SIOS Protection Suite for Linux
PostgreSQL Recovery Kit
v8.4.0

Administration Guide

Mar 2015

To maintain the quality of our publications, we welcome your comments on the accuracy, clarity, organization, and value of this document.

Address correspondence to:
ip@us.sios.com

Table of Contents

Chapter 1: Introduction

PostgreSQL Recovery Kit Documentation

Overview

The SIOS Protection Suite for Linux PostgreSQL Recovery Kit is an SQL compliant, object-relational database management system (ORDBMS) based on POSTGRES. Since its inception, PostgreSQL has become one of the most advanced open source relational database management systems.

The SPS for Linux PostgreSQL Recovery Kit provides a mechanism for protecting PostgreSQL instances within LifeKeeper. The PostgreSQL software, LifeKeeper Core and PostgreSQL Recovery Kit are installed on two or more servers in a cluster. Once the PostgreSQL database instance is under LifeKeeper protection, clients connect to the database using a LifeKeeper protected IP address. The LifeKeeper protected IP address must be created separately and a dependency made manually between the parent PostgreSQL resource instance and the child IP address resource. In the event that the PostgreSQL server fails, LifeKeeper will first attempt to recover it on the local server. If the local recovery fails, then LifeKeeper will fail over to a backup server.

PostgreSQL Resource Hierarchy

SIOS Protection Suite Documentation

The following SIOS Protection Suite product documentation is available from the SIOS Technology Corp. website:

- SPS for Linux Release Notes
- SPS for Linux Technical Documentation
- Optional Recovery Kit Documentation

PostgreSQL Documentation

You can find the PostgreSQL documentation, including the *Administration Guide*, *User Guide* and *Reference Guide* at the following location on the web:

http://www.postgresql.org/docs

PostgreSQL Resource Hierarchy

The following example shows a typical PostgreSQL resource hierarchy:

The dependencies in the above example correspond to the following protected resources:

Resource	PostgreSQL Software Component
LKIP.EXAMPLE.COM	Protects the switchable IP address used for client connections
var/lib/pgsql/data	Protects the database data directory (PGDATA)
var/lib/pgsql/exec	Protects the PostgreSQL server and client executables (when executables are installed on a shared file system)
var/lib/pgsql/log	Protects the database log file directory (when the log path is located on a shared file system)
var/lib/pgsql/pg_xlog	Protects the database transaction log directory (*PGDATA/pg_xlog*) The transaction log directory is also referred to as Write-Ahead-Log directory.
var/lib/pgsql/socket_path	Protects the database socket directory (when the socket path is located on a shared file system).

In the event of failover, LifeKeeper will bring the file system, IP address and database resources (including all the resource dependencies) in service on a backup server. Clients will be disconnected and will need to re-connect to the server. Any SQL statement that has not been committed will need to be re-entered.

Requirements

PostgreSQL Recovery Kit Requirements

Your LifeKeeper configuration must meet the following requirements prior to the installation of LifeKeeper for Linux PostgreSQL Recovery Kit. Please refer to the SPS for Linux Installation Guide for specific instructions regarding the installation and configuration of your LifeKeeper hardware and software.

Hardware Requirements

Software Requirements

Hardware Requirements

Servers - Servers should be configured in accordance with the requirements described in the SPS for Linux Technical Documentation and the SPS for Linux Release Notes.

IP Network Interface Cards - Each server requires at least one Ethernet TCP/IP-supported network interface card. Remember, however, that best practice is for a LifeKeeper cluster to have at least two communication paths. Two separate LAN-based communication paths using dual independent sub-nets are recommended for heartbeats, and at least one of these should be configured as a private network. Using a combination of TCP and TTY heartbeats is also supported.

Software Requirements

- **TCP/IP Software** – Each server in your LifeKeeper configuration requires TCP/IP Software.
- **PostgreSQL Software** – The same version of the PostgreSQL software must be installed on all servers in the cluster. The PostgreSQL software can be downloaded from one of the mirrors available at http://www.postgresql.org/download.
- **LifeKeeper software** – It is imperative that you install the same version of the LifeKeeper software and apply the same versions of the LifeKeeper software patches to each server in your cluster.
- **LifeKeeper for Linux PostgreSQL Recovery Kit** – The PostgreSQL Recovery Kit is provided on the SPS for Linux Installation Image File (`sps.img`) via ftp download. It is packaged, installed and removed via Red Hat Package Manager, rpm:

  steeleye-lkPGSQL

Chapter 2: Configuration Considerations

This section contains information that you should consider before you start to configure and administer the PostgreSQL Recovery Kit.

Protecting PostgreSQL Best Practices

In an Active/Standby configuration, the backup server is not actively running the PostgreSQL, but stands by in case the primary server experiences a failure. In an Active/Active configuration, each server is actively running a PostgreSQL instance, while acting as a backup for the other server in case of failure. The following list provides requirements that should be adhered to when protecting a PostgreSQL resource instance in an active/standby or active/active configuration.

