configuration Parameters for the PostgreSQL Resource

HA for PostgreSQL provides a script that automates the process of configuring the PostgreSQL resource. This script obtains configuration parameters from the `pgs_config` file. A template for this file is in the `/opt/SUNWscPostgreSQL/util` directory. To specify configuration parameters for the PostgreSQL resource, copy the `pgs_config` file to another directory and edit the copied `pgs_config` file.

---

**Note** – This configuration file needs to be accessible from the zone where the PostgreSQL software is installed.

---

Each configuration parameter in the `pgs_config` file is defined as a keyword-value pair. The `pgs_config` file already contains the required keywords and equals signs. For more information, see “Listing of `pgs_config`” on page 55. When you edit the `/myplace/pgs_config` file, add the required value to each keyword.

The keyword-value pairs in the `pgs_config` file are as follows:

```
RS=PostgreSQL-resource
RG=PostgreSQL-resource-group
PORT=80
LH=PostgreSQL-logical-hostname-resource-name
HAS_RS=PostgreSQL-has-resource
PFILE=pgsql-parameter-file
ZONE=pgsql-zone
ZONE_BT=pgsql-zone-rs
PROJECT=pgsql-zone-project
USER=pgsql-user
PGROOT=pgsql-root-directory
PGDATA=pgsql-data-directory
PGPORT=pgsql-port
PGHOST=pgsql-host
PGLOGFILE=pgsql-log-file
LD_LIBRARY_PATH=pgsql-ld-library-path
ENVSCRIPT=pgsql-environment-script
SCDB=pgsql-mon-db
SCUSER=pgsql-mon-user
SCTABLE=pgsql-mon-table
SCPASS=pgsql-mon-pwd
NOCONRET=pgsql-noconn-rtcode
STDBY_RS=PostgreSQL-standbyresource
STDBY_RG= PostgreSQL-standby-resource-group
```

```

STDBY_USER=PostgreSQL-standby-user
STDBY_HOST=PostgreSQL-standby-host
STDBY_PARFILE=PostgreSQL-standby-parameter-file
STDBY_PING=Number-of-packets
ROLECHG_RS=PostgreSQL-rolechanger-resource
SSH_PASSDIR=PostgreSQL-user-passphrase-directory
    
```

The meaning and permitted values of the keywords in the `pgs_config` file are as follows:

**RS=PostgreSQL-resource**

Specifies the name that you are assigning to the PostgreSQL resource. You must specify a value for this keyword.

**RG=PostgreSQL-resource-group**

Specifies the name of the resource group where the PostgreSQL resource will reside. You must specify a value for this keyword.

**PORT=80**

In a global zone configuration, specifies the value of a dummy port only if you specified the LH value for the PostgreSQL resource. This variable is used only at registration time. If you will not specify an LH, omit this value.

In an HA for Solaris Zones configuration, omit this value.

**LH=PostgreSQL-logical-hostname-resource-name**

In a global zone configuration, specifies the name of the `SUNW.LogicalHostname` resource for the PostgreSQL resource. This name must be the `SUNW.LogicalHostname` resource name you assigned when you created the resource in [“How to Enable a Non-Global Zone to Run PostgreSQL in an HA for Solaris Zones Configuration” on page 19](#). If you did not register a `SUNW.LogicalHostname` resource, omit this value.

In a configuration using HA for Solaris Zones and WAL file shipping without shared storage, omit this value.

**HAS\_RS=PostgreSQL-has-resource**

Specifies the names of resources on which your PostgreSQL will depend, for example, the `SUNW.HASStoragePlus` resource, for the PostgreSQL resource. This name must be the `SUNW.HASStoragePlus` resource name that you assigned when you created the resource in [“How to Enable a PostgreSQL Database to Run in the Global Zone” on page 17](#).

Dependencies to additional resources can be specified here. They must be separated by a comma. In a configuration using WAL file shipping without shared storage, omit this value.

**PFILE=pgsql-parameter-file**

Specifies the name of the parameter file where the PostgreSQL specific parameters of the PostgreSQL resource are stored. This file is automatically created at registration time. You must specify a value for this keyword.

**ZONE=pgsql-zonename**

Specifies the name of the HA for Solaris Zones configuration to host the PostgreSQL database. Omit this value if you configure a global zone environment.

**ZONE\_BT**=*pgsql-zone-rs*

Specifies the name of the zone boot resource in an HA for Solaris Zones configuration. Omit this value if you configure a global zone environment.

**PROJECT**=*pgsql-zone-project*

Specifies the name of the resource management project in the HA for Solaris Zones. Omitting this value in an HA for Solaris Zones configuration results in the default project for USER. Leave the value blank for a global zone configuration.

**USER**=*pgsql-user*

Specifies the name of the Oracle Solaris user who owns the PostgreSQL database. You must specify a value for this keyword.

**PGROOT**=*pgsql-root-directory*

Specifies the name of the directory in which PostgreSQL is installed. For example, if PostgreSQL version 8.1.2 is installed in `/global/postgres/postgresql-8.1.2`, the variable PGROOT needs to be set to `/global/postgres/postgresql-8.1.2`. A valid PGROOT variable contains the file `pg_ctl`, which is located in its subdirectory `bin`. You must specify a value for this keyword.

Examples for PGROOT:

<code>/usr</code>	Root path for PostgreSQL shipped with Oracle Solaris software.
<code>/usr/local/pgsql</code>	Root path for the PostgreSQL build without a prefix.
<code>/your-path</code>	Fully customized root path for PostgreSQL. This is where to place the binaries on the shared storage. A known convention is <code>/path/postgresql-x.y.z</code> .

**PGDATA**=*pgsql-data-directory*

Specifies the name of the directory where the “PostgreSQL data cluster” is initialized. This directory is where the data directories and at least the `postgresql.conf` file are located. You must specify a value for this keyword.

**PGPORT**=*pgsql-port*

Specifies the port on which the PostgreSQL server will listen.

**PGHOST**=*pgsql-host*

Specifies the hostname or directory that is used by the probe. If PGHOST is a hostname, the hostname is used by the probe to connect to the database. If PGHOST is a directory, the probe expects the UNIX domain socket in this directory to establish its connection. The PGHOST variable is referenced only by the probe and the database must be configured according to this setting.

**PGLOGFILE**=*pgsql-log-file*

Specifies the name of the log file of PostgreSQL. All server messages will be found in this file. You must specify a value for this keyword.

**LD\_LIBRARY\_PATH**=*pgsql-ld-library-path*

Specifies the libraries needed to start the PostgreSQL server and utilities. This parameter is optional.

**ENVSCRIPT**=*pgsql-environment-script*

Specifies the name of a script to source PostgreSQL-specific environment variables. In a global zone configuration, the script type is either C shell or Korn shell, according to the login shell of the PostgreSQL user. In an HA for Solaris Zones configuration, the script type must be a valid Korn shell script.

This parameter is optional.

**SCDB**=*pgsql-mon-db*

Specifies the name of the PostgreSQL database that will be monitored. You must specify a value for this keyword.

**SCUSER**=*pgsql-mon-user*

Specifies the name of the PostgreSQL database user, which is needed to monitor the condition of the database. This user will be created during the installation process. You must specify a value for this keyword.

**SCTABLE**=*pgsql-mon-table*

Specifies the name of the table that will be modified to monitor the health of the PostgreSQL application. This table will be created during the installation process. You must specify a value for this keyword.

**SCPASS**=*pgsql-mon-pwd*

Specifies the password for SCUSER. If no password is specified, the user set by SCUSER needs to be allowed to log in from the `localhost` without a password challenge or the stored password must be encrypted.

This parameter is optional.

**NOCONRET**=*pgs-noconn-rtcode*

Specifies the value below 100 of the return code for failed database connections. For more information, see [“Tuning the HA for PostgreSQL Fault Monitor” on page 49](#).

**STDBY\_RS**=*PostgreSQL-standbyresource*

Specifies the name you assigned to the PostgreSQL standby resource. You must specify a value for the keyword on this primary if you configure WAL file shipping as a replacement for the shared storage.

**STDBY\_RG**=*PostgreSQL-standby-resource-group*

Specifies the name of the resource group where the PostgreSQL standby resource resides. You must specify a value for this keyword on the primary if you configure WAL file shipping as a replacement for shared storage.

*STDBY\_USER=PostgreSQL-standby-resource-user*

Specifies the name of the Oracle Solaris user who owns the PostgreSQL standby database. You must specify a value for this keyword on the primary if you configure WAL file shipping as a replacement for shared storage.

*STDBY\_HOST=PostgreSQL-standby-host*

Specifies name of the cluster node that hosts the designated standby database. You must specify a value for this keyword on the primary if you configure WAL file shipping as a replacement for shared storage.

*STDBY\_PARFILE=PostgreSQL-standby-parameterfile*

Specifies the name of the parameter file of the PostgreSQL standby resource. You must specify a value for this keyword on the primary if you configure WAL file shipping as a replacement for shared storage.

*STDBY\_PING=Number of packets*

Specifies the number of packages the primary uses to ping the standby host. This value is optional and the default is five packets.

*ROLECHG\_RS=PostgreSQL-rolechanger-resource*

Specifies the name of the PostgreSQL Rolechanger resource. You must specify a value for the keyword on the standby host if you configure WAL file shipping as a replacement for shared storage.

*SSH\_PASSDIR=PostgreSQL-user-passphrase-directory*

Specifies the directory where a ssh passphrase is stored at registration time. This parameter is optional.

For illustration purposes, two examples for the `pgs_config` file are provided. The first example shows the `pgs_config` file for a global zone configuration and second example shows the `pgs_config` file for an HA for Solaris Zones configuration.

**EXAMPLE 1-1** Sample `pgs_config` File for a Global Zone Configuration

This example shows a `pgs_config` file in which configuration parameters are set as follows:

- The name of the PostgreSQL resource is `postgres - rs`.
- The name of the resource group for the PostgreSQL resource is `postgres - rg`.
- The value of the dummy port for the PostgreSQL resource is `80`.
- The name of the `SUNW.LogicalHostname` resource is `postgres - lh`.
- The name of the `SUNW.HAStoragePlus` resource which manages the file system for PostgreSQL is `postgres - has - rs`.
- The parameter file will be generated in `/global/postgres/pfile`.
- The `null` value for `ZONE`, `ZONE_BT`, and `PROJECT` indicates, that it is a global zone configuration.
- The name of the Oracle Solaris user who owns PostgreSQL is `postgres`.

**EXAMPLE 1-1** Sample `pgs_config` File for a Global Zone Configuration (Continued)

- The PostgreSQL software is installed in `/global/postgres/postgresql-8.1.2`.
- The PostgreSQL data and configuration files are installed under `/global/postgres/data`.
- The PostgreSQL database server listens on port 5432. The probe connects by using the UNIX domain socket in the `/tmp` directory.
- The log file for the database server is `/global/postgres/logs/scinstance1`.
- The libraries for the PostgreSQL server are stored in the paths of the `LD_LIBRARY_PATH` `/usr/sfw/lib:/usr/local/lib:/usr/lib:`.
- Additional PostgreSQL variables are set in `/global/postgres/variables.ksh`.
- The database that will be monitored is `testdb`.
- The user for the database monitoring is `testusr`.
- The table `testtbl` will be modified to probe the condition of the database.
- The password for the user `testusr` is `testpwd`, although it is not stored in the configuration file.
- If a connection to the database `testdb` fails, the probe returns with return code 10.

```
RS=postgres-rs
RG=postgres-rg
PORT=80
LH=postgres-lh
HAS_RS=postgres-has-rs
PFILE=/global/postgres/pfile
ZONE=
ZONE_BT=
PROJECT=
USER=postgres
PGROOT=/global/postgres/postgresql-8.1.2
PGDATA=/global/postgres/data
PGPORT=5432
PGHOST=
PGLOGFILE=/global/postgres/logs/scinstance1
LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:
ENVSCRIPT=/global/postgres/variables.ksh
SCDB=testdb
SCUSER=testusr
SCTABLE=testtbl
SCPASS=
NOCONRET=10
```

**EXAMPLE 1-2** Sample `pgs_config` File for an HA for Solaris Zones Configuration

This example shows an `pgs_config` file in which configuration parameters are set as follows:

- The name of the PostgreSQL resource is `postgres-zrs`.
- The name of the resource group for the PostgreSQL resource is `postgres-rg`.
- The values for the `PORT` variable, `LH` variable, and the `HAS-RS` variable are not set.

