SIOS Protection Suite for Windows
PostgreSQL Server Recovery Kit
v8.6

Administration Guide

August 2017

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

Introduction

SIOS Protection Suite PostgreSQL Server

The SIOS Protection Suite PostgreSQL Server Recovery Kit software lets you tie the data integrity of PostgreSQL-based databases to the increased availability provided by SIOS Protection Suite for Windows.

The LifeKeeper GUI allows you to easily create a PostgreSQL resource hierarchy. SIOS Protection Suite can then protect all of the disk resources used by the PostgreSQL Server instance, as well as the LifeKeeper network resources used by clients to access the database.

PostgreSQL Server Installation

Proper operation of the SIOS Protection Suite PostgreSQL Server Recovery Kit depends upon correct setup of the hardware and software.

Before continuing, please preview the Hierarchy Administration section of this guide. This section provides general guidelines, configuration details and troubleshooting hints to help you administer PostgreSQL Server in a SIOS Protection Suite environment.

Installation and Configuration Details - Adding LifeKeeper to an Existing PostgreSQL Configuration

This section covers installation and configuration of the SIOS Protection Suite and PostgreSQL software when adding LifeKeeper to an existing PostgreSQL configuration. The Primary server is assumed to be the location of the active PostgreSQL database cluster that will be protected. These steps assume the database cluster to be protected is the version created by the default installation of PostgreSQL (e.g. postgresql-x64-9.6 with PostgreSQL v9.6) These steps below must be followed in order.

1. Install the SIOS Protection Suite software on the Primary and Backup Server.
2. Using the LifeKeeper GUI on the Primary Server, create comm paths between the primary and backup server.
3. On the Primary Server:
    a. Use the **Disk Management** utility to configure your disk resources and define the shared or replicated volumes that you want to use. Be sure the volume size is adequate. If you are configuring shared volumes, power down the backup server during configuration to avoid simultaneous access to your storage.
    b. It is recommended that you use **Windows Explorer** to unshare all volumes to be used by the PostgreSQL Server Instance from the network.
    c. Configure your networking to support the SIOS Protection Suite TCP/IP comm path(s) and the switchable IP address if applicable.
4. On the Backup Server:
    a. Start the backup server if it was stopped to configure shared volumes in Step 3.
    b. Use the **Disk Management** utility to configure your disk resources and define the shared or replicated volumes that you want to use. If you are using shared storage, assign the same drive letter to the shared volume as assigned on the primary server. For replicated storage be sure the volume size is adequate.
5. On the Primary Server:

a. In SIOS Protection Suite, create your shared or replicated Volume resource (where the PostgreSQL database cluster will reside) and extend it to the backup server. Later when you create your PostgreSQL Server resource hierarchy. SIOS Protection Suite will automatically bring the resource into the hierarchy as a dependency.

6. On the Backup Server:

   a. Bring the volume resource hierarchy In Service using the **LifeKeeper GUI**.

   b. Install PostgreSQL to the same program folder as it is installed to on the primary server using the following guidelines:

      i. Using the --extract-only argument to the PostgreSQL installer is not recommend as it does not configure all of the information required by the PostgreSQL Recovery Kit.

      ii. By default the installation of the PostgreSQL software creates a single database cluster (a database cluster is a collection of databases that is managed by a single instance of a running PostgreSQL database server). During installation the Data Directory prompt determines where the database cluster will be created. The default instance can be installed at any location since it will be deleted in step 6d.

   c. The default database cluster instance created during installation is not required and can be removed.

      i. Stop the database cluster instance created during installation.

      ii. Open **Explorer** and access the drive associated with the replicated volume.

      iii. Delete the PostgreSQL database cluster directory created during the installation.

      iv. Delete the PostgreSQL service that was created during installation. You can use the Windows "sc delete <servicename>" command to do this.