1. The PostgreSQL DataDir and Write-Ahead-LogPath *(PGDATA/pg_xlog*) must be installed on one or more shared file systems. The paths *DataDir* and *WAL-Path* must be shared between all servers that will protect the resource instance.

- The PostgreSQL Operating System User must own the data directory and directory containing the Write-Ahead-LogPath.
- The PostgreSQL database must have been created using the utility `initdb`. The `initdb` utility must be run as the PostgreSQL owner using the *–D <datadir>* option.
- The automatic startup of the default PostgreSQL instance must either be disabled or the default PostgreSQL instance must be restricted to running on a port other than those intended for use with LifeKeeper.
- The automatic startup of the PostgreSQL instance to be protected by LifeKeeper must be disabled. LifeKeeper will control the starting and stopping of the protected instance.
- The PostgreSQL instance must be started manually prior to hierarchy creation. It is required that the instance be started with the backend option –o "–p <port>" specified to the `pg_ctl` utility.

2. The `StartupLogPath`, `SocketPath` and the `ExecutablePath` can be installed to optional shared file systems on the primary server or each local node file system.

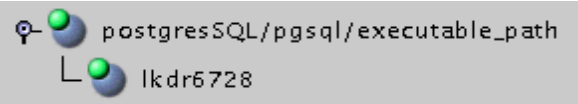
- The PostgreSQL Operating System User must own the directory containing the socket path.
- The PostgreSQL Operating System User must have write permissions on the directory containing the StartupLogPath.

3. It is recommended that each instance use a unique port and socket path when running multiple instances in either an Active/Standby or Active/Active scenario.

Using Mirrored File Systems with DataKeeper

The PostgreSQL Recovery Kit supports the use of SIOS DataKeeper as a shared file system. The mirrored file systems can be used for the PostgreSQL installation path, log path, the data directory and the executable path.

For example, a dependent file system for a PostgreSQL resource would look similar to the following, which shows a file system for the data directory and its dependency, the DataKeeper resource mirror.



Chapter 3: Installation

Installing/Configuring PostgreSQL with LifeKeeper

The following sequence is recommended for installing and configuring the PostgreSQL database and LifeKeeper software. Each of these steps links to detailed tasks.

1. Install the PostgreSQL software.
2. Create the PostgreSQL database.
3. Install the LifeKeeper Core and PostgreSQL Recovery Kit.
4. Configure LifeKeeper Tunable Settings for PostgreSQL Resources.

After you have performed these tasks, you will be ready to create the LifeKeeper resource hierarchy to protect your PostgreSQL database.

Resource Configuration Tasks

Once you have completed the setup tasks described in the previous section, you are ready to create and extend your PostgreSQL resource hierarchies.

The following tasks are available for configuring the LifeKeeper for Linux PostgreSQL Recovery Kit:

- Create Resource Hierarchy - Creates a PostgreSQL resource hierarchy.
- Delete Resource Hierarchy - Deletes a PostgreSQL resource hierarchy.
- Extend Resource Hierarchy - Extends a PostgreSQL resource hierarchy from the primary server to the backup server.
- Unextend Resource Hierarchy - Unextends (removes) a PostgreSQL resource hierarchy from a single server in the LifeKeeper cluster.
- Viewing PostgreSQL Configuration Settings - Allows viewing of the Resource Properties dialog.

Refer to the GUI Administrative Tasks section of the SPS for Linux Technical Documentation for instructions on configuring LifeKeeper Core resource hierarchies, for instance, file system and IP resources.

The following tasks are described in the Administration section within the SPS for Linux Technical Documentation because they are common tasks with steps that are identical across all Recovery Kits.

- Create a Resource Dependency. Creates a parent/child dependency between an existing resource hierarchy and another resource instance and propagates the dependency changes to all applicable

servers in the cluster.

- Delete a Resource Dependency. Deletes a resource dependency and propagates the dependency changes to all applicable servers in the cluster.
- In Service. Brings a resource hierarchy into service on a specific server.
- Out of Service. Takes a resource hierarchy out of service on a specific server.
- View/Edit Properties. View or edit the properties of a resource hierarchy on a specific server.

Note: The configuration tasks throughout this section are performed using the **Edit** menu. You may also perform most of the tasks:

- from the toolbar.
- by right-clicking on a global resource in the left pane of the status display.
- by right-clicking on a resource in the right pane of the status display.

Using the right-click method allows you to avoid entering information that is required using the **Edit** menu.