## EXAMPLE 1-2 Sample pgs\_config File for an HA for Solaris Zones Configuration (Continued)

- The parameter file will be generated in /postgres/pfile.
- The PostgreSQL database server will be started in zone pgs-zone.
- The boot component resource for the zone pgs-zone is named pgs-zone-rs.
- The PostgreSQL database server will be started under the project pgs-project.
- The name of the Oracle Solaris user who owns PostgreSQL is zpostgr.
- The PostgreSQL software is installed in /postgres/postgresql-8.1.2.
- The PostgreSQL data and configuration files are installed in /postgres/data.
- The PostgreSQL database server listens on port 5432. The probe connects using the UNIX domain socket in /tmp.
- The log file for the database server is /postgres/logs/scinstance1.
- The libraries for the PostgreSQL server are stored in the paths of LD\_LIBRARY\_PATH /usr/sfw/lib:/usr/local/lib:/usr/lib:.
- Additional PostgreSQL variables are set in /postgres/variables.ksh.
- The database that will be monitored is testdb.
- The user for the database monitoring is testusr.
- The table testtbl will be modified to probe the condition of the database.
- The password for the user testusr is testpwd, although it is not stored in the configuration file.
- If a connection to the database testdb fails, the probe returns with return code 10.

```

RS=postgres-zrs
RG=postgres-rg
PORT=
LH=
HAS_RS=
PFILE=/postgres/pfile
ZONE=pgs-zone
ZONE_BT=pgs-zone-rs
PROJECT=pgs-project
USER=zpostgr
PGROOT=/postgres/postgresql-8.1.2
PGDATA=/postgres/data
PGPORT=5432
PGHOST=
PGLOGFILE=/postgres/logs/scinstance1
LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:
ENVSCRIPT=/postgres/variables.ksh
SCDB=testdb
SCUSER=testusr
SCTABLE=testtbl
SCPASS=
NOCONRET=10

```

## Specifying the Parameters for the Rolechanger Resource

Oracle Solaris Cluster HA for PostgreSQL software provides a script that automates the process of configuring the PostgreSQL Rolechanger resource. This script obtains configuration parameters from the `rolechg_config` file. A template for this file is in the `/opt/SUNWscPostgreSQL/rolechg/util` directory. To specify configuration parameters for the PostgreSQL resource, copy the `rolechg_config` file to another directory and edit the file.

Each configuration parameter in the `rolechg_config` file is defined as a keyword-value pair. The `rolechg_config` file already contains the required keywords and equals signs. For more information, see the “[Listing of `rolechg\_config`](#)” on page 58. When you edit the `/myplace/rolechg_config` file, add the required value to each keyword.

The keyword-value pairs in the `rolechg_config` file are as follows:

```
RS=Rolechanger-resource-name
RG=Rolechanger-resource-group
PORT=80
LH=Rolechanger-logical-host
HAS_RS=Rolechanger-dependency-list
STDBY_RS=PostgreSQL-standby-resource-name
PRI_RS=PostgreSQL-primary-resource-name
STDBY_HOST=PostgreSQL-standby-hostname
STDBY_PFILE=PostgreSQL-standby-parameter-file
TRIGGER=PostgreSQL-pg_standby-trigger-file
WAIT=Seconds-before-trigger
```

The permitted values of the keywords `rolechg_config` and their explanation are as follows:

**RS=Rolechanger-resource-name**

Specifies the name assigned to the Rolechanger resource. You must specify a value for this keyword.

**RG=Rolechanger-resource-group**

Specifies the name assigned to the Rolechanger resource group. You must specify a value for this keyword.

**PORT=80**

In a global zone configuration, specifies the value of a dummy port only if you specified the LH value for the Rolechanger resource. This variable is used only during registration.

**LH=Rolechanger-logical-host**

In a global zone configuration, specifies the name of the `SUNW.LogicalHostname` resource for the Rolechanger resource.

**HAS\_RS=Rolechanger-dependency-list**

Specifies the dependency list for the Rolechanger resource. If you have only the Rolechanger resource and the logical host in your resource group, omit this value.



*STDBY\_RS=PostgreSQL-standby-resource-name*

Specifies the name assigned to the PostgreSQL standby resource. You must specify a value for this keyword.

*PRI\_RS=PostgreSQL-primary-resource-name*

Specifies the name of the PostgreSQL primary resource. You must specify a value for this keyword.

*STDBY\_HOST=PostgreSQL-standby-hostname*

Specifies the name of the host running the PostgreSQL standby resource group. You must specify a value for this keyword.

*STDBY\_PFILE=PostgreSQL-standby-parameter-file*

Specifies the name of the PostgreSQL standby resource parameter file. You must specify a value for this keyword on the primary if you configure WAL file shipping as a replacement for shared storage.

*TRIGGER=PostgreSQL-pg\_standby-trigger-file*

Specifies the trigger file for the PostgreSQL `pg_standby` utility. The trigger file must be an absolute path to a file name. You must specify a value for this keyword.

*WAIT=Seconds-before-trigger*

Specifies the number of seconds to wait before touching the trigger file, which starts the conversion from a standby to a primary. You must specify a value for this keyword.

The Rolechanger component of the PostgreSQL agent delivers two resilver scripts in the `/opt/SUNWscPostgreSQL/rolecht/util` directory. The scripts are called `resilver-step1` and `resilver-step2`. The PostgreSQL user needs to copy, modify, and execute these scripts. The purpose of these scripts is to automate an exact copy from the standby to the primary after a failover. These scripts should incur a minimal amount of downtime, and provide a maximum amount of guidance.

The scripts rely on certain assumptions for the PostgreSQL configuration to work. You need to prepare your PostgreSQL installation according to the following assumptions:

- The file `postgresql.conf` is linked to another directory than PGDATA, for example:  
`postgresql.conf -> ../conf/postgresql.conf.`
- The file `recovery.conf/recovery.done` is linked to another directory than PGDATA, for example: `recovery.conf -> ../conf/recovery.conf.`
- Every other configuration file in PGDATA, which has to vary between the designated primary and the designate standby is linked to another directory than PGDATA.
- The Postgres users on the primary and on the standby are identical and trust each other on a ssh login without password request.
- Each PostgreSQL installation is configured with an appropriate archive command and `recovery.conf/done` file.

When a `recovery.conf` file exists in the `PGDATA` directory, PostgreSQL executes the command specified in this file to obtain the WAL logs for its recovery. After finishing the recovery, PostgreSQL renames the file `recovery.conf` to `recovery.done`. To make the WAL file shipping and resilver scripts work properly, and for any other type of resilvering you might implement, you need to perform two steps. You have to create a link `recovery.conf` on the designated standby and a link `recovery.done` on the designated primary from your `PGDATA` directory to `../conf/recovery.conf`.

The following examples show the different PostgreSQL configurations on the designated primary and standby servers. The designated primary and standby servers have different archive and recovery commands. In [Example 1–4](#), the resilvering scripts are also explained in detail.

#### EXAMPLE 1-3 Example for the Designated Primary

This example shows the required archive and recovery configuration for the designated primary server.

The archive command in `postgresql.conf`:

```
archive_command = '/usr/local/bin/rsync -arv %p \
standby:/pgs/82_walarchives/%f</dev/null'
```

The contents of `recovery.conf/done`:

```
restore_command = 'cp /pgs/82_walarchives/%f %p'
```

#### EXAMPLE 1-4 Example for the Designated Standby

This example shows the required archive, recovery, and resilver configuration for the designated standby server.

The archive command in `postgresql.conf`:

```
archive_command = '/usr/local/bin/rsync -arv %p \
standby:/pgs/82_walarchives/%f</dev/null'
```

The contents of `recovery.conf/done`:

```
restore_command = '/pgs/postgres-8.2.5/bin/pg_standby -k 10 -t \
/pgs/data/failover /pgs/82_walarchives %f %p'
```

The two scripts have various variables that need to be customized. The key-value pair and explanation for the two scripts are as follows:

Explanation for the script `resilver-step1`

`SOURCE_DATA=PGDATA` of the standby

Specifies the `PGDATA` directory of the current node. For normal use, it would be the one on the designated standby node.

`TARGET_DATA=PGDATA of the primary`

Specifies the PGDATA of the target node. For normal use, it would be the one on the designated primary node.

`TARGET=Primary-host`

Specifies the name of the target node. For normal use, it would be the name of the designated primary node.

`PGS_BASE=/pgs/postgres-8.2.5`

Specifies the PostgreSQL base directory, where the PostgreSQL binaries are located.

`PRI_GRP=primary-rg`

Specifies the resource group, which contains the cluster resource of the designated primary.

`STDBY_GRP=standby-rg`

Specifies the resource group, which contains the cluster resource of the designated standby.

`STDBY_RS=standby-rs`

Specifies the resource name of the designated standby.

`PGPORT=5432`

Specifies the database port.

`ROLECHG_GRP=rolechg-rg`

Specifies the resource group, which contains the Rolechanger resource.

`RSYNC=/usr/local/bin/rsync -rav`

Specifies the absolute path to the RSYNC command including the necessary options.

`SSH_PASSPHRASE=false`

Specifies whether your ssh key is secured by a passphrase or not.

Explanation for the script `resilver-step2`

`SOURCE=Standby-host`

Specifies the name of the source node. For normal use, it would be the name of the designated standby node.

`SOURCE_DATA=PGDATA of the standby`

Specifies the PGDATA directory of the current node. For normal use, it would be the name of the designated standby node.

`TARGET_DATA=PGDATA of the primary`

Specifies the PGDATA of the target node. For normal use, it would be the name of the designated primary node.

`TARGET=Primary-host`

Specifies the name of the target node. For normal use, it would be the name of the designated primary node.

`PGS_BASE=/pgs/postgres-8.2.5`

Specifies the PostgreSQL base directory, where the PostgreSQL binaries are located.

`PRI_GRP=primary-rg`

Specifies the resource group, which contains the cluster resource of the designated primary.

`STDBY_GRP=standby-rg`

Specifies the resource group, which contains the cluster resource of the designated standby.

`STDBY_RS=standby-rs`

Specifies the resource name of the designated standby resource. This name should be unique on your standby. The script `resilver-step2` requires this file generated by the script `resilver-step1` under `/var/tmp/${STDBY_RS}-resilver`.

`ROLECHG_GRP=rolechg-rg`

Specifies the resource group, which contains the Rolechanger resource.

`PRI_NODE=primary-host:primary-zone`

Specifies the node name or zone name of the designated primary host or zone.

`RSYNC=/usr/local/bin/rsync -rav`

Specifies the absolute path to RSYNC command including the necessary options.

`SSH_PASSPHRASE=false`

Specifies whether your ssh key is secured by a passphrase or not.

## Specifying Configuration Files for WAL File Shipping Without Shared Storage

You need three configuration files:

- A file for the PostgreSQL primary resource
- A file for the PostgreSQL standby resource
- A file for the Rolechanger resource for WAL file shipping without shared storage configuration

In addition to these requirements, you also need to customize copies of `resilver-step1` and `resilver-step2`.

The configuration files are as follows:

- `pgs_primary_config` for the primary resource
- `pgs_standby_config` for the standby resource
- `rolechg_config` for the Rolechanger resource
- Modified copy of the `resilver-step1` script
- Modified copy of the `resilver-step2` script

This example shows a `pgs_primary_config` file, a `pgs_standby_config` file, and a `rolechg_config` file with configuration parameters are set.

The key-value pairs and explanation for a sample `pgs_primary_config` file are follows:

*RS=postgres-prim-rs*

The name of the PostgreSQL resource is `postgres-prim-rs`.

*RG=postgres-prim-rg*

The name of the resource group for the PostgreSQL resource is `postgres-prim-rg`.

*PORT=80*

The value for the dummy port for the PostgreSQL resource is 80.

*LH=*

`SUNW.LogicalHostname` resource is not present in `postgres-sta-rg`.

*HAS\_RS=*

`SUNW.HASStoragePlus` resource is not present in `postgres-sta-rg`.

*PFILE=/postgres/pfile*

The parameter file is generated in `/postgres/pfile`.

*ZONE=*

Specifies a global zone configuration.

*ZONE\_BT=*

Specifies a global zone configuration.

*PROJECT=*

Specifies a global zone configuration.

*USER=pgs*

The name of the Oracle Solaris user who owns PostgreSQL is `pgs`.

*PGROOT=/postgres/postgresql-8.3.1*

The PostgreSQL software is installed in `/postgres/postgresql-8.3.1`.

*PGDATA=/postgres/data*

The PostgreSQL data and configuration files are installed under `/postgres/data`.

*PGPORT=5432*

The PostgreSQL database server listens on port 5432.

*PGLOGFILE=/postgres/logs/scinstance1*

The log file for the database server is `/postgres/logs/scinstance1`.

*LD\_LIBRARY\_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:*

The libraries for the PostgreSQL server are stored in the paths of

`LD_LIBRARY_PATH/usr/sfw/lib:/usr/local/lib:/usr/lib` directory.

*ENVSCRIPT=/postgres/variables.ksh*

Additional PostgreSQL variables are set in `/global/postgres/variables.ksh`.

*SCDB=testdb*

The monitored database is `testdb`.

SCUSER=*testusr*

The user for the database monitoring is *testusr*.

SCTABLE=*testtb1*

The table *testtb1* is modified to probe the condition of the database.

SCPASS=

The password for the user *testusr* is *testpwd*, although it is not specified.

NOCONRET=*10*

If a connection to the database *testdb* fails, the probe returns with return code *10*.

STDBY\_RS=*postgres-sta-rs*

The resource name of the PostgreSQL standby resource is *postgres-sta-rs*.

STDBY\_RG=*postgres-sta-rg*

The resource group name of the PostgreSQL standby resource group is *postgres-sta-rg*.

STDBY\_USER=*pgs*

The user who owns the PostgreSQL standby database is *pgs*.

STDBY\_HOST=*phys-node2*

The name of the standby host is *phys-node2*.

STDBY\_PARFILE=*/postgres/pfile*

The parameter file of the PostgreSQL standby resource is */postgres/pfile*.

ROLECHG\_RS=

The Rolechanger resource name has a null value because it is not needed on the primary.

SSH\_PASSDIR=

The *SSH\_PASSDIR* has a null value to indicate that the *sshkeys* are not protected by a passphrase.