7. On the Primary Server:

   a. Bring the volume resource hierarchy In Service using the **LifeKeeper GUI**.

   b. Stop the PostgreSQL database cluster instance that is to be protected by the PostgreSQL Recovery Kit.

   c. **Optional** - Perform a backup of the PostgreSQL database cluster data directory prior to moving it to the protected volume created above.

   d. Move the database cluster data directory to the protected volume created above.

   e. Set the access rights on the database cluster data directory. The user account setup to control the Window's Service for this instance must have full control file permissions on the data directory.

   f. Follow the steps outlined in Configuring the Postmaster Port Argument on the primary server. The configuration should also include modification of –D argument which specifies the location of the data directory. It should be changed to the path on the protected volume.

   g. Follow the steps outlined in Configure for Unattended Connections on the primary server.

   h. When the installation and configuration is complete, start the Windows PostgreSQL service to

verify that the PostgreSQL Server can start properly on the primary server with the Postmaster port argument and the data directory now located on the protected volume.

i. Create the PostgreSQL Server hierarchy on the primary server and extend it to the backup server. See Creating the PostgreSQL Hierarchy for more information. Test the new PostgreSQL Server hierarchy by performing a manual switchover.

Installation and Configuration - Adding PostgreSQL to an Existing LifeKeeper Configuration

This section covers installation and configuration of the SIOS Protection Suite and PostgreSQL software when adding PostgreSQL to an existing LifeKeeper cluster. These steps must be followed in order.

1. On the Primary Server:
    a. Use the **Window Disk Management** tool to configure your disk resources and define the shared or replicated volumes that you want to use. Be sure the volume size is adequate. If you are configuring shared volumes, power down the backup server during configuration to avoid simultaneous access to your storage.
    b. It is recommended that you use **Windows Explorer** to unshare all volumes to be used by the PostgreSQL Server Instance from the network.
    c. Configure your networking to support the SIOS Protection Suite TCP/IP comm path(s) and the switchable IP address if applicable.
2. On the Backup Server:
    a. Start the backup server if it was stopped to configure shared volumes in Step 1.
    b. Use the **Disk Management** utility to configure your disk resources and define the shared or replicated volumes that you want to use. If you are using shared storage, assign the same drive letter to the shared volume as assigned on the primary server. For replicated storage be sure the volume size is adequate.
3. On the Primary Server:
    a. In SIOS Protection Suite, create your shared or replicated Volume resource (where the PostgreSQL database cluster will reside) and extend it to the backup server. Later when you create your PostgreSQL Server resource hierarchy, SIOS Protection Suite will automatically bring the resource into the hierarchy as a dependency.
4. On the Backup Server:
    a. Bring the volume resource hierarchy In Service using the **LifeKeeper GUI**.
    b. Install the PostgreSQL Server software using the following guidelines:
        i. Using the --extract-only argument to the PostgreSQL installer is not recommend as it does not configure all of the information required by the PostgreSQL Recovery Kit.
        ii. By default the installation of the PostgreSQL software creates a single database cluster (a database cluster is a collection of databases that is managed by a single instance of a

running PostgreSQL database server). During installation the Data Directory prompt determines where the database cluster will be created. The location entered should be on the protected volume created above in Step 3a.

iii. The database service created during installation does not configure the postmaster process to start with the port argument (-p port) which is required by the SIOS Protection Suite PostgreSQL Server Recovery Kit to properly manage the instance. The Windows service (e.g. postgresql-x64-9.6 with PostgreSQL v9.6) created for the default database cluster will need to be updated to include this option if it will be protected by the SIOS Protection Suite. See Configuring the Postmaster Port Argument for more information.

c. Follow the steps outlined in Configure for Unattended Connections if the pgpass.conf will be used for authentication (if setting up a trust relationship via the pg_hba.conf file this step can be skipped as that will be done as part of the configuration on the primary server).

d. Verify the Windows PostgreSQL database cluster will start once the installation and configuration (for the Postmaster port argument and for unattended connections) is complete. This requires stopping and restarting the default Windows Service for PostgreSQL. Once the verification is complete, stop the PostgreSQL service.

5. On the Primary Server:

a. Bring the volume resource hierarchy In Service using the **LifeKeeper GUI**.

b. Open **Explorer** and access the drive associated with the replicated volume.

c. Delete the PostgreSQL database cluster directory created during the installation on the backup server. (You will recreate it in the next step).

d. Install the PostgreSQL Server software EXACTLY as you did on the backup server (program files in the same directory on the local disk and data files in the same location on the protected volume).

e. Follow the steps outlined in Configuring the Portmaster Port Argument on the primary server.

f. Follow the steps outlined in Configure for Unattended Connections on the primary server.

g. When the installation is complete, start the Windows PostgreSQL service to verify that the PostgreSQL Server can start properly on the primary server.

h. Create the PostgreSQL Server hierarchy on the primary server and extend it to the backup server. See Creating the PostgreSQL Hierarchy for more information. Test the new PostgreSQL Server hierarchy by performing a manual failover.