Upgrading

Upgrading From Previous Version of the PostgreSQL Recovery Kit

Install the PostgreSQL Software

Install the PostgreSQL software on all servers in the cluster using identical parameters/settings. Refer to the *PostgreSQL Administration Guide* for details. The following are additional recommendations and reminders to ensure that LifeKeeper will work with PostgreSQL:

- The PostgreSQL client software packages must be installed. These packages must include the PostgreSQL `psql` client utility.
- The PostgreSQL server software packages must be installed. These packages must include the PostgreSQL `pg_ctl` and `initdb` utilities.
- The PostgreSQL client and server packages must be the same version on all servers.
- A PostgreSQL Operating System User must exist on all servers as follows:
    - This PostgreSQL Operating System User should be designated as the owner of the PostgreSQL software installation and subdirectories.
    - This PostgreSQL Operating System User must have authority to use the `pg_ctl` utility. The PostgreSQL Operating System User must be able to start and stop the postmaster instance using the `pg_util` commands.
    - The PostgreSQL Operating System User name should contain alphanumeric characters only.
    - The user id and group id of this PostgreSQL Operating System User must be identical on all servers.

- A PostgreSQL Database Administrator User must exist within the PostgreSQL database for LifeKeeper client connections through the `psql` utility.
    - This PostgreSQL Database Administrator User must have the ability to connect to the database (*template1*), as well as obtain the listing of defined databases for the instance.
    - This PostgreSQL Database Administrator User must have the ability to view system tables and make generalized queries.
    - The PostgreSQL Database Administrator User is different from the PostgreSQL Operating System User, although they can have the same name.
    - Example: PostgreSQL Operating System User=postgres, and PostgreSQL Database Administrator User=lkpostgres; or PostgreSQL Operating System User=postgres, and PostgreSQL Database Administrator User=postgres.

Create the PostgreSQL Database

Follow the instructions in your *PostgreSQL Administration Guide* to create your database. In addition, please note the following recommendations:

- The PostgreSQL data directory should be initialized using the `initdb` utility, specifying the *–D <data dir>* option. The `initdb` command must be run as the PostgreSQL Operating System User.
- The PostgreSQL instance data directory must reside on a shared file system.
- The PostgreSQL transaction log directory must reside on a shared file system.
- The PostgreSQL database name should contain alphanumeric characters only.
- After creating your database, you should disable automatic startup of the PostgreSQL database instance. Once under LifeKeeper protection, LifeKeeper will handle the start and stop of the database.
- The PostgreSQL instance must be started manually prior to hierarchy creation. It is required that the instance be started with the backend option *-o "–p <port>"* specified to the `pg_ctl` utility.

No Password Protection (Instance is not Password Protected)

- If the PostgreSQL database instance will not be password protected or will not require a password for local client connections from the PostgreSQL Database Administrator User, then an entry must exist allowing local trust connections. The following is an example of a *pg_hba.conf* entry to enable local client connects for the PostgreSQL Database Administrator User:

```
========================
.
.
Local all postgres trust
.
.
==========================
```

Enabling Password Protected (Instance requires a Password for Connections)

- Password Protected database instances require a password entry for the PostgreSQL Database Administrator User to exist in the *.pgpass* credentials file on each server in the cluster where the resource will be protected. The *.pgpass* file must contain a valid and tested entry for each PostgreSQL Database Administrator User requiring a password.
- The *.pgpass* file must be located in the home directory of the PostgreSQL Operating System User. Please set the appropriate file permissions to restrict access to the file.
- The following is an example of a valid *.pgpass* file with the format

  <hostname>:<port>:<database>:<user>:<password>

  ==========

  **:5443:*:lifekeeper:jh43tmp2009*

  ==========

  Note: The *.pgpass* file is required for the utility `psql` for unattended (non-terminal or scripted) connections. The *.pgpass* file must exist on each server where the password protected instance will be protected.

Install the LifeKeeper Software

Once you have installed the PostgreSQL software and created your database, you are ready to install the LifeKeeper Core software and any required patches followed by the PostgreSQL Recovery Kit.

Refer to the SPS for Linux Installation Guide for details on installing the LifeKeeper packages.

LifeKeeper Tunable Settings for PostgreSQL

The PostgreSQL Recovery Kit provides tunable environment variables to help customize resource protection in certain scenarios. To change the values of these variables, edit the file */etc/default/LifeKeeper*. No processes need to be restarted for the new settings to take effect. The default values will work for most environments where the PostgreSQL Recovery Kit will be installed.

- ## LKPGSQL_CONN_RETRIES

  This tunable controls the amount of time the PostgreSQL Recovery Kit will wait for the database to start. The amount of time is calculated by the Recovery Kit using the following formula: (LKPGSQL_ CONN_RETRIES* 5) = total time in seconds to wait for a database instance to start. The setting of this variable affects both the resource in-service requests and the resource local recovery.

- ## LKPGSQL_DISCONNECT_CLIENT

  This tunable controls whether active clients will be disconnected in the event of a postmaster crash. When the value is set to 1 (true), active clients will be disconnected while resource local recovery is in

progress. When the value is set to 0 (false), active clients will not be disconnected while resource local recovery is in progress. This variable affects only the resource local recovery events and is only applicable during local recovery events where the postmaster process is not running.