The key-value pairs and explanation for a *pgs\_standby\_config* file are as follows:

RS=*postgres-sta-rs*

The name of the PostgreSQL resource is *postgres-sta-rs*.

RG=*postgres-sta-rg*

The name of the resource group for the PostgreSQL resource is *postgres-sta-rg*.

PORT=*80*

The value for the dummy port for the PostgreSQL resource is *80*.

LH=*postgres-sta-rg*

*SUNW.LogicalHostname* resource is not present in *postgres-sta-rg*.

HAS\_RS=

*SUNW.HASStoragePlus* resource is not present in *postgres-sta-rg*.

PFILE=*/postgres/pfile*

The parameter file is generated in */postgres/pfile*.

`ZONE=`

The null value indicates that it is a global zone configuration.

`ZONE_BT=`

The null value indicates that it is a global zone configuration.

`PROJECT=`

The null value indicates that it is a global zone configuration.

`USER=pgs`

The name of the Oracle Solaris user who owns PostgreSQL is *pgs*.

`PGROOT=/postgres/postgresql-8.3.1`

The PostgreSQL software is installed in */postgres/postgresql-8.3.1*.

`PGDATA=/postgres/data`

The PostgreSQL data and configuration files are installed under */postgres/data*.

`PGPORT=5432`

The PostgreSQL database server listens on port 5432.

`PGLOGFILE=/postgres/logs/scinstance1`

The log file for the database server is */postgres/logs/scinstance1*.

`LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:`

The libraries for the PostgreSQL server are stored in the paths of the `LD_LIBRARY_PATH`/*usr/sfw/lib: /usr/local/lib: /usr/lib:* directory.

`ENVSCRIPT=/postgres/variables.ksh`

Additional PostgreSQL variables are set in */global/postgres/variables.ksh*.

`SCDB=testdb`

The monitored database is *testdb*.

`SCUSER=testusr`

The user for the database monitoring is *testusr*.

`SCTABLE=testtbl`

The table *testtbl* is modified to probe the condition of the database.

`SCPASS=`

The password for the user *testusr* is *testpwd*, although it is not specified.

`NOCONRET=10`

If a connection to the database *testdb* fails, the probe returns with return code 10.

`STDBY_RS=`

The value for the `STDBY_RS` is not required in a standby configuration.

`STDBY_RG=`

The value for `STDBY_RG` is not required in a standby configuration.

STDBY\_USER=

The value for STDBY\_USER is not required in a standby configuration.

STDBY\_HOST=

The value for STDBY\_HOST is not required in a standby configuration.

STDBY\_PARFILE=

The value for STDBY\_PARFILE is not required in a standby configuration.

ROLECHG\_RS=*rolechg-rs*

The Rolechanger resource is *rolechg-rs*.

SSH\_PASSDIR=

The SSH\_PASSDIR has a null value, which means that the sshkeys are not protected by a passphrase.

The key-value pairs and explanation for configuration file *rolechg-config* are as follows:

RS=*rolechg-rs*

The name of the Rolechanger resource is *rolechg-rs*.

RG=*rolechg-rg*

The name of the resource group for the PostgreSQL resource is *rolechg-rg*.

PORT=5432

The value of the dummy port for the PostgreSQL resource is 5432.

LH=*pgs-1h-1*

The resource name for the SUNW.LoicalHostname resource is *pgs-1h-1*.

HAS\_RS=

SUNW.HASStoragePlus resource or other dependencies are not present.

STDBY\_RS=*postgres-sta-rs*

The name of the PostgreSQL standby resource is *postgres-sta-rs*.

PRI\_RS=*postgres-pri-rs*

The name of the PostgreSQL primary resource is *postgres-prim-rs*.

STDBY\_HOST=*phys-node*

The physical node name of the standby is *phys-node2*.

STDBY\_PFILE=*/postgres/pfile*

The parameter file on the standby is */postgres/pfile*.

TRIGGER=*/postgres/data/failover*

The trigger file on which the *pg\_standby* utility reacts is *phys-node2*.

WAIT=30

After the resource is started, Rolechanger waits for 30 seconds until it touches the trigger file.

Modifications in a copy of *resilver-step1*



`SOURCE_DATA=/postgres/data`

PGDATA of the standby is in `/postgres/data`.

`TARGET_DATA=/postgres/data`

PGDATA of the primary is in `/postgres/data`.

`TARGET=phys-node1`

Specifies the name of the target node. The usual name is the name of the designated primary node.

`PGS_BASE=/pgs/postgres-8.3`

Specifies the PostgreSQL base directory, where the PostgreSQL binaries are located.

`PRI_GRP=primary-rg`

Specifies the resource group that contains the cluster resource of the designated primary.

`STDBY_RS=standby-rs`

Specifies the resource group that contains the cluster resource of the designated standby.

`PGPORT=5432`

Specifies the database port.

`ROLECHG_GRP=rolechg-rg`

Specifies the resource group that contains the Rolechanger resource.

`RSYNC="/usr/local/bin/rsync -rsync-path=/usr/local/bin/rsync --rav"`

Specifies the absolute path to the RSYNC command, including the necessary options.

`SSH_PASSPHRASE=false`

Specifies whether your ssh key is secured by a passphrase or not.

Modifications in a copy of `resilver-step2`

`SOURCE=phys-node2`

The source node is `phys-node2`.

`SOURCE_DATA=/postgres/data`

PGDATA of the standby is in `/postgres/data`.

`TARGET_DATA=/postgres/data`

PGDATA of the primary is in `/postgres/data`.

`TARGET=phys-node1`

The target node is `phys-node1`.

`PGS_BASE=/user/postgres/8.3`

Specifies the PostgreSQL base directory, where the PostgreSQL binaries are located.

`PRI_GRP=primary-rg`

Specifies the resource group which contains the cluster resource of the designated primary.

`STDBY_GRP=postgres-sta-rg`

The resource group for the standby resource is `postgres-sta-rg`.

`STDBY_RS=standby-rs`

Specifies the resource group which contains the cluster resource of the designated standby.

`PGPORT=5432`

Specifies the database port.

`ROLECHG_GRP=rolechg-rg`

The resource group for the Rolechanger resource group is `rolechg-rg`.

`PRI_NODE=phys-node1`

The primary node is the global zone of `phys-node1`.

`RSYNC="/usr/local/bin/rsync -r -sync-path=/usr/local/bin/rsync -rav"`

Specifies the absolute path to the RSYNC command including the necessary options.

`SSH_PASSPHRASE=false`

Specifies whether your `ssh` key is secured by a passphrase or not.

## Preparing Your PostgreSQL Installation for Cluster Control

To prepare your PostgreSQL installation for cluster control, you create a database, a user, and a table to be monitored by the PostgreSQL resource. Because you need to differentiate between a global zone configuration and an HA for Solaris Zones configuration, two procedures are provided.

### ▼ How to Prepare Your PostgreSQL for Oracle Solaris Cluster Registration in the Global Zone

**Before You Begin** Ensure that you have edited the `pgs_config` file to specify configuration parameters for the HA for PostgreSQL data service. For more information, see [“Specifying Configuration Parameters for the PostgreSQL Resource”](#) on page 25.

- 1 As superuser change the rights of the configuration file to be accessible for your PostgreSQL user.

```
# chmod 755 /myplace/pgs_config
```

- 2 Switch to your PostgreSQL user.

```
# su - postgres
```

- 3 If the login shell is not the Korn shell, switch to `ksh`.

```
% ksh
```

- 4 Set the necessary variables.

```
$ . /myplace/pgs_config
$ export PGDATA PGPORT LD_LIBRARY_PATH
```

**5 If your PostgreSQL is not already running, start the PostgreSQL server.**

```
$ $PGRROOT/bin/pg_ctl -l $PGLLOGFILE start
```

**6 Prepare the database.**

- If you specified the password for the monitoring user in the configuration file or if you do not have a password, do the following:

```
$ /opt/SUNWscPostgreSQL/util/pgs_db_prep -f /myplace/pgs_config
```

- If you have a password and do not want to specify it in clear text, do the following:

```
$ /opt/SUNWscPostgreSQL/util/pgs_db_prep -f /myplace/pgs_config -e
```

**7 (Optional) Configure your PostgreSQL instance to listen on the logical host's TCP/IP name.**

If you want your PostgreSQL databases to listen on more than localhost, configure the `listen_address` parameter in the file `postgresql.conf`. Use a plain text editor such as `vi`, and set the value of `listen_address` to an appropriate value.



**Caution** – The PostgreSQL instance must listen on localhost. For additional information, see <http://www.postgresql.org>.

```
listen_address = 'localhost,myhost'
```

**8 Set the security policy for the test database.**

Use a plain text editor such as `vi` to add the following line to the file `pg_hba.conf`.

```
local testdb all password
```

**Note** – For additional information about the `pg_hba.conf` file, see <http://www.postgresql.org>.

**9 Stop the PostgreSQL database server.**

```
$ $PGRROOT/bin/pg_ctl stop
```

## ▼ How to Prepare Your PostgreSQL for Oracle Solaris Cluster Registration in an HA for Solaris Zones Configuration

**Before You Begin**

Ensure, that you have edited the `pgs_config` file to specify configuration parameters for the HA for PostgreSQL data service. For more information, see “[Specifying Configuration Parameters for the PostgreSQL Resource](#)” on page 25. Also make sure that the package directory of the HA for PostgreSQL, `/opt/SUNWscPostgreSQL`, is available in the target zone.

**1 As superuser change the rights of the configuration file to be accessible for your PostgreSQL user.**

---

**Note** – Ensure, that your `pgs_config` file is accessible from your zone. Otherwise, transfer the file to your zone by using appropriate methods.

---

```
# chmod 755 /myplace/pgs_config
```

**2 Switch to the target zone.**

```
# zlogin pgsq1-zone
```

**3 Switch to the PostgreSQL user.**

```
# su - zpostgr
```

**4 If the login shell is not the Korn shell, switch to ksh.**

```
% ksh
```

**5 Set the necessary variables.**

```
$ . /myplace/pgs_config
$ export PGDATA PGPORT LD_LIBRARY_PATH
```

**6 If your PostgreSQL is not already running, start the PostgreSQL server.**

```
$ $PGROOT/bin/pg_ctl -l $PGLOGFILE start
```

**7 Prepare the database.**

- If you specified the password for the monitoring user in the configuration file or if you do not have a password, do the following:

```
$ /opt/SUNWscPostgreSQL/util/pgs_db_prep -f /myplace/pgs_config
```

- If you have a password and do not want to specify it in clear text, do the following:

```
$ /opt/SUNWscPostgreSQL/util/pgs_db_prep -f /myplace/pgs_config -e
```

**8 (Optional) Configure your PostgreSQL instance to listen on the logical hosts TCP/IP name.**

If you want your PostgreSQL databases to listen on more than `localhost`, configure the `listen_address` parameter in the file `postgresql.conf`. Use a plain text editor such as `vi`, and set the value of `listen_address` to an appropriate value.




---

**Caution** – The PostgreSQL instance must listen on `localhost`. For additional information, see <http://www.postgresql.org>.

---

```
listen_address = 'localhost,myhost'
```

**9 Set the security policy for the test database.**

Use a plain text editor such as `vi` to add the following line to the `pg_hba.conf` file.

```
local testdb all password
```

---

**Note** – For additional information, see <http://www.postgresql.org>.

---

**10 Stop the PostgreSQL database server.**

```
$ $PGRROOT/bin/pg_ctl stop
```

**11 Leave the target zone and return to the global zone.**

## Creating and Enabling Resources for PostgreSQL

### ▼ How to Create and Enable Resources for PostgreSQL

**Before You Begin** Ensure that you have edited the `pgs_config` file to specify configuration parameters for the HA for PostgreSQL data service. For more information, see “[Specifying Configuration Parameters for the PostgreSQL Resource](#)” on page 25.

**1 Become superuser on one of the nodes in the cluster that will host PostgreSQL.****2 Go to the directory that contains the script for creating the HA for PostgreSQL resource.**

```
# cd /opt/SUNWscPostgreSQL/util
```

**3 Run the script that creates the PostgreSQL resource.**

- If you specified the password for the monitoring user in the configuration file or if you do not have a password, do the following:

```
# ksh ./pgs_register -f /myplace/pgs_config
```

- If you have a password and do not want to store it in clear text form, do the following:

- a. Perform the following command on all nodes to run the PostgreSQL resource.

```
# ksh ./pgs_register -f /myplace/pgs_config -e
```

- b. Perform the following command on the node where your shared storage is online.

```
# ksh ./pgs_register -f /myplace/pgs_config
```

If you omit the `-f` option, the file `/opt/SUNWscPostgreSQL/util/pgs_config` is used.

**4 Bring the PostgreSQL resource online.**

```
# clresource enable postgres-rs
```

## ▼ How to Modify Parameters in the HA for PostgreSQL Manifest

Perform this task to change parameters in the HA for PostgreSQL manifest and to validate the parameters in the HA container. Parameters for the HA for PostgreSQL manifest are stored as properties of the SMF service. To modify parameters in the manifest, change the related properties in the SMF service then validate the parameter changes.

- 1 Become superuser or assume a role that provides `solaris.cluster.modify` and `solaris.cluster.admin` RBAC authorizations on the zones console.
- 2 Change the Oracle Solaris Service Management Facilities (SMF) properties for the HA for PostgreSQL manifest.

```
# svccfg svc:/application/sczone-agents:resource
```

For more information, see the [svccfg\(1M\)](#) man page.