Additional Setup Tasks for Extended Configurations

If your configuration uses a shared storage device or you are using SIOS DataKeeper, you may choose a configuration that will be extended to a third (or more) server(s).

1. If it has not already been done, configure two systems following the steps given in Installation and Configuration - Adding PostgreSQL to an Existing LifeKeeper Configuration.

2. Use the **Disk Management** utility to configure your disk resources and define the shared or replicated volumes that you want to use. If you are using shared storage assign the same drive letter to the shared volume as assigned on the primary server. For replicated storage be sure the volume size is adequate.

3. Install the PostgreSQL Server software EXACTLY as you did on the primary server (program files in the same directory on the local disk and data files in the same location on the protected volume) using the following guidelines:

    a. Using the --extract-only argument to the PostgreSQL installer is not recommend as it does not configure all of the information required by the PostgreSQL Recovery Kit.

    b. By default the installation of the PostgreSQL software creates a single database cluster (a database cluster is a collection of databases that is managed by a single instance of a running PostgreSQL database server). During installation the Data Directory prompt determines where the database cluster will be created. Select any location as the data directory will be removed in a later step. If replicated storage is being used it can be the volume created above in step 2. If shared storage is being used do not use the volume located above to prevent overwriting the current PostgreSQL database cluster.

    c. The database service created during install does not configure the postmaster process to start with the port argument (-p port) which is required by the SIOS Protection Suite PostgreSQL Server Recovery Kit to properly manage the instance. The Windows service (e.g. postgresql-x64-9.6 with PostgreSQL v9.6) created for the default database cluster will need to be updated to include this option if it will be protected by the SIOS Protection Suite. See Configuring the Postmaster Port Argument for more information. While updating the Postmaster Port Argument you will also need to update the data directory path (-D argument value) to match the install location on the primary server.

4. Follow the steps outlined in Configure for Unattended Connections if the pgpass.conf will be used for authentication (if setting up a trust relationship via the pg_hba.conf file this step can be skipped as that will have already been done as part of the configuration on the primary server).

5. The default database cluster instance created during installation is not required and can be removed.

    a. Stop the database cluster instance created during installation.

    b. Open **Explorer** and migrate to the location of the data directory entered during installation of the PostgreSQL software.

    c. Delete the PostgreSQL database cluster directory.

    d. Delete the PostgreSQL service that was created during installation. You can use the Windows "sc delete <servicename>" command to do this.

6. If no comm paths exist to the new LifeKeeper cluster node, use the LifeKeeper GUI on the Primary Server to create comm paths between the primary and the new LifeKeeper cluster node and to create comm paths between the backup server and the new LifeKeeper cluster node.

7. Extend the PostgreSQL hierarchy to the new LifeKeeper cluster node. See Creating the PostgreSQL Hierarchy for more information. Test the new PostgreSQL Server hierarchy by performing a manual switchover to the new LifeKeeper cluster node.

PostgreSQL Server Configuration Considerations

Before you install and configure your PostgreSQL database clusters, it is important to understand how to configure them. It is also important to understand the concepts of Active/Standby and Active/Active configurations, and how they can be set up in a PostgreSQL configuration.

PostgreSQL Database Cluster Configuration Considerations

The SIOS Protection Suite for PostgreSQL uses Windows services for administration of the PostgreSQL database cluster. If a Windows Service does not already exist for the PostgreSQL database cluster, then one will be setup when the PostgreSQL hierarchy is created.