- ### LKPGSQL_SDIRS

  This tunable controls the client disconnect behavior when the PostgreSQL database is shut down. This comma separated tunable must be added to the defaults file. By setting this option, the specified resource instance or instances corresponding to the protected data directory will not force clients to disconnect during shutdown.

  ```
  LKPGSQL_SDIRS=/protected/pgsql-datadir

  LKPGSQL_SDIRS=/protected/pgsql-datadir,/otherprotected/pgsql-datadir
  ```

  Where */protected/pgsql-datadir* and */otherprotected/pgsql-datadir* are the PostgreSQL data directories under LifeKeeper protection.

  Note: The options LKPGSQL_SDIRS and LKPGSQL_IDIRS are exclusive. The value placed in the LKPGSQL_SDIRS or LKPGSQL_IDIRS tunable must match exactly with the protected datadir value selected during hierarchy creation.

- ### LKPGSQL_IDIRS

  This tunable controls the client disconnect behavior when the PostgreSQL database is shut down. This comma separated tunable must be added to the defaults file. By setting this option, the specified resource instance or instances corresponding to the protected data directory will force clients to do an immediate disconnect during shutdown.

  ```
  LKPGSQL_IDIRS=/protected/pgsql-datadir

  LKPGSQL_IDIRS=/protected/pgsql-datadir,/otherprotected/pgsql-datadir
  ```

  Where */protected/pgsql-datadir* and */otherprotected/pgsql-datadir* are the PostgreSQL data directories under LifeKeeper protection.

  Note: The options LKPGSQL_SDIRS and LKPGSQL_IDIRS are exclusive. The value placed in the LKPGSQL_SDIRS or LKPGSQL_IDIRS tunable must match exactly with the protected datadir value selected during hierarchy creation.

Creating a PostgreSQL Resource Hierarchy

Perform the following steps on the primary server:

1. On the **Edit** menu, select **Server**, then **Create Resource Hierarchy**.

   The **Create Resource Wizard** dialog will appear.

2. Select **PostgreSQL Database** from the drop-down list and click **Enter**.

3. You will be prompted for the following information. When the **Back** button is active in any of the dialog

boxes, you can go back to the previous dialog box. This is helpful should you encounter any error requiring you to correct the previously entered information. You may click **Cancel** at any time to cancel the entire creation process.

Field	Tips
Switchback Type	Choose either **intelligent** or **automatic**. This determines how the PostgreSQL resource will be switched back to the primary server after it comes in-service (active) on the backup server following a failover. Intelligent switchback requires administrative intervention to switch the resource back to the primary server, while automatic switchback occurs as soon as the primary server is back on line and re-establishes LifeKeeper communication paths. **Note:** The switchback strategy must match that of the dependent resources to be used by the PostgreSQL resource.
PostgreSQL Executable Location	This field is used to specify the directory path containing the PostgreSQL executables. The valid characters allowed for the pathname are letters, digits and the following special characters: - _ . /
PostgreSQL Client Executable Location	This field is used to specify the directory path containing the PostgreSQL executable `psql`. The valid characters allowed for the pathname are letters, digits and the following special characters: - _ . /
PostgreSQL Administration Executable Location	This field is used to specify the directory path containing the PostgreSQL executable `pg_ctl`. The valid characters allowed for the pathname are letters, digits and the following special characters: - _ . /
PostgreSQL Data Directory	This field is used to specify the location of the PostgreSQL data directory (*datadir*) that will be placed under LifeKeeper protection. The specified directory must exist and reside on a shared file system. The valid characters allowed for the pathname are letters, digits and the following special characters: - _ . /
PostgreSQL Port	This field is used to specify the TCP/IP port number on which the postmaster daemon is listening for connections from client applications.
PostgreSQL Socket Path	This field is used to specify the full path to the Unix-domain socket on which the postmaster daemon is listening for connections from client applications. The valid characters allowed for the pathname are letters, digits and the following special characters: - _ . /
PostgreSQL Database Administrator User	This field is used to specify a PostgreSQL Database Administrator User name for the specified database instance with connection and administrator privileges for the instance.
PostgreSQL Logfile	This field is used to specify the log file path that will be used for the PostgreSQL log file.

Field	Tips
PostgreSQL Database Tag	This is a unique tag name for the new PostgreSQL database resource on the primary server. The default tag name consists of the word pgsql followed by the port number for the database instance. You may type in another unique tag name. The valid characters allowed for the tag are letters, digits and the following special characters: - _ . /

4. Click **Create**. The **Create Resource Wizard** will then create your PostgreSQL resource hierarchy. LifeKeeper will validate the data entered. If LifeKeeper detects a problem, an error message will appear in the information box.
5. You should see a message indicating that you have successfully created a PostgreSQL resource hierarchy, and you must extend that hierarchy to another server in your cluster to achieve failover protection. Click **Next**.
6. Click **Continue**. LifeKeeper will then launch the **Pre-extend Wizard**. Refer to **Step 2** in the topic Extending a PostgreSQL Resource Hierarchy for details on how to extend your resource hierarchy to another server.