- 3 Validate the parameter changes.

```
# /opt/SUNWscPostgreSQL/bin/control_pgs validate resource
```

Messages for this command are stored in the `/var/adm/messages/` directory of the HA container.

- 4 Disconnect from the HA for Solaris Zones console.

## ▼ How to Remove a HA for PostgreSQL Resource From an HA Container

- 1 Become superuser or assume a role that provides `solaris.cluster.modify` and `solaris.cluster.admin` RBAC authorizations.
- 2 Disable and remove the resource that is used by the HA for PostgreSQL data service.
- 3 Log in as superuser to the HA for Solaris Zones console.
- 4 Unregister HA for PostgreSQL from the Oracle Solaris Service Management Facilities (SMF) service.

```
# /opt/SUNWscPostgreSQL/util/pgs_smf_remove -f filename
```

`-f` Specifies the configuration file name.

`filename` The name of the configuration file that you used to register HA for PostgreSQL with the SMF service.

---

**Note** – If you no longer have the configuration file that you used to register HA for PostgreSQL with the SMF service, create a replacement configuration file:

- a. Make a copy of the default file, `/opt/SUNWscPostgreSQL/util/pgs_config`.
  - b. Set the ZONE and RS parameters with the values that are used by the data service.
  - c. Run the `pgs_smf_remove` command and use the `-f` option to specify this configuration file.
- 

- 5 **Disconnect from the HA for Solaris Zones console.**

## ▼ How to Create and Enable Resources for PostgreSQL Rolechanger

**Before You Begin** Ensure that you have edited the `rolechg_config` file to specify configuration parameters for the HA for PostgreSQL Rolechanger data service. For more information, see <http://www.postgresql.org>.

- 1 **Become superuser on one of the nodes in the cluster that hosts PostgreSQL.**
- 2 **Go to the directory that contains the script for creating the HA for PostgreSQL Rolechanger resource.**

```
# cd /opt/SUNWscPostgreSQL/util
```

- 3 **Run the script that creates the PostgreSQL resource.**

```
# ksh ./rolechg_register -f /myplace/rolechg_config
```

If you omit the `-f` option, the file `/opt/SUNWscPostgreSQL/rolechg_util/rolechg_config` is used.

- 4 **Bring the PostgreSQL Rolechanger resource online.**

```
# clresource enable rolechg-rs
```

# Verifying the HA for PostgreSQL Installation and Configuration

After you install, register, and configure HA for PostgreSQL, verify this installation and configuration to determine whether the HA for PostgreSQL data service makes your PostgreSQL database highly available.

## ▼ How to Verify the HA for PostgreSQL Installation and Configuration

1 Become superuser on a cluster node that is to host the PostgreSQL component.

2 Ensure that all the PostgreSQL resources are online.

For each resource, perform the following steps:

a. Determine whether the resource is online.

```
# clresource status postgres-rs
```

b. If the resource is not online, bring the resource online.

```
# clresource enable postgres-rs
```

3 Switch the resource group to another cluster node, such as *node2*.

```
# clresourcegroup switch -h node2 postgres-rg
```

4 Confirm that the resource is now online on *node2*.

```
# clresource status postgres-rs
```

## ▼ How to Verify the Oracle Solaris Cluster HA for PostgreSQL WAL File Shipping Installation and Configuration

1 Become superuser on a cluster node that is to host the PostgreSQL component.

2 Ensure that all the PostgreSQL resources are online.

For each resource, perform the following steps:

a. Determine whether the resource is online.

```
# clresource status postgres-prim-rs  
# clresource status postgres-sta-rs  
# clresource status rolechg-rs
```

b. If the resource is not online, bring the resource online.

```
# clresource enable postgres-sta-rs  
# clresource enable postgres-prim-rs  
# clresource enable rolechg-rs
```



**3 Reboot the primary node where the primary database runs.**

```
# reboot
```

**4 Confirm that the resource is now online on *node 2*.**

```
# clresource status rolechg-rs
```

**5 For a failback, log in as the Postgres user on the standby node and resilver the primary node by executing scripts *resilver-step1* and *resilver-step2*.**

---

**Note** – Ensure that you follow the configuration steps for the scripts.

---

## Tuning the HA for PostgreSQL Fault Monitor

The HA for PostgreSQL fault monitor verifies that the data service is running in a healthy condition.

A HA for PostgreSQL fault monitor is contained in each resource that represents the PostgreSQL instance. You created these resources when you registered and configured HA for PostgreSQL. For more information, see [“Registering and Configuring HA for PostgreSQL” on page 24](#).

System properties and extension properties of the PostgreSQL resources control the behavior of the fault monitor. The default values of these properties determine the preset behavior of the fault monitor. Because the preset behavior should be suitable for most Oracle Solaris Cluster installations, tune the HA for PostgreSQL fault monitor *only* if you need to modify this preset behavior.

Tuning the HA for PostgreSQL fault monitor involves the following tasks:

- Setting the return value for failed PostgreSQL monitor connections
- Setting the interval between fault monitor probes
- Setting the time-out for fault monitor probes
- Defining the criteria for persistent faults
- Specifying the failover behavior of a resource

The fault monitor HA for PostgreSQL differentiates between connection problems and definitive application failures. The value of `NOCONRET` in the PostgreSQL parameter file specifies the return code for connection problems. This value results in a certain amount of ignored consecutive failed probes as long as they all return the value of `NOCONRET`. The first successful probe reverts this “failed probe counter” back to zero. The maximum number of failed probes is calculated as

$100 / \text{NOCONRET}$ . A definitive application failure will result in an immediate restart or failover.

The definition of the return value `NOCONRET` defines one of two behaviors for failed database connections of a PostgreSQL resource.

1. Retry the connection to the test database several times before considering the PostgreSQL resource as failed and triggering a restart or failover.
2. Complain at every probe that the connection to the test database failed. No restart or failover will be triggered.

To achieve either of these behaviors, you need to consider the standard resource properties `retry_interval` and `thorough_probe_interval`.

- A “just complaining” probe is achieved as soon as the following inequation is true:  
$$\text{retry\_interval} < \text{thorough\_probe\_interval} * 100 / \text{NOCONRET}$$
- As soon as this inequation is false, the PostgreSQL resource restarts or fails over after `100 / NOCONRET` consecutive probe failures.

The value `100/NOCONRET` defines the maximum number of retries for the probe in the case of a failed connection.

Assume that the following resource parameters are set:

- `thorough_probe_interval=60`
- `retry_interval=900`
- `NOCONRET=10`

If you encounter, for example, a shortage of available database sessions for 7 minutes, you will see 7 complaints in `/var/adm/messages`, but no resource restart. If the shortage lasts 10 minutes, you will have a restart of the PostgreSQL resource after the 10th probe.

If you do not want a resource restart in the previous example, set the value of `NOCONRET=10` to 5 or less.

For more information, see “[Tuning Fault Monitors for Oracle Solaris Cluster Data Services](#)” in *Oracle Solaris Cluster Data Services Planning and Administration Guide*.

## Operation of the HA for PostgreSQL Parameter File

The HA for PostgreSQL resources use a parameter file to pass parameters to the `start`, `stop`, and `probe` commands. Changes to these parameters take effect at least at every restart or enabling, disabling of the resource.

Changing one of the following parameters, takes effect at the next probe of the PostgreSQL resource:

- `USER`

- PGROOT
- PGPORT
- PGHOST
- LD\_LIBRARY\_PATH
- SCDB
- SCUSER
- SCTABLE
- SCPASS
- NOCONRET

---

**Note** – A false change of the parameters with an enabled PostgreSQL resource might result in an unplanned service outage. Therefore, disable the PostgreSQL resource first, execute the change, and then re-enable the resource.

---

## Operation of the Fault Monitor for HA for PostgreSQL

The fault monitor for HA for PostgreSQL ensures that all the requirements for the zone boot component to run are met:

- The HA for PostgreSQL main `postmaster` process is running.  
If this process is not running, the fault monitor restarts the PostgreSQL database server. If the fault persists, the fault monitor fails over the resource group that contains the resource for the PostgreSQL.
- Connections to the PostgreSQL database server are possible, and the database catalog is accessible.  
If the connection fails, the probe exits with the connection failed return code `NOCONRET`. If the database catalog is not accessible, the fault monitor restarts the PostgreSQL resource.
- The test database is healthy.  
If the test table in the test database can be manipulated, the database server is considered healthy. If table manipulation fails, it is differentiated, whether the problem was a connection error or the database manipulation was unsuccessful for any other reason.  
If the connection was impossible the probe exits with the connection failed return code `NOCONRET`. If the table manipulation itself was unsuccessful, the fault monitor triggers a restart or a failover the PostgreSQL database server resource.

# Debugging HA for PostgreSQL

HA for PostgreSQL has a file named `config` that enables you to activate debugging for PostgreSQL resources. This file is in the `/opt/SUNWscPostgreSQL/etc` directory.

## ▼ How to Activate Debugging for HA for PostgreSQL

### 1 Determine whether you are in the global zone or in an HA for Solaris Zones configuration.

If your PostgreSQL resource is dependent on an Oracle Solaris Zones boot component resource, you are in an HA for Solaris Zones configuration. In any other case, you are in a global zone configuration.

### 2 Determine whether debugging for HA for PostgreSQL is active.

```
# grep daemon /etc/syslog.conf
*.err;kern.debug;daemon.notice;mail.crit      /var/adm/messages
*.alert;kern.err;daemon.err                    operator
```

If debugging is inactive, `daemon.notice` is set in the file `/etc/syslog.conf` of the appropriate zone.

### 3 If debugging is inactive, edit the `/etc/syslog.conf` file in the appropriate zone to change `daemon.notice` to `daemon.debug`.

### 4 Confirm that debugging for HA for PostgreSQL is active.

If debugging is active, `daemon.debug` is set in the file `/etc/syslog.conf`.

```
# grep daemon /etc/syslog.conf
*.err;kern.debug;daemon.debug;mail.crit      /var/adm/messages
*.alert;kern.err;daemon.err                  operator
```

### 5 Restart the `syslogd` daemon in the appropriate zone.

```
# svcadm refresh svc:/system/system-log:default
```

### 6 Edit the `/opt/SUNWscPostgreSQL/etc/config` file to change the `DEBUG=` variable according to one of the examples:

- `DEBUG=ALL`
- `DEBUG=resource name`
- `DEBUG=resource name,resource name,...`

```
# cat /opt/SUNWscPostgreSQL/etc/config
#
# Copyright (c) 2006,2012, Oracle and/or its affiliates. All rights reserved.
#
# Usage:
#     DEBUG=<RESOURCE_NAME> or ALL
```

```
#  
DEBUG=ALL  
#
```

---

**Note** – To deactivate debugging, repeat Steps 1 to 6, changing `daemon.debug` to `daemon.notice` and changing the `DEBUG` variable to `DEBUG=`.

---



## Files for Configuring Oracle Solaris Cluster HA for PostgreSQL Resources

---

The /opt/SUNWscPostgreSQL/util directory contains files that automate the process of configuring HA for PostgreSQL resources. These files include a registration script, a database preparation script, a configuration file to provide parameters for the first two scripts and a template for the rolechg\_config file. This appendix shows a listing of the configuration files.