Deleting a PostgreSQL Resource Hierarchy

To delete a PostgreSQL resource hierarchy from all servers in your LifeKeeper configuration, complete the following steps:

1. On the **Edit** menu, select **Resource**, then **Delete Resource Hierarch**y.
2. Select the name of the **Target Server** where you will be deleting your PostgreSQL resource hierarchy.

   Note: If you selected the **Delete Resource** task by right-clicking from either the left pane on a global resource or the right pane on an individual resource instance, this dialog will not appear.
3. Select the **Hierarchy to Delete**. (This dialog will not appear if you selected the **Delete Resource** task by right-clicking on a resource instance in the left or right pane.) Click **Next**.
4. An information box appears confirming your selection of the target server and the hierarchy you have selected to delete. Click **Next**.
5. Another information box appears confirming that the PostgreSQL resource was deleted successfully.
6. Click **Done** to exit.

Extending a PostgreSQL Resource Hierarchy

This operation can be started from the **Edit** menu or initiated automatically upon completing the **Create Resource Hierarchy** option, in which case you should refer to **Step 2** below.

1. On the **Edit** menu, select **Resource**, then **Extend Resource Hierarchy**. The ***Pre-Extend Wizard*** appears. If you are unfamiliar with the **Extend** operation, click **Next**. If you are familiar with the LifeKeeper Extend Resource Hierarchy defaults and want to bypass the prompts for

input/confirmation, click **Accept Defaults**.

2. The ***Pre-Extend Wizard*** will prompt you to enter the following information.

   Note: The first two fields appear only if you initiated the **Extend** from the **Edit** menu.

Field	Tips
Template Server	Select the server where your PostgreSQL resource is currently in service.
Tag to Extend	Select the PostgreSQL resource you wish to extend.
Target Server	Enter or select the server you are extending *to*.
Switchback Type	This determines how the PostgreSQL resource will be switched back to the primary server after it comes in service (active) on the backup server following a failover. You can choose either **intelligent** or **automatic**. The switchback type can be changed later, if desired, from the **General** tab of the **Resource Properties** dialog box. **Note:** Remember that the switchback strategy must match that of the dependent resources to be used by the PostgreSQL resource.
Template Priority	Select or enter a **Template Priority**. This is the priority for the PostgreSQL hierarchy on the server where it is currently in service. Any unused priority value from 1 to 999 is valid, where a lower number means a higher priority (1=highest). The extend process will reject any priority for this hierarchy that is already in use by another system. The default value is recommended. **Note:** This selection will appear only for the initial extend of the hierarchy.
Target Priority	This is the priority for the new extended PostgreSQL hierarchy relative to equivalent hierarchies on other servers. Any unused priority value from 1 to 999 is valid indicating a server's priority in the cascading failover sequence for the resource. Note that LifeKeeper assigns the number "1" to the server on which the hierarchy is created by default. The priorities need not be consecutive, but no two servers can have the same priority for a given resource.

3. After receiving the message that the pre-extend checks were successful, click **Next**.
4. Depending upon the hierarchy being extended, LifeKeeper will display a series of information boxes showing the Resource Tags to be extended, some of which cannot be edited.
5. The***Extend Wizard*** will prompt you to enter the following information.

Field	Tips
PostgreSQL Executable Location	This field is used to specify the directory path containing the PostgreSQL executables. The valid characters allowed for the pathname are letters, digits and the following special characters: - _ . /
PostgreSQL Database Tag	This is a unique tag name for the new PostgreSQL database resource on the primary server. The default tag name consists of the word pgsql followed by the port number for the database instance. You may type in another unique tag name. The valid characters allowed for the tag are letters, digits and the following special characters: - _ . /

6. After receiving the message "`Hierarchy extend operations completed`", click **Next Server** to extend the hierarchy to another server, or click **Finish** if there are no other extend operations to perform.
7. After receiving the message "`Hierarchy Verification Finished`", click **Done**.

Unextending a PostgreSQL Resource Hierarchy

To remove a resource hierarchy from a single server in the LifeKeeper cluster, do the following:

1. On the **Edit** menu, select **Resource**, then **Unextend Resource Hierarchy**.
2. Select the **Target Server** where you want to unextend the PostgreSQL resource. It cannot be the server where the resource is currently in service. (This dialog box will not appear if you selected the Unextend task by right-clicking on a resource instance in the right pane.) Click **Next**.
3. Select the PostgreSQL hierarchy to unextend and click **Next**. (This dialog will not appear if you selected the Unextend task by right-clicking on a resource instance in either pane).
4. An information box appears confirming the target server and the PostgreSQL resource hierarchy you have chosen to unextend. Click **Unextend**.
5. Another information box appears confirming that the PostgreSQL resource was unextended successfully. Click **Done** to exit the **Unextend Resource Hierarchy** menu selection.

Viewing PostgreSQL Configuration Settings

The **Resource Properties** dialog is available from the **Edit** menu or from a resource context menu. This dialog displays the properties for a particular resource on a server. When accessed from the **Edit** menu, you can select the resource and the server. When accessed from a **Resource Context** menu, you can select the server.

From the **Configuration** tab, you can view the following PostgreSQL settings:

- Executable Path
- Client Executable Name
- Admin Executable Name
- Bind Setting
- Startup Log Location
- PostgreSQL Operating System User Name
- PostgreSQL Database Administrator User
- Version Number
- Data Directory
- Socket Location

- Port Number
- OS Daemon Name

Upgrading From Previous Version of PostgreSQL Recovery Kit

During an upgrade from a previous version of the SPS for Linux PostgreSQL software, the upgrade will make modifications to the existing SPS PostgreSQL resource instance. When the SPS software is updated on the server, the following stored values will be added to the internal SPS information field automatically.

- **Client Executable Location (*psql*)** – the location of the *psql* or equivalent client utility used for connecting to the protected database instance. After an upgrade, this value can be verified from the LifeKeeper GUI properties display. The value can also be verified from the LifeKeeper command line using the `set_value` utility.

  `set_value` is the name of a LifeKeeper utility provided for the LifeKeeper PostgreSQL Recovery Kit to update the internal resource information field values. The use of this utility should be limited to issues explained in this topic or at the request and instruction of the SIOS Technology Corp. Support team.

  Note: The `set_value` utility does not perform rigorous error checking and therefore is not intended for general use.

- **Administration Executable Location (*pg_ctl*)** – the location of the *pg_ctl* or equivalent administration utility used for starting, stopping and checking the status of the protected database instance. After an upgrade, this value can be verified from the LifeKeeper GUI properties display. The value can also be verified from the LifeKeeper command line using the `set_value` utility.
- **PostgreSQL Database Administrator User**– the PostgreSQL Database Administrator User for the LifeKeeper protected instance. This user must have connection and administrator privileges for the protected database instance. The default value used following an upgrade is the PostgreSQL Operating System User that owns the PostgreSQL data directory. After an upgrade, this value can be verified from the LifeKeeper GUI properties display. The value can also be verified from the LifeKeeper command line using the `set_value` utility.
- **PostgreSQL Daemon Name (postmaster)** – the name of the running backend daemon. This value is determined during the first status check of the database instance. The default value is postmaster. After an upgrade, this value can be verified from the LifeKeeper GUI properties display. The value can also be verified from the LifeKeeper command line using the `set_value` utility.
- **Default Test Database (template1)** – the database used by LifeKeeper during the database instance monitoring to verify basic connectivity. After an upgrade, the default test database will be set to **template1**.
- **PostgreSQL Maximum Monitoring Hangs (`LKPGSQL_QCKHANG_MAX`)** – the setting that provides protection against an unlimited number of connection hangs before a restorative or reparative failover action is initiated. A portion of PosgreSQL Recovery Kit's monitoring requires a connection to the protected database. The number of connection hangs allowed is determined during resource creation by the setting `LKPGSQL_QCKHANG_MAX`. The default value previous to version 8.1.2 was 15. After upgrading to version 8.1.2 (or later), the default value is 2. Since this value is stored with the resource

at create time, any resources created prior to upgrading to version 8.1.2 will remain at the default value of 15 unless updated by the user while any resources created after upgrading to 8.1.2 (or later) will contain a default value of 2. The value can also be verified from the SPS command line using the `set_value` utility.

IMPORTANT NOTE: Following the upgrade of the SPS for Linux PostgreSQL Recovery Kit software, you should test your PostgreSQL resource hierarchy by initiating a manual switchover that will simulate a failover of the resource instance from the primary server to a backup server.

Important Upgrade Considerations	If a resource does not come into service following the upgrade, check the following conditions:
	
	Client Executable name is not found or incorrect
	
	The value can be updated using the `set_value` utility. The syntax for the Client Executable update is as follows:
	
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value <tag> 'clientexe' <full path to the psql utility>.`
	
	Example:
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value pgsql-5443 'clientexe' '/pgsql/clientutils/psql'.`
	
	Administration Executable name is not found or incorrect
	
	The value can be updated using the `set_value` utility. The syntax for the Administration Executable update is as follows:
	
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value <tag> 'osexe' <full path to the pg_ctl utility>.`
	
	Example:
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value pgsql-5443 'osexe' '/pgsql/adminutils/pg_ctl'.`

	Lowering the interval for recovering from multiple hang events (LKPGSQL_QCKHANG_MAX)
	
	Maximum Monitoring Hangs value is too large in versions prior to 8.1.2
	
	The value for **Maximum Monitoring Hangs** for existing PostgreSQL resource instances can viewed or set using the set_value utility.
	