### Listing of pgs\_config

```
#
# Copyright (c) 2006,2012, Oracle and/or its affiliates. All rights reserved.
#
# This file will be sourced in by pgs_register and the parameters
# listed below will be used.
#
# These parameters can be customized in (key=value) form
#
# RS - name of the resource for the application.
# RG - name of the resource group containing RS.
# PORT - name of the port number.
# Do not set the PORT variable if you plan to have a network
# unaware installation, or an installation in a
# HA for Solaris Zones.
# LH - name of the LogicalHostname SC resource.
# Do not set the LH variable if you plan to have a network
# unaware installation, or an installation in a
# HA for Solaris Zones.
# HAS_RS - Name of the HAStoragePlus SC resource.
# PFILE - Parameter file which contains the PostgreSQL specific
# parameters, this file will be created by the register script.
#
# The following variables need to be set only, if the agent runs in a
# HA for Solaris Zones
#
# ZONE - Zonename where the zsmf component should be registered
# ZONE_BT - Resource name of the zone boot component
# PROJECT - A project in the zone, that will be used for the PostgreSQL
# smf service.
# If the variable is not set it will be translated as :default for
```

```
#           the smf credentials.
#           Optional
#

RS=
RG=
PORT=
LH=
HAS_RS=
PFILE=

# HA container specific options

ZONE=
ZONE_BT=
PROJECT=

#
# Content for the parameter file
#
#       USER - The Solaris user which owns the PostgreSQL database.
#       PGTROOT - Contains the path to the PostgreSQL directory. Below this
#                directory the postgres binaries are located in the ./bin
#                directory.
#       PGDATA - Contains the path to the databases of this specific PostgreSQL
#                instance.
#       PGPORT - Port where the postmaster process will be listening.
#       PGHOST - Hostname where the postmaster process is listening, or a directory
#                where the Unix socket file is stored.
#                If set to a valid hostname, the PGHOST variable forces the probe to
#                traverse the TCP/IP stack. If the PGHOST variable is empty
#                or starts with a "/",
#                the probe will use a socket. If the PGHOST variable starts with a
#                "/", the entry must
#                be the directory which contains the socket file.
#       PGLOGFILE - Logfile where the log messages of the postmaster will be stored.
#       LD_LIBRARY_PATH - This path contains all the necessary libraries for this PostgreSQL
#                installation.
#                Optional
#       ENVSCRIPT - Script to contain PostgreSQL specific runtime variables.
#                Optional
#       SCDB - This database will be monitored. The database will be generated at
#                database preparation time.
#       SCUSER - PostgreSQL user to connect to the $SCDB database. The user will
#                be generated at database preparation time
#       SCTABLE - Table name in the $SCDB database. This table name will be
#                manipulated to check if PostgreSQL is alive. This table will be
#                generated at database preparation time.
#       SCPASS - Password of the SCUSER. If no password is provided, the authentication method
#                for the SCDB database needs to be trusted for requests from the localhost.
#                If you do not want to store a readable password in a file, leave the SCPASS
#                variable empty and encrypt it with the pgs_register -f <config-file> -e command.
#                Optional
#       NOCONRET - Return code for connection errors. This return code has to
#                follow the rules for the generic data service. The value has
#                to be between 1 and 100.
#                100/NOCONRET defines the number of consecutive probes to ignore for
#                failed connections. A restart or failover will occur, if the
#                number is exceeded within the retry interval.
```



```

USER=
PGROOT=
PGDATA=
PGPORT=
PGHOST=
PGLOGFILE=
LD_LIBRARY_PATH=
ENVSCRIPT=
SCDB=
SCUSER=
SCTABLE=
SCPASS=
NOCONRET=10

# The following parameters need to be configured only if logfile shipping is configured to
# ship the PostgreSQL WAL logs between a designated primary and a designated standby
# resource.
# They needed to be configured only by the primary.
#
# These parameters can be customized in (key=value) form
#
#           STDBY_RS - The resource name of the PostgreSQL standby resource.
#           STDBY_RG - The resource group name of the
#           PostgreSQL resource group.
#           STDBY_USER - User which is the owner of the standby postgres database.
#           STDBY_HOST - Resolvable name of the standby host or the standby zone.
#                   This name has to be reached through SSH
# STDBY_PARFILE - The standby postgres parameter file to get the rest
#                   of the necessary parameters.
#           STDBY_PING - The number of packets the primary uses to ping the
#                   standby host.If this variable is empty , it will be
#                   set to five packets.

#           ROLECHG_RS - The resource name of the rolechanger.
# SSH_PASSDIR - A directory where the ssh passphrase is stored in file
#                   resourcename-sshpass.
#                   This parameter is required only if you configured WAL
#                   shipping and secured your SSH key with a passphrase.
#                   If the passphrase is empty, leave it undefined.
#                   If you configure the logfile shipping in a without
#                   storage configuration, do not set the LH parameter
#
# Configure the following parameters on the primary host.
STDBY_RS=
STDBY_RG=
STDBY_USER=
STDBY_HOST=
STDBY_PARFILE=
STDBY_PING=
#
# Configure the following parameters on the standby host

ROLECHG_RS=
#
# Configure the following parameter on both hosts.
#
SSH_PASSDIR=

```

## Listing of rolechg\_config

```
#
# CDDL HEADER START
#
#The contents of this file are subjected to the terms of the Common Development and Distribution
#License (the License).
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/CDDL.txt
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/CDDL.txt.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets [] replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner] Use is subject to license terms.
#
# CDDL HEADER END
#

#
#Copyright (c) 2006,2012, Oracle and/or its affiliates. All rights reserved.
#

# ident    "@(#)rolechg_config.ksh 1.2    08/05/06"
#
# This file will be sourced in by rolechg_register and the parameters
# listed below will be used.
#
# These parameters can be customized in (key=value) form
#
#           RS - name of the resource for the application.
#           RG - name of the resource group containing RS.
#           PORT - name of the port number.
#           LH - name of the LogicalHostname SC resource.
#               Do not set the LH variable if you plan to have a network
#               unaware installation.
#           HAS_RS - Name of the HASToragePlus SC resource.
#           STDBY_RS - The resource name of designated standby database
#           PRI_RS - The resource name of designated primary database
#           STDBY_HOST - Hostname or zonename of the standby host. If empty, a role switch
#                       will be initiated on any host.
#           SDBY_PFILE - Parameter file which contains the PostgreSQL specific
#                       parameters for the standby database. This file is mentioned in
#                       the Start_command of the PostgreSQL standby resource.
#           TRIGGER - The filename which will get created to tell pg_standby to end the
#                       recovery mode, this filename is mentioned in the recovery.conf file
#                       of the PostgreSQL standby database.
#           WAIT - The number of seconds the start method waits before it touches
#                  the trigger file. This little break is necessary because, if the
#                  trigger file should be touched before, or in the middle of the
#                  PostgreSQL start process, it would get removed automatically.
```

RS=

```
RG=  
PORT=  
LH=  
HAS_RS=  
STDBY_RS=  
PR_I_RS=  
STDBY_HOST=  
STDBY_PFILE=  
TRIGGER=  
WAIT=
```



# Deployment Example: Installing PostgreSQL in the Global Zone or a Zone Cluster

---

This appendix presents a complete example of how to install and configure the PostgreSQL application and data service in the global zone or in a zone cluster. It presents a simple two-node cluster configuration. If you need to install the application in any other configuration, refer to the general-purpose procedures presented elsewhere in this manual.

For an example of PostgreSQL installation in a non-global zone that is configured with HA for Solaris Zones, see [Appendix C, “Deployment Example: Installing PostgreSQL in a Non-Global Zone With HA for Solaris Zones.”](#)

## Target Cluster Configuration

This example uses a two-node cluster with the following node names:

- phys-schost-1 (a physical node, which owns the file system)
- phys-schost-2 (a physical node)

This configuration also uses the logical host name ha-host-1.

## Software Configuration

This deployment example uses the following software products and versions:

- Oracle Solaris 11 software for SPARC or x86 platforms
- Oracle Solaris Cluster 4.1 core software
- Oracle Solaris Cluster Data Service for PostgreSQL
- PostgreSQL version 8.1.0 source files
- readline
- gmake
- Your preferred text editor
- Your preferred C compiler

This example assumes that you have already installed and established your cluster. It illustrates installation and configuration of the data service application only.

## Assumptions

The instructions in this example were developed with the following assumptions:

- **Shell environment:** All commands and the environment setup in this example are for the Korn shell environment. If you use a different shell, replace any Korn shell-specific information or instructions with the appropriate information for your preferred shell environment.
- **User login:** Unless otherwise specified, perform all procedures as superuser or assume a role that provides `solaris.cluster.admin`, `solaris.cluster.modify`, and `solaris.cluster.read` RBAC authorization.

## Installing and Configuring PostgreSQL on Shared Storage in the Global Zone or a Zone Cluster

The tasks you must perform to install and configure PostgreSQL in the global zone or zone cluster are as follows:

- “[Example: Preparing the Cluster for PostgreSQL](#)” on page 62
- “[Example: Configuring Cluster Resources for PostgreSQL](#)” on page 63
- “[Example: Modifying the PostgreSQL Configuration File](#)” on page 63
- “[Example: Building and Installing the PostgreSQL Software on Shared Storage](#)” on page 64
- “[Example: Enabling the PostgreSQL Software to Run in the Cluster](#)” on page 65

### ▼ Example: Preparing the Cluster for PostgreSQL

- 1 **Install and configure the cluster as instructed in [Oracle Solaris Cluster Software Installation Guide](#).**

Install the following cluster software components on both nodes.

- Oracle Solaris Cluster core software
- Oracle Solaris Cluster data service for PostgreSQL

- 2 **Install the following utility software on both nodes:**

- `readline`
- `gmake`
- Your C compiler

**3 Beginning on the node that owns the file system, add the postgres user.**

```
phys-schost-1# groupadd -g 1000 postgres
phys-schost-2# groupadd -g 1000 postgres
phys-schost-1# useradd -g 1000 -d /global/mnt3/postgres -s /bin/ksh postgres
phys-schost-2# useradd -g 1000 -d /global/mnt3/postgres -s /bin/ksh postgres
```

**▼ Example: Configuring Cluster Resources for PostgreSQL****1 Register the necessary data types on both nodes.**

```
phys-schost-1# clresourcetype register SUNW.gds SUNW.HASStoragePlus
```

**2 Create the PostgreSQL resource group.**

```
phys-schost-1# clresourcegroup create RP-PGS
```

**3 Create the logical host.**

```
phys-schost-1# clreslogicalhostname create -g RG-PGS ha-host-1
```

**4 Create the HASStoragePlus resource in the RG-PGS resource group.**

```
phys-schost-1# clresource create -g RG-PGS -t SUNW.HASStoragePlus -p AffinityOn=TRUE \
-p FilesystemMountPoints=/global/mnt3,/global/mnt4 RS-PGS-HAS
```

**5 Enable the resource group.**

```
phys-schost-1# clresourcegroup online -M RG-PGS
```

**▼ Example: Modifying the PostgreSQL Configuration File****1 Modify the PGR00T and PD\_LIBRARY\_PATH environment variables according to the needs of your build.**

The databases are stored under /global/mnt3/postgres/data.

The log is stored under /global/mnt3/postgres/logs/sclog.

```
phys-schost-1# PGR00T=/global/mnt3/postgres/postgresql-8.1.0
```

```
phys-schost-1# LD_LIBRARY_PATH=/global/mnt3/postgres/postgresql-8.1.0/lib \
/usr/sfw/lib:/usr/local/lib/usr/lib:/opt/csw/lib
```

```
phys-schost-1# export PG_ROOT
```

```
phys-schost-1# export LD_LIBRARY_PATH
```

If you are installing the software in the default directory, set PGR00T to `/usr/local/pgsql` and LD\_LIBRARY\_PATH to `/usr/local/pgsql/lib:/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib`.

- 2 **Copy the PostgreSQL configuration file from the agent directory to its deployment location.**

```
phys-schost-1# cp /opt/SUNWscPostgreSQL/util/pgs_config /global/mnt3
```

- 3 **Add this cluster's information to the configuration file.**

The following listing shows the relevant file entries and the values to assign to each entry.

```
...
RS=RS-PGS
RG=RG-PGS
PORT=5432
LH=hahostix1
HAS_RS=RS-PGS-HAS
PFILE=/global/mnt3/postgres/RS-PGS-pfile
...
USER=postgres
PGR00T=/usr/local/pgsql
#PGR00T=/global/mnt3/postgres/postgresql-8.1.0
PGDATA=/global/mnt3/postgres/data
PGPORT=5432
PGHOST=
PGLOGFILE=/global/mnt3/postgres/logs/sclog
LD_LIBRARY_PATH=/usr/local/pgsql/lib:/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
#LD_LIBRARY_PATH=/global/mnt3/postgres/postgresql-8.1.0/lib:/usr/sfw/lib
#LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:/usr/lib:/opt/csw/lib
SCDB=sctest
SCUSER=scusser
SCTABLE=sctable
SCPASS=
```

- 4 **Save and close the file.**

## ▼ Example: Building and Installing the PostgreSQL Software on Shared Storage

These steps illustrate how to install the PostgreSQL software on shared storage. You can also build and install the PostgreSQL binaries in the default directory `/usr/local/pgsql`.

- 1 **Create the home directory for PostgreSQL user.**

```
phys-schost-1# mkdir /global/mnt3/postgres
```

- 2 **Change the ownership of the *postgres* directory.**

```
phys-schost-1# chown -R postgres:postgres /global/mnt3/postgres
```

- 3 **Log in as the PostgreSQL user.**

```
phys-schost-1# su - postgres
```



**4 Set up the build environment.****a. Create a build directory.**

```
phys-schost-1$ mkdir build
phys-schost-1$ cd build
```

**b. Add the C compiler and ar to your PATH.**

```
phys-schost-1$ PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
phys-schost-1$ export PATH
```

**5 Install the source and configure the build.**

```
phys-schost-1$ gzcat /tmp/postgresql-8.1.0.tag.gz | tar xvf -
phys-schost-1$ cd /global/mnt3/postgres/build/postgresql-8.1.0
phys-schost-1$ ./configure --prefix=/global/mnt3/postgres/postgresql-8.1.0
```

**6 Build the PostgreSQL binaries.**

```
phys-schost-1$ make
```

If you use gcc to build the postgres binaries, build them in a failover file system.