	The syntax for setting the value for the **Maximum Monitoring Hangs** (LKPGSQL_QCKHANG_MAX) is as follows:
	
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value <tag> 'hangmax' <number>.`
	
	Example:
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value pgsql-5443 'hangmax' 3.`
	
	Note: Include the `-c` argument to update the value on all nodes in the cluster (`set_value -c <tag>...`).
	
	The syntax for **viewing** the value is as follows:
	
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value -l <tag> 'hangmax'`
	
	Example:
	`/opt/LifeKeeper/lkadm/subsys/database/pgsql/bin/set_value -l pgsql-5443 'hangmax'`

Chapter 4: Administration

Updating Database Administrator User

The Update User option allows the LifeKeeper administrator to change the current PostgreSQL Database Administrator User for the LifeKeeper PostgreSQL resource instance.

Testing Your PostgreSQL Resource Hierarchy

You can test your PostgreSQL resource hierarchy by initiating a manual switchover that will simulate a failover of the resource instance from the primary server to a backup server.

Performing a Manual Switchover from the LifeKeeper GUI

EnterpriseDB Postgres Plus Advanced Server Environments

Protecting EnterpriseDB Postgres Plus Advanced Server Resources

Performing a Manual Switchover from the LifeKeeper GUI

You can initiate a manual switchover from the LifeKeeper GUI by selecting **Edit**, **Resource**, and **In Service**. For example, an in-service request executed on a backup server causes the PostgreSQL resource hierarchy to be placed in service on the backup server and taken out-of-service on the primary server. At this point, the original backup server is now the primary server and original primary server has now become the backup server.

If you execute the Out of Service request, the resource hierarchy is taken out-of-service without bringing it in service on the other server.

Important: After bringing your resource hierarchy in service on the backup server, you should attempt to connect to the databases. With password protected instances, it is of particular importance that the *.pgpass* file is verified on the backup server. To verify the *.pgpass* file is valid, a client connection to the database should be made using both the `psql` utility and the PostgreSQL Database Administrator User. A valid *.pgpass* file exists if the connection succeeds without prompting for an interactive password.

Protecting EnterpriseDB Postgres Plus Advanced Server

No additional LifeKeeper configuration settings are needed to protect EnterpriseDB Postgres Plus Advanced Server Resources.

Issue	Solution

During the installation of EnterpriseDB Postgres Plus Advanced Server, if the option **PostgreSQL-compatible defaults and samples** is chosen in the **Configuration Mode** dialog, the '*edb*' database that is used by LifeKeeper is not created.	Manually add the '*edb*' database using the utility '`createdb`'.
	
	The command '`createdb -p <port> -h <socket path> edb`' should be executed as the PostgreSQL Operating System User. The following is an example:
	
	`su - postgres`
	
	`postgres@server1 ~>createdb -p 5435 -h /var/lib/postgres edb`

Updating Database Administrator User

This **Update User** option will update the stored value for the PostgreSQL Database Administrator User on all systems where the resource is protected. The **Update User** option can be invoked from either the **LifeKeeper resource toolbar** or the **LifeKeeper resource context menu**.

To update the PostgreSQL Database Administrator User, perform the following steps on the primary server:

Note: The **Update User** menu and toolbar options will be disabled for any out-of-service resources.

1. On the toolbar, select the **Update User** icon or select Update User from the resource context menu.

   The **Update User Wizard** dialog will appear.

2. You will be prompted for the following information. When the **Back** button is active in any of the dialog boxes, you can go back to the previous dialog box. This is helpful should you encounter any error requiring you to correct the previously entered information. You may click **Cancel** at any time to cancel the entire creation process.

Field	Tips
Enter **PostgreSQL Database Administrator User**	This dialog requests a PostgreSQL Database Administrator User name for the specified database instance with connection and administrator privileges for the instance. **Note**: A validation script will verify connectivity using the value specified. A password protected instance will require a valid entry in the *.pgpass* file for the PostgreSQL Database Administrator User.
Confirm Update Action	This dialog requests confirmation of the update user change of the previous user value to the new user value.

3. Click **Update**. The PostgreSQL Database Administrator User will be updated on all servers where the resource is currently protected.

Chapter 5: Troubleshooting

This section provides a list of messages that you may encounter while creating and extending an SPS PostgreSQL resource hierarchy or removing and restoring a resource. Where appropriate, it provides an additional explanation of the cause of an error and necessary action to resolve the error condition.

Messages from other SPS components are also possible. In these cases, please refer to the Message Catalog (located on our Technical Documentation site under "Search for an Error Code") which provides a listing of all error codes, including operational, administrative and GUI, that may be encountered while using SIOS Protection Suite for Linux and, where appropriate, provides additional explanation of the cause of the error code and necessary action to resolve the issue. This full listing may be searched for any error code received, or you may go directly to one of the individual Message Catalogs for the appropriate SPS component.

General Tips

The following error messages and conditions may be encountered while using the recovery kit.

Error	Solution
Unable to protect PostgreSQL database using the same port as another LK protected PostgreSQL database.	Verify the version of PostgreSQL includes a *postgresql.conf* file. In the *postgresql.conf* file, set the entry listen_address= to the IP address to be used with the database instance. **Note:** The format of the listen_address= in the *postgresql.conf* file is important as syntax errors can result in a failure to start the database server.
Unable to perform a manual switchover of version 8.X when clients are connected.	The default (smart) shutdown option fails to disconnect clients on a switchover. If shutdown continues to fail with connected clients, verify that the LKPGSQL_SDIRS tunable is not set. If the problem persists, set the LifeKeeper tunable LKPGSQL_IDIRS.
Unable to connect from a remote client to the database server.	To enable remote host login for PostgreSQL, refer to the *PostgreSQL Administration Guide* on configuring the *pg_hba.conf* file.

Error	Solution
psql: connectDBStart() -- connect() failed: No such file or directory. Is the postmaster running at 'localhost' and accepting connections on Unix socket '<port>'?"	Verify that the socket file exists and the instance is currently running. If the socket file resides in */tmp*, it may have been removed by a cron job that cleans up the */tmp* directory. Take the resource out of service and back in service. Then modify the cron job to leave PostgreSQL socket files.
PostgreSQL resource hierarchy fails to come in service but the database is running.	The database may have failed to respond to the LifeKeeper client request within the specified interval. Adjust the tunable LKPGSQL_CONN_RETRIES in */etc/default/LifeKeeper* to increase the number of seconds allowed for the recovery and restart of the PostgreSQL database instance.
PostgreSQL resource hierarchy fails local recovery following a postmaster crash with active client connections.	When a large number of active clients are connected to PostgreSQL, the database may be unable to properly restart until the client connections have terminated. In this scenario, it may be best to force client connections to terminate so that local recovery will be successful. The variable LKPGSQL_DISCONNECT_CLIENT can be set in */etc/default/LifeKeeper* to control the behavior of the PostgreSQL resource hierarchy in this scenario. When the value is set to 1(**true)**, client processes will be sent a SIGTERM signal to force them to disconnect from the database. This action will only be taken if the postmaster process is not running during local recovery.
Unable to connect to database with error "WARNING: password file "*/home/<osuser>/.pgpass*" has world or group read access"	The *.pgpass* file permissions should be u=rw(0600). Change the permissions and owner of the *.pgpass* file.
FATAL: syntax error in file "/<datadir>/postgresql.conf" line 50, near token ".17"	The *postgresql.conf* file listen_address= entry does not contain proper quoting. Verify entries are valid and the entry is enclosed in proper quotes.

Tunables

Tunable	Function
LKPGSQL_KILLPID_TIME	Time to wait after a process id is killed before rechecking for this process.
LKPGSQL_CONN_RETRIES	Replaces LKPGSQLMAXCOUNT – number of times to try a client connection after an action (start or stop)

Tunable	Function
LKPGSQL_ ACTION_ RETRIES	Number of times to attempt start or stop action before failing the action command.
LKPGSQL_ STATUS_TIME	Timeout for status command.
LKPGSQL_ QCKHANG_MAX	Number of quickCheck script hangs allowed before a failover/sendevent is triggered for the database instance.
LKPGSQL_ CUSTOM_ DAEMON	Allows a user to specify additional aliases for the postgres daemons (default postmaster).
LKPGSQL_IDIRS	Replaces LKPGSQL_IPORTS – Contains datadir entries for instances that will be shutdown using the immediate option only.
LKPGSQL_ SDIRS	Contains datadir entries for instances that will be shutdown using the smart option.
LKPGSQL_ DISCONNECT_ CLIENT	Controls the behavior the PostgreSQL resource hierarchy during a database failure scenario. When the value is set to 1(**true)**, client processes will be sent a SIGTERM signal to force them to disconnect from the database. This action will only be taken if the postmaster process is not running during local recovery.
LKPGSQL_ DISCONNECT_ CLIENT_BYTAG	Similar to LKPGSQL_DISCONNECT_CLIENT, this setting limits the action to the comma separated list of tags specified by this tunable.
LKPGSQL_ RESUME_PROC	Determines if process found in the stopped state (state = ~T) will be resumed when detected or ignored.
LKPGSQL_ CLIENT	Provides a hint for the name of the client utility psql.
LKPGSQL_UTIL	Provides a hint for the name of the administrative utilitypg_ctl.
LKPGSQLDEBUG	Turns on debug for PostgreSQL database kit as well as for the postgres database. Valid entry range: 0 – 5. Larger numbers produce greater debug information. This tunable will be passed on to the postmaster database using the option –d <LKPGSQLDEBUG>.