**7 Run the PostgreSQL regression tests.**

```
phys-schost-1$ make check
```

**8 Install the PostgreSQL binaries.**

```
phys-schost-1# make install
```

**9 Clean the distribution.**

```
phys-schost-1$ make clean
```

## ▼ Example: Enabling the PostgreSQL Software to Run in the Cluster

**1 Create the directories for the databases and the log file.**

```
phys-schost-1$ mkdir /global/mnt3/postgres/data
phys-schost-1$ mkdir /global/mnt3/postgres/logs
```

**2 Change to the PostgreSQL root directory and initialize the data cluster.**

```
phys-schost-1$ cd /global/mnt3/postgres/postgresql-8.1.0
phys-schost-1$ ./bin/initdb -D /global/mnt3/postgres/data
```

**3 Start the database.**

```
phys-schost-1$ ./bin/postmaster -D /global/mnt3/postgresql-8.1.0
```

- 4 **Prepare the Oracle Solaris Cluster-specific test database.**  

```
phys-schost-1$ ksh /opt/SUNWscPostgreSQL/util/pgs_db_prep -f \  
/global/mnt3/pgs_config -e
```
- 5 **Stop the postmaster.**  

```
phys-schost-1$ ./bin/pg_ctl -D /global/mnt3/data stop
```
- 6 **Exit the postgres user ID.**  

```
phys-schost-1$ exit
```
- 7 **Encrypt the password of the monitoring user on all nodes, to run the PostgreSQL resource.**  

```
phys-schost-1$ ksh /opt/SUNWscPostgreSQL/util/pgs_register -f \  
/global/mnt3/pgs_config -e  
phys-schost-2$ ksh /opt/SUNWscPostgreSQL/util/pgs_register \  
/global/mnt3/pgs_config -e
```
- 8 **Run the pgs\_register script to register the resource.**  

```
phys-schost-1# ksh /opt/SUNWscPostgreSQL/util/pgs_register -f /global/mnt3/pgs_config
```
- 9 **Add the following line to the postgresql.conf file in the PGDATA directory.**  

```
listen_addresses = 'localhost,ha-host-1'
```
- 10 **Add the following line to the pg\_hba.conf file in the PGDATA directory.**  

```
host    all             all             0.0.0.0/0       password
```
- 11 **Enable the resource.**  

```
phys-schost-1# clresource enable RS-PGS
```

## Installing the PostgreSQL Binaries in the Default Directory (Alternative Installation)

The instructions in [“Installing and Configuring PostgreSQL on Shared Storage in the Global Zone or a Zone Cluster” on page 62](#) install the PostgreSQL software on shared cluster storage. You can also install this software in the default directory `/usr/local/postgresql`.

To install the PostgreSQL software in the default directory, perform the steps provided in the following example procedures:

- [“Example: Preparing the Cluster for PostgreSQL” on page 62](#)
- [“Example: Configuring Cluster Resources for PostgreSQL” on page 63](#)
- [“Example: Modifying the PostgreSQL Configuration File” on page 63](#)
- [“Example: Building and Installing the PostgreSQL Software in the Default Directory in the Global Zone” on page 67](#)
- [“Example: Enabling the PostgreSQL Software to Run in the Cluster” on page 65](#)

## ▼ Example: Building and Installing the PostgreSQL Software in the Default Directory in the Global Zone

These steps illustrate how to install the PostgreSQL software in the default directory `/usr/local/pgsql`. You can also build and install the PostgreSQL binaries on shared storage. See [“Installing and Configuring PostgreSQL on Shared Storage in the Global Zone or a Zone Cluster”](#) on page 62.

- 1 **Create the home directory for PostgreSQL user.**

```
phys-schost-1# mkdir /global/mnt3/postgres
```

- 2 **Change the ownership of the `postgres` directory.**

```
phys-schost-1# chown -R postgres:postgres /global/mnt3/postgres
```

- 3 **Log in as the PostgreSQL user.**

```
phys-schost-1# su - postgres
```

- 4 **Expand the software tar file.**

```
phys-schost-1$ gzip /tmp/postgresql-8.1.0.tar.gz |tar xvf -
```

- 5 **Add the C compiler and `ar` to your `PATH`.**

```
phys-schost-1$ PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
phys-schost-1$ export PATH
```

- 6 **Add the C compiler and readline libraries to your `LD_LIBRARY_PATH`.**

```
phys-schost-1$ LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
phys-schost-1$ export LD_LIBRARY_PATH
```

- 7 **Install the source and configure the build.**

```
phys-schost-1$ gzip /tmp/postgresql-8.1.0.tar.gz |tar xvf -
phys-schost-1$ cd /global/mnt3/postgres/build/postgresql-8.1.0
phys-schost-1$ ./configure
```

- 8 **Build the PostgreSQL binaries.**

```
phys-schost-1$ gmake
```

If you use `gcc` to build the `postgres` binaries, build them in a failover file system.

- 9 **Run the PostgreSQL regression tests.**

```
phys-schost-1$ gmake check
```

- 10 **Log back in as root.**

```
phys-schost-1$ su
```

- 11 **Add the C compiler and `ar` to your `PATH`.**

This example assumes the following:

- The compiler is `gcc`, located in `/usr/sfw/bin`.
- `ar` is located in `/usr/ccs/bin`.

```
phys-schost-1# PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
phys-schost-1# export PATH
```

**12 Add the C compiler and readline libraries to your `LD_LIBRARY_PATH`.**

```
phys-schost-1# LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
phys-schost-1# export LD_LIBRARY_PATH
```

**13 Install the binaries.**

```
phys-schost-1# gmake install
```

**14 Copy the binaries to the second node.**

```
phys-schost-1# scp -rp /usr/local/pgsql phys-schost-2:/usr/local
```

**15 Exit from root access.**

```
phys-schost-1# exit
```

**16 Clean the distribution.**

```
phys-schost-1% gmake clean
```

**Next Steps** Perform the steps in [“Example: Enabling the PostgreSQL Software to Run in the Cluster”](#) on [page 65](#) to complete installation and configuration of PostgreSQL.

# Deployment Example: Installing PostgreSQL in a Non-Global Zone With HA for Solaris Zones

---

This appendix presents a complete example of how to install and configure the PostgreSQL application and data service in a non-global zone that is configured with Oracle Solaris Cluster HA for Solaris Zones. It presents a simple two-node cluster configuration. If you need to install the application in any other configuration, refer to the general-purpose procedures presented elsewhere in this manual.

For an example of PostgreSQL in the global zone, see [Appendix B, “Deployment Example: Installing PostgreSQL in the Global Zone or a Zone Cluster.”](#)

## Target Cluster Configuration

This example uses a two-node cluster with the following node names:

- phys-schost-1 (a physical node, which owns the file system)
- phys-schost-2 (a physical node)

## Software Configuration

This deployment example uses the following software products and versions:

- Oracle Solaris 11 software for SPARC or x86 platforms
- Oracle Solaris Cluster 4.1 core software
- Oracle Solaris Cluster Data Service for PostgreSQL
- Oracle Solaris Cluster Data Service for Oracle Solaris Zones
- PostgreSQL version 8.1.0 source files
- readline
- gmake
- Your preferred text editor
- Your preferred C compiler

This example assumes that you have already installed and established your cluster. It illustrates installation and configuration of the data service application only.

## Assumptions

The instructions in this example were developed with the following assumptions:

- **Shell environment:** All commands and the environment setup in this example are for the Korn shell environment. If you use a different shell, replace any Korn shell-specific information or instructions with the appropriate information for your preferred shell environment.
- **User login:** Unless otherwise specified, perform all procedures as superuser or assume a role that provides `solaris.cluster.admin`, `solaris.cluster.modify`, and `solaris.cluster.read` RBAC authorization.

## Installing and Configuring PostgreSQL on Shared Storage in a Non-Global Zone with HA for Solaris Zones

These instructions assume that you are installing the PostgreSQL software as the `postgres` user in a shared directory. For instructions on installing the software in the default directory `/usr/local/pgsql`, see [“Installing the PostgreSQL Binaries in the Default Directory in a Non-Global Zone With HA for Solaris Zones \(Alternative Installation\)”](#) on page 76.

The tasks you must perform to install and configure PostgreSQL in the global zone are as follows:

- [“Example: Preparing the Cluster for PostgreSQL”](#) on page 70
- [“Example: Configuring Cluster Resources for PostgreSQL”](#) on page 71
- [“Example: Configuring HA for Solaris Zones”](#) on page 71
- [“Example: Modifying the PostgreSQL Configuration File”](#) on page 73
- [“Example: Building and Installing the PostgreSQL Software on Shared Storage in a Non-Global Zone With HA for Solaris Zones”](#) on page 74
- [“Example: Enabling the PostgreSQL Software to Run in the Cluster”](#) on page 75

### ▼ Example: Preparing the Cluster for PostgreSQL

- 1 **Install and configure the cluster as instructed in** [Oracle Solaris Cluster Software Installation Guide](#).

Install the following cluster software components on both nodes.

- Oracle Solaris Cluster core software
- Oracle Solaris Cluster data service for PostgreSQL
- Oracle Solaris Cluster data service for Oracle Solaris Zones

**2 Install the following utility software on both nodes:**

- readLine
- gmake
- Your C compiler

## ▼ Example: Configuring Cluster Resources for PostgreSQL

This example is based upon “[How to Enable a PostgreSQL Database to Run in the Global Zone](#)” on page 17.

**1 Register the HAStoragePlus resource type.**

```
phys-schost-1# clresourcetype register SUNW.gds SUNW.HAStoragePlus
```

**2 Create the PostgreSQL resource group.**

```
phys-schost-1# clresourcegroup create RP-PGS
```

**3 Create the HAStoragePlus resource in the RG-PGS resource group.**

```
phys-schost-1# clresource create -g RG-PGS -t SUNW.HAStoragePlus -p AffinityOn=TRUE \
-p FilesystemMountPoints=/global/mnt3,/global/mnt4 RS-PGS-HAS
```

**4 Enable the resource group.**

```
phys-schost-1# clresourcegroup online -M RG-PSG
```

## ▼ Example: Configuring HA for Solaris Zones

**1 On shared cluster storage, create a directory for the HA for Solaris Zones root path.**

```
phys-schost-1# mkdir /global/mnt3/zones
```

**2 Create a temporary file, for example /tmp/x, and include the following entries:**

```
create -b
set zonepath=/global/mnt3/zones/clu1
set autoboot=false
set pool=pool_default
add net
set address=hahostix1
```

```
set physical=net1
end
add attr
set name=comment
set type=string
set value="PostgreSQL cluster zone"      Put your desired zone name between the quotes here.
end
```

**3 Configure HA for Solaris Zones, using the file you created.**

```
phys-schost-1# zonecfg -z clu1 -f /tmp/x
```

**4 Install the non-global zone.**

```
phys-schost-1# zoneadm -z clu1 install
```

**5 Log in to the zone.**

```
phys-schost-1# zlogin -C clu1
```

**6 Open a new window to the same node and boot the zone.**

```
phys-schost-1a# zoneadm -z clu1 boot
```

**7 Close this terminal window and disconnect from the zone console.**

```
phys-schost-1# ~~.
```

**8 Copy the zones configuration file to a temporary location.**

```
phys-schost-1# cp /opt/SUNWsczone/sczbt/util/sczbt_config /tmp/sczbt_config
```

**9 Edit the /tmp/sczbt\_config file and set variable values as shown:**

```
RS=RS-PGS-ZONE
RG=RG-PGS
PARAMETERDIR=/global/mnt3/zonepar
SC_NETWORK=false
SC_LH=
FAILOVER=true
HAS_RS=RS-PGS-HAS
```

```
Zonename=clu1
Zonebootopt=
Milestone=multi-user-server
Mounts=
```

**10 Create the zone according to the instructions in the [Oracle Solaris Cluster Data Service for Oracle Solaris Zones Guide](#).**

**11 Register the zone resource.**

```
phys-schost-1# ksh /opt/SUNWsczone/sczbt/util/sczbt_register -f /tmp/sczbt_config
```

**12 Enable the zone resource.**

```
phys-schost-1# clresource enable RS-PGS-ZONE
```



## ▼ Example: Modifying the PostgreSQL Configuration File

- 1 **Modify the PGRROOT and PD\_LIBRARY\_PATH environment variables according to the needs of your build.**

The databases are stored under `/global/mnt3/postgres/data`.

The log is stored under `/global/mnt3/postgres/logs/sclog`.

```
phys-schost-1# PG_ROOT=/global/mnt3/postgres/postgresql-8.1.0
phys-schost-1# LD_LIBRARY_PATH=/global/mnt3/postgres/postgresql-8.1.0/lib \
phys-schost-1# LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/sfw/lib:/usr/local/lib: \
/usr/lib:/opt/csw.lib
phys-schost-1# export LD_LIBRARY_PATH PG_ROOT
```

- 2 **Store the pfile in a directory in the zone clu1.**

The configuration file name must be available in the zone.

- 3 **Copy the PostgreSQL configuration file from the agent directory to its deployment location.**

```
phys-schost-1# cp /opt/SUNWscPostgreSQL/util/pgs_config /global/mnt3
```

- 4 **Add this cluster's information to the configuration file.**

The following listing shows the relevant file entries and the values to assign to each entry.

```
...
RS=RS-PGS
RG=RG-PGS
PORT=5432
LH=hahostix1
HAS_RS=RS-PGS-HAS
PFILE=/global/mnt3/postgres/RS-PGS-pfile
...
# local zone specific options
ZONE=clu1
ZONE_BT=RS-PGS-ZONE
ZUSER=postgres
PROJECT=
...
USER=postgres
PGRROOT=/usr/local/pgsql
#PGRROOT=/global/mnt3/postgres/postgresql-8.1.0
PGDATA=/global/mnt3/postgres/data
PGPORT=5432
PGHOST=
PGLOGFILE=/global/mnt3/postgres/logs/sclog
LD_LIBRARY_PATH=/usr/local/pgsql/lib:/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
#LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
#LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
ENVSCRIPT=
SCDB=sctest
SCUSER=scuser
SCTABLE=sctable
SCPASS=
```

- 5 Save and close the file.
- 6 Transfer this configuration file in the zone `clu1` under `/tmp/pgs_config`.  

```
phys-schost-1# scp /global/mnt3/pgs_config clu1:/tmp
```

## ▼ Example: Building and Installing the PostgreSQL Software on Shared Storage in a Non-Global Zone With HA for Solaris Zones

This example illustrates how to install the PostgreSQL software on shared storage. You can alternatively build and install the PostgreSQL binaries in the default directory `/usr/local/postgresql`. See [“Installing the PostgreSQL Binaries in the Default Directory in a Non-Global Zone With HA for Solaris Zones \(Alternative Installation\)”](#) on page 76.

- 1 Log in to the non-global zone.

```
phys-schost-1# zlogin clu1
```

- 2 Add the postgres user.

```
zone# groupadd -g 1000 postgres
zone# useradd -g 1000 -u 1006 -d /postgres -m -s /bin/ksh postgres
```

- 3 Log in as the PostgreSQL user.

```
zone# su - postgres
```

- 4 Set up the build environment.

- a. Create a build directory.

```
zone$ mkdir build
zone$ cd build
```

- b. Add the C compiler and ar to your PATH.

```
zone PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
zone$ export PATH
```

- 5 Install the source and configure the build.

```
zone$ gzcatt /tmp/postgresql-8.1.0.tag.gz | tar xvf -
zone$ cd /global/mnt3/postgres/build/postgresql-8.1.0
zone$ ./configure --prefix=/global/mnt3/postgres/postgresql-8.1.0
```

- 6 Build the PostgreSQL binaries.

```
zone$ gmake
```

If you use `gcc` to build the `postgres` binaries, build them in a failover file system.

**7 Run the PostgreSQL regression tests.**

```
zone$ gmake check
```

**8 Install the PostgreSQL binaries.**

```
zone$ gmake install
```

**9 Clean the distribution.**

```
zone$ gmake clean
```

## ▼ Example: Enabling the PostgreSQL Software to Run in the Cluster

**1 Create the directories for the databases and the log file.**

```
zone$ mkdir /global/mnt3/postgres/data
zone$ mkdir /global/mnt3/postgres/logs
```

**2 Change to the PostgreSQL root directory and initialize the data cluster.**

```
zone$ cd /postgres/postgresql-8.1.0
zone$ ./bin/initdb -D postgres/data
```

**3 Start the database.**

```
zone$ ./bin/postmaster -D /postgresql-8.1.0
```

**4 Prepare the Oracle Solaris Cluster-specific test database.**

```
zone$ ksh /opt/SUNWscPostgreSQL/util/pgs_db_prep -f /tmp/pgs_config -e
```

**5 Stop the postmaster.**

```
zone$ ./bin/pg_ctl -D /postgres/data stop
```

**6 Add the following line to the /postgres/data/postgresql.conf file.**

```
listen_addresses = 'localhost,ha-host-1'
```

**7 Add the following line to the /postgres/data/pg\_hba.conf file.**

```
host all all 0.0.0.0/0 password
```

**8 Encrypt the password of the monitoring user on the zone, to run the PostgreSQL resource.**

```
zone# ksh /opt/SUNWscPostgreSQL/util/pgs_register -f /global/mnt3/pgs_config -e
```

**9 Leave the zone.**

```
zone$ exit
```

**10 Run the pgs\_register script to register the resource.**

```
phys-schost-1# ksh /opt/SUNWscPostgreSQL/util/pgs_register -f /global/mnt3/pgs_config
```

**11 Enable the resource.**

```
phys-schost-1# clresource enable RS-PGS
```

## Installing the PostgreSQL Binaries in the Default Directory in a Non-Global Zone With HA for Solaris Zones (Alternative Installation)

The example instructions in [“Installing and Configuring PostgreSQL on Shared Storage in a Non-Global Zone with HA for Solaris Zones” on page 70](#) install the PostgreSQL software on shared cluster storage. You can also install this software into the default directory `/usr/local/postgresql` by following the instructions in this section.

To install the PostgreSQL software in the default directory, perform the steps provided in the following example procedures:

- [“Example: Preparing the Cluster for PostgreSQL” on page 70](#)
- [“Example: Configuring Cluster Resources for PostgreSQL” on page 71](#)
- [“Example: Configuring HA for Solaris Zones” on page 71](#)
- [“Example: Modifying the PostgreSQL Configuration File” on page 73](#)
- [“Example: Building and Installing the PostgreSQL Software in the Default Directory in a Non-Global Zone With HA for Solaris Zones” on page 76](#)
- [“Example: Enabling the PostgreSQL Software to Run in the Cluster” on page 75](#)

### ▼ Example: Building and Installing the PostgreSQL Software in the Default Directory in a Non-Global Zone With HA for Solaris Zones

This example illustrates how to install the PostgreSQL software in the default directory `/usr/local/postgresql`. You can also build and install the PostgreSQL binaries on shared storage. See [“Installing and Configuring PostgreSQL on Shared Storage in a Non-Global Zone with HA for Solaris Zones” on page 70](#).

**Before You Begin** You can only install the PostgreSQL software in the default directory if one of the following conditions is true:

- `/usr` is not inherited
- `/usr/local/postgresql` is linked to somewhere in the global zone

If `/usr/local/postgresql` is linked to the global zone, create this directory in the non-global zone as well.

**1 Log in as the PostgreSQL user.**

```
zone# su - postgres
```

**2 Create the directory in the non-global zone.**

```
zone$ mkdir /pgsql-linksource
```

**3 Expand the software tar file.**

```
zone$ gzcat /tmp/postgresql-8.1.0.tar.gz | tar xvf -
```

**4 Add the C compiler and ar to your PATH.**

```
zone$ PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
zone$ export PATH
```

**5 Add the C compiler and readLine libraries to your LD\_LIBRARY\_PATH.**

```
zone$ LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
zone$ export LD_LIBRARY_PATH
```

**6 Install the source and configure the build.**

```
zone$ gzcat /tmp/postgresql-8.1.0.tar.gz | tar xvf -
zone$ cd /global/mnt3/postgres/build/postgresql-8.1.0
zone$ ./configure
```

**7 Build the PostgreSQL binaries.**

```
zone$ gmake
```

If you use gcc to build the postgres binaries, build them in a failover file system.

**8 Run the PostgreSQL regression tests.**

```
zone$ gmake check
```

**9 Switch to the root user.**

```
zone$ su
```

**10 Add the C compiler and ar to your PATH.**

This example assumes the following:

- The compiler is gcc, located in /usr/sfw/bin.
- ar is located in /usr/ccs/bin.

```
zone# PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
zone# export PATH
```

**11 Add the C compiler and readLine libraries to your LD\_LIBRARY\_PATH.**

```
zone# LD_LIBRARY_PATH=/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
zone# export LD_LIBRARY_PATH
```

**12 Install the binaries.**

```
zone# gmake install
```

**13 Exit from root access.**

```
zone$ exit
```

**14 Clean the distribution.**

```
zone# gmake clean
```

**Next Steps** Perform the steps in [“Example: Enabling the PostgreSQL Software to Run in the Cluster”](#) on [page 75](#) to complete installation and configuration of PostgreSQL.

# Deployment Example: Installing PostgreSQL in the Global Zone Using WAL File Shipping

---

This appendix presents an example of how to install and configure the PostgreSQL application and data service in the global zone using WAL file shipping as a replacement for shared storage. It is a two-node cluster configuration.

If you need to install the application in any other configuration, refer to the general-purpose procedures given in other sections of this manual. For information about PostgreSQL WAL file shipping installation in a non-global zone, see the notes in this document.

## Target Cluster Configuration

This example uses a two-node cluster with the following node names:

- `phys-schost-1` (a physical node)
- `phys-schost-2` (a physical node)

This configuration also uses the logical host name `ha-host-1`.

## Software Configuration

This deployment example uses the following software products and versions:

- Oracle Solaris 11 software for SPARC or x86 platforms
- Oracle Solaris Cluster 4.1 core software
- Oracle Solaris Cluster Data Service for PostgreSQL
- PostgreSQL version 8.3.1 source files
- `readLine` utility
- `gmake` utility
- Your preferred text editor
- Your preferred C compiler

This example assumes that you have already installed and established your cluster. It illustrates installation and configuration of the data service application only.

## Assumptions

The instructions in this example make the following assumptions:

- **Shell environment.** All commands and the environment setup given in this example are for the Korn shell environment. If you use a different shell, replace any Korn shell-specific information or instructions with the appropriate information for your preferred shell environment.
- **User login.** Unless otherwise specified, perform all procedures as superuser or assume a role that provides `solaris.cluster.admin`, `solaris.cluster.modify`, and `solaris.cluster.read` RBAC authorizations.

## Installing and Configuring PostgreSQL on Shared Storage in the Global Zone

The tasks you must perform to install and configure PostgreSQL in the global zone are as follows:

- [“Example: Preparing the Cluster for PostgreSQL” on page 80](#)
- [“Example: Configuring Cluster Resources for PostgreSQL” on page 81](#)
- [“Example: Modifying the PostgreSQL Configuration File” on page 82](#)
- [“Example: Building and Installing the PostgreSQL Software on Shared Storage” on page 84](#)
- [“Example: Enabling the PostgreSQL Software to Run in the Cluster” on page 85](#)

### ▼ Example: Preparing the Cluster for PostgreSQL

- 1 **Install and configure cluster as instructed in *Oracle Solaris Cluster Software Installation Guide*.**

Install the following cluster software components on both nodes:

- Oracle Solaris Cluster core software
- Oracle Solaris Cluster data service for PostgreSQL

- 2 **Install the following utility software on both nodes:**

- `readline` utility
- `rsync` utility
- `gmake` utility
- Your C compiler



**3 Beginning from the node that owns the file system, add the postgres users.**

```
phys-schost-1# groupadd -g 1000 postgres
```

```
phys-schost-2# groupadd -g 1000 postgres
```

```
phys-schost-1# useradd -g 1000 -d /global/mnt3/postgres -m -s /bin/ksh pgs
```

```
phys-schost-2# useradd -g 1000 -d /global/mnt3/postgres -m -s /bin/ksh pgs
```

---

**Note** – For a zone cluster, perform the steps in each zone-cluster node.

---

**4 Ensure that the PostgreSQL users can login to each other's profile using ssh without a password prompt.**

## ▼ Example: Configuring Cluster Resources for PostgreSQL

**1 Register the necessary data types on both nodes.**

```
phys-schost-1# clresourcetype register SUNW.gds
```

**2 Create the PostgreSQL resource group.**

```
phys-schost-1# clresourcegroup create -n phys-schost-1 POSTGRES-PRIM-RG
```

```
phys-schost-1# clresourcegroup create -n phys-schost-2 POSTGRES-STA-RG
```

```
phys-schost-1# clresourcegroup create - ROLECHG-RG
```

```
phys-schost-1# clresourcegroup create set -p Auto_start_on_new_cluster=false ROLECHG-RG
```

**3 Create the logical host.**

```
phys-schost-1# clreslogicalhostname create -g ROLECHG-RG ha-host-1
```

**4 Enable the resource groups.**

```
phys-schost-1# clresourcegroup online -eM ROLECHG-RG
```

```
phys-schost-1# clresourcegroup online -M POSTGRES-PRIM-RG
```

```
phys-schost-1# clresourcegroup online -M POSTGRES-STA-RG
```

## ▼ Example: Modifying the PostgreSQL Configuration File

- 1 **Modify the PGRROOT and LD\_LIBRARY\_PATH environment variables according to the needs of your build.**

The databases are stored under /postgres/data. The log file is stored under /postgres/logs/sclog.

```
phys-schost-1# PGRROOT=/postgres/postgresql-8.3.1
```

```
phys-schost-1# LD_LIBRARY_PATH=/postgres/postgresql-8.3.1:/usr/sfw/lib: \
/usr/local/lib:/usr/lib:/opt/csw/lib
```

```
phys-schost-1# export PGRROOT
```

```
phys-schost-1# export LD_LIBRARY_PATH
```

If you are installing the software in the default directory, set PGRROOT to /usr/local/pgsql and LD\_LIBRARY\_PATH to /usr/local/pgsql/lib:/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib.

- 2 **Copy the PostgreSQL configuration files from the agent directory to its deployment location.**

```
phys-schost-1# cp /opt/SUNWscPostgreSQL/util/pgs_config /postgres/pgs_config_pri
```

```
phys-schost-1# cp /opt/SUNWscPostgreSQL/rolechg/util/rolchg_config /postgres \
/rolechg_config
```

```
phys-schost-2# cp /opt/SUNWscPostgreSQL/util/pgs_config /postgres/pgs_config_sta
```

- 3 **Add the cluster information to the configuration files.**

The following list shows the relevant file entries for pgs\_config\_pri and the values to assign to each entry.

```
RS=PRIM-RS
RG=POSTGRES-PRIM-RG
PORT=5432
LH=
HAS_RS=
PFILE=PRIM-RS-pfile
```

```
USER=pgs
PGRROOT=/usr/local/pgsql
PGRROOT=/postgres/postgresql-8.3.1
PGPORT=5432
PGHOST=pgsql-port
PGLOGFILE=/postgres/logs/sclog
# LD_LIBRARY_PATH=/usr/local/pgsql/lib:/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
LD_LIBRARY_PATH=/postgres/postgresql-8.3.1/lib:/usr/sfw/lib/opt/csw/lib
SCDB=sctest
SCUSER=scusser
```

```

SCTABLE=sctable
SCPASS=
STDBY_RS=STA-RS
STDBY_RG=POSTGRES-STA-RG
STDBY_USER=pgs
STDBY_HOST=phys-schost-2
STDBY_PARFILE=/postgres/STA-RS-pfile
STDBY_PING=
ROLECHG_RS=
SSH_PASSDIR=

```

The following list shows the relevant file entries for `pgs_config_sta` and the values to assign to each entry.

```

RS=STA-RS
RG=POSTGRES-STA-RG
PORT=5432
LH=
HAS_RS=
PFILE=postgres/STA-RS-pfile

```

```

USER=pgs
PGROOT=/usr/local/pgsql
PGROOT=/postgres/postgresql-8.3.1
PGPORT=5432
PGLOGFILE=/postgres/logs/sclog
# LD_LIBRARY_PATH=/usr/local/pgsql/lib:/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
LD_LIBRARY_PATH=/postgres/postgresql-8.3.1/lib:/usr/sfw/lib/opt/csw/lib
SCDB=sctest
SCUSER=scusser
SCTABLE=sctable
SCPASS=
STDBY_RS=
STDBY_RG=
STDBY_USER=
STDBY_HOST=
STDBY_PARFILE=
STDBY_PING=
ROLECHG_RS=ROLECHG-RS
SSH_PASSDIR=

```

The following listing shows the relevant file entries for `rolechg_config` and the values you need to assign to each entry.

```

RS=ROLECHG-RS
RG=ROLECHG-RG
PORT=5432
LH=ha-host1
FILE=postgres/STA-RS-pfile
HAS_RS=
STDBY_RS=STA-RS
PRI_RS=PRIM-RS
STDBY_HOST=phys-schost-2
STDBY_PFILE=/postgres/STA-RS-pfile
TRIGGER=/postgres/data/failover
WAIT=20

```

- 4 Save and close the files.

## ▼ Example: Building and Installing the PostgreSQL Software on Shared Storage

These steps illustrate how to install the PostgreSQL software. You can build and install the PostgreSQL binaries in the default directory `/usr/local/postgresql`. Perform the following steps on both hosts.

- 1 Log in as the PostgreSQL user to the target environment.

```
phys-schost-1# su - postgres
```

- 2 Set up the build environment by performing the following steps.

- a. Create a build directory.

```
phys-schost-1$ mkdir build
```

```
phys-schost-1$ cd build
```

- b. Add the C compiler to your PATH and set the LD\_LIBRARY\_PATH.

```
phys-schost-1$ PATH=$PATH:/usr/local/bin:/usr/sfw/bin:/usr/ccs/bin
```

```
phys-schost-1$ LD_LIBRARY_PATH=/postgres/postgresql-8.3.1: \
/usr/sfw/lib:/usr/local/lib:/usr/lib:/opt/csw/lib
```

```
phys-schost-1$ export PATH LD_LIBRARY_PATH
```

- 3 Install the source and configure the build.

```
phys-schost-1$ gzcat /tmp/postgresql-8.3.1.tag.gz | tar xvf
```

```
phys-schost-1$ cd /postgres/build/postgresql-8.3.1
```

```
phys-schost-1$ ./configure --prefix=/postgres/postgresql-8.3.1
```

- 4 Build the PostgreSQL binaries.

```
phys-schost-1$ gmake
```

- 5 Run the PostgreSQL regression tests.

```
phys-schost-1$ gmake check
```

- 6 Install the PostgreSQL binaries.

```
phys-schost-1$ gmake install
```

**7 Install the utilities, including `pg_standby`.**

```
phys-schost-1$ cd contrib
```

```
phys-schost-1$ gmake install
```

```
phys-schost-1$ cd ..
```

**8 Clean the distribution.**

```
phys-schost-1$ gmake clean
```

## ▼ Example: Enabling the PostgreSQL Software to Run in the Cluster

**1 Create the directories for the databases, WAL archives, configurations, utilities, and the log file.**


---

**Note** – Perform the following steps in your target environment either in the global zone or in the zone-cluster node unless it is specified otherwise.

---

```
phys-schost-1$ mkdir /postgres/data
```

```
phys-schost-1$ mkdir /postgres/logs
```

```
phys-schost-1$ mkdir /postgres/83_walarchives
```

```
phys-schost-2$ mkdir /postgres/data
```

```
phys-schost-2$ mkdir /postgres/utilities
```

```
phys-schost-2$ mkdir /postgres/log
```

```
phys-schost-2$ mkdir /postgres/83_walarchives
```

**2 Change to the PostgreSQL root directory and initialize the data cluster.**

```
phys-schost-1$ cd /postgres/postgresql-8.3.1
```

```
phys-schost-1$ ./bin/initdb -D /postgres/data
```

```
phys-schost-2$ cd /postgres/postgresql-8.3.1
```

```
phys-schost-2$ ./bin/initdb -D /postgres/data
```

**3 Start the database.**

```
phys-schost-1$ ./bin/postmaster -D /postgres/postgresql-8.3.1
```

#### 4 Prepare the Oracle Solaris Cluster-specific test database.

---

**Note** – If you are in a zone-cluster node, ensure that you have access to a copy of your configuration file.

---

```
phys-schost-1$ ksh /opt/SUNWscPostgreSQL/util/pgs_db_prep -f  
\postgres/pgs_config_pri -e
```

#### 5 Stop the postmaster.

```
phys-schost-1$ ./bin/pg_ctl -D /postgres/data stop
```

#### 6 Copy the PGDATA directory to the standby.

```
phys-schost-1$ cd /postgres
```

```
phys-schost-1$ /usr/local/bin/rysync -arv ./data phys-schost-2:/postgres
```

#### 7 Protect the PostgreSQL configuration files.

---

**Note** – The PostgreSQL configuration files are overwritten during resilvering. You need to move the configuration files to prevent them from being overwritten.

---

```

phys-schost-1$ cd /postgres

phys-schost-1$ mkdir config

phys-schost-1$ cd data

phys-schost-1$ mv postgresql.conf ../config

phys-schost-1$ ln -s ../config/postgresql.conf ./postgresql.conf

phys-schost-1$ touch ../config/recovery.conf

phys-schost-1$ ln -s ../config/recovery.conf ./recovery.done

phys-schost-2$ cd /postgres

phys-schost-2$ mkdir config

phys-schost-2$ cd data

phys-schost-2$ mv postgresql.conf ../config

phys-schost-2$ ln -s ../config/postgresql.conf ./postgresql.conf

phys-schost-2$ touch ../config/recovery.conf

phys-schost-2$ ln -s ../config/recovery.conf ./recovery.conf

```

**8 Provide the contents for the PostgreSQL recovery file.**

```

phys-schost-1$ echo restore_command = 'cp /pgs/83_walarchives/%f %p' \
> /postgresql/data/recovery.done

phys-schost-2$ echo restore_command = '/postgres/postgres-8.3.1/bin \
/pg_standby -k 10 -t /postgres/data/failover /postgres/83_walarchives %f %p' \
/postgresql/data/recovery.conf

```

**9 Configure the archive command in the postgresql.conf file on phys-schost-1 by providing the following content.**

```

archive_command = '/usr/local/bin/rsync -arv %p \
phys-schost-2:/postgres/83_walarchives/%f </dev/null'

```

**10 Configure the archive command in the postgresql.conf file on phys-schost-2 by providing the following content.**

```

archive_command = '/usr/local/bin/rsync -arv %p \
phys-schost-1:/postgres/83_walarchives/%f </dev/null'

```

**11 Exit from the postgres user ID.**

```

phys-schost-1# exit

```

**12 Encrypt the password of the monitoring user on all nodes, to run the PostgreSQL resources.**

```
phys-schost-1# ksh /opt/SUNWscPostgreSQL/util/pgs_register -f /postgres/pgs_config_pri -e
```

```
phys-schost-2# ksh /opt/SUNWscPostgreSQL/util/pgs_register -f /postgres/pgs_config_sta -e
```

**13 Run the pgs\_register script in the global zone to register the resources.**

```
phys-schost-1# ksh /opt/SUNWscPostgreSQL/util/pgs_register -f /postgres/pgs_config_pri
```

```
phys-schost-2# ksh /opt/SUNWscPostgreSQL/util/pgs_register -f /postgres/pgs_config_sta
```

```
phys-schost-1# ksh /opt/SUNWscPostgreSQL/rolechg/util/rolechg-register -f /postgres/rolechg_config
```

**14 Add the following line to the postgresql.conf file in the PGDATA directory.**

```
listen_addresses = 'localhost, ha-host1'
```

**15 Add the following line to the pg\_hba.conf file in the PGDATA directory.**

```
host all all 0.0.0.0/0 password
```

**16 Enable the resources in the global zone.**

```
phys-schost-1# clresource enable STA-RS
```

```
phys-schost-1# clresource enable PRIM-RS
```

```
phys-schost-1# clresource enable ROLECHG-RS
```

**17 Copy the resilver scripts by performing the following steps in your target environment.**

```
phys-schost-2# cp /opt//SUNWscPostgreSQL/rolechg/util/resilver-step1 /postgres/utilities
```

```
phys-schost-2# cp /opt//SUNWscPostgreSQL/rolechg/util/resilver-step2 /postgres/utilities
```

```
phys-schost-2# chown -R postgres:postgres /postgres/utilities
```

**18 Modify the following variables in your copy of the resilver-step1 script.**

```
##### Customize the following variables#####
```

```
SOURCE_DATA=/postgres/data
TARGET_DATA=/postgres/data
TARGET=phys-schost-1
PGS_BASE=/postgres/postgresql-8.3.1
PRI_GRP=POSTGRES-PRI-RG
STDBY_GRP=POSTGRES-STA-RG
STDBY_RS=STA-RS
PGPORT=5432
ROLECHG_GRP=ROLECHG-RG
RESYNC=/usr/local/bin/rsync -rav
SSH_PASSPHRASE=false
```

```
##### End of customizations #####
```



**19 Modify the following variables in your copy of the `resilver-step2` script.**

```
##### Customize the following variables#####
```

```
SOURCE=phys-schost-2  
SOURCE_DATA=/postgres/data  
TARGET_DATA=/postgres/data  
TARGET=phys-schost-1  
PGS_BASE=/postgres/postgresql-8.3.1  
PRI_GRP=POSTGRES-PRI-RG  
STDBY_GRP=POSTGRES-STA-RG  
STDBY_RS=STA-RS  
PGPORT=5432  
ROLECHG_GRP=ROLECHG-RG  
PRI_NODE=phys-schost-1  
RESYNC=/usr/local/bin/rsync -rav  
SSH_PASSPHRASE=false
```

```
##### End of customizations #####
```



# Index

---

## C

### commands

- clreslogicalhostname, 17–18, 19–21
- clresource, 17–18, 19–21, 45
- clresourcegroup, 17–18, 19–21, 21
- clresourcetype, 17–18, 19–21
- configure, 19, 22
- createdb, 23
- gmake, 19, 22
- groupadd, 19, 22
- initdb, 23
- pg\_ctl, 23, 42, 43, 44
- useradd, 19, 22

### configuration file

- pg\_hba.conf, 45
- postgresql.conf, 44

### configuration files, 24

- pg\_hba.conf, 43
- pgs\_configPostgreSQL resource, 25
- postgresql.conf, 43

### configuration Files, WAL file shipping without shared storage, 36–42

### configuration requirements, 15–16

### configuration restrictions, 13–15

## D

### debugging, HA for PostgreSQL, 52–53

### dependency, component, 15

### deployment examples

- PostgreSQL in a non-global zone, 69

### deployment examples (*Continued*)

- PostgreSQL in a zone cluster, 61–68
- PostgreSQL in the global zone, 61–68
- PostgreSQL in the global zone using WAL file shipping, 79
- PostgreSQL using HA for Solaris Zones, 69

## E

### enabling resource group, 17, 21

### examples

- PostgreSQL in a non-global zone, 69
- PostgreSQL in a zone cluster, 61–68
- PostgreSQL in the global zone, 61–68
- PostgreSQL in the global zone using WAL file shipping, 79
- PostgreSQL using HA for Solaris Zones, 69

### extension properties, effect on fault monitor, 49

## F

### fault monitor

- operation, 50–51
- tuning, 49–51

## H

### HA for PostgreSQL

- debugging, 52–53
- fault monitors, 49–51

**HA for PostgreSQL (Continued)**

- installing, 24
  - registering resources, 45
  - software package, installing, 24
  - verifying installation, 47–49
- help, 10

**I**

- installing
  - HA for PostgreSQL, 24
  - PostgreSQL, 16–22

**O**

- Oracle Solaris Cluster software, publisher, 24
- Oracle Solaris software, publisher, 24
- overview
  - installation, 12
  - product, 11

**P**

- package, 24
- parameter file, 15–16
- parameters, Rolechanger resource, 32–36
- PostgreSQL, WAL shipping, 13
- PostgreSQL application, fault monitor, 49–51
- publisher
  - Oracle Solaris Cluster software, 24
  - Oracle Solaris software, 24

**R**

- register, `sczbt_register`, 45
- resource group
  - enabling, 17, 21
- resource types, fault monitor, 49–51
- resources, PostgreSQL application debugging, 52–53

**S**

- software package, 24
- system properties, effect on fault monitors, 49

**T**

- technical support, 10
- tuning, fault monitors, 49–51

**V**

- verifying installation
  - HA for PostgreSQL, 47–49
  - PostgreSQL, 22–23

**W**

- WAL file shipping deployment examples, PostgreSQL
  - in the global zone, 79
- WAL file shipping with out shared storage, configuration Files, 36–42