For SIOS Protection Suite to protect a PostgreSQL database cluster the following conditions must exist:

- The PostgreSQL Server must be running
- The PostgreSQL postmaster process must be running with the port option: -p port
- The PostgreSQL database cluster data directory must reside on a protected volume
- The PostgreSQL database cluster data directory, sub-directories, and all files must be accessible by the Windows Service account on all servers

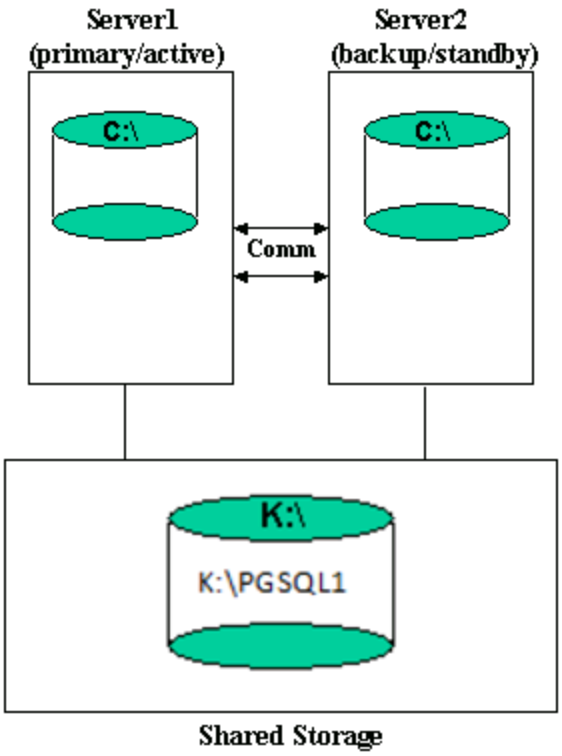
The following configuration limitations currently exist in the kit:

- Does not automatically include IP resource instances as part of the hierarchy. If a user connects remotely, then an IP resource will need to be created and added as a child resource in the PostgreSQL hierarchy.
- Only the location of the database cluster data directory is taken into consideration when determining which volume resources need to be included as part of the resource hierarchy. If any database table space is not located on the same protected volume as the data directory, then the volume containing the table space will need to be protected and added as a child resource in the PostgreSQL hierarchy.

PostgreSQL Active/Standby Configuration

A configuration is Active/Standby when there is only one PostgreSQL database cluster, located on a shared or replicated volume. The PostgreSQL database cluster services run on only one system at a time. The servers are assigned priorities within SIOS Protection Suite which determine the order of failover for a particular hierarchy.

The figure below depicts a single PostgreSQL instance installed on a pair of servers. The instance contains one database cluster, PGSQL1 residing on a single volume.

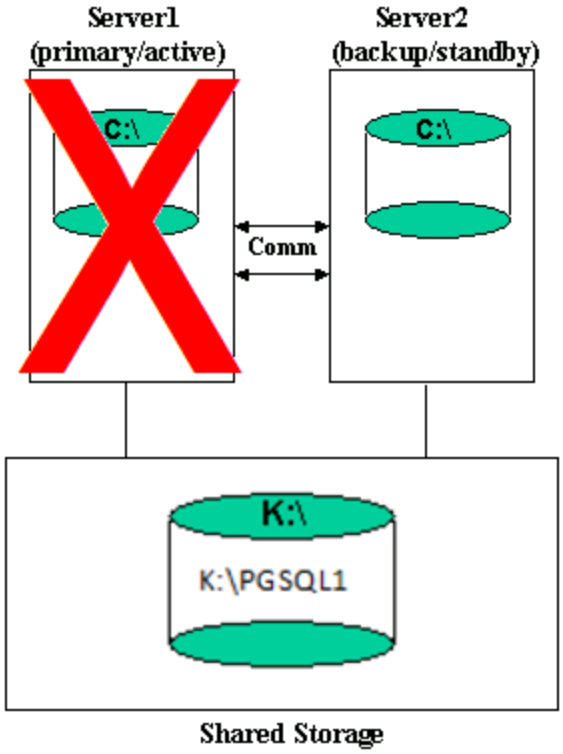


When you create the PostgreSQL hierarchy within SIOS Protection Suite, you are asked to specify the PostgreSQL data directory (database cluster location). If remote connections to the database cluster will be made then a protected IP resource will need to be configured and added to the resource hierarchy. SIOS Protection Suite then reads the configuration data for that instance and pulls the associated volumes into the hierarchy.

Once the hierarchy is created, it will appear as follows in the LifeKeeper GUI.

Active/Standby Failover

In the event of failure, SIOS Protection Suite brings the PostgreSQL Server hierarchy In Service on the backup Server. PostgreSQL Server is started on the backup server and it takes over protection of the database cluster as depicted in the figure below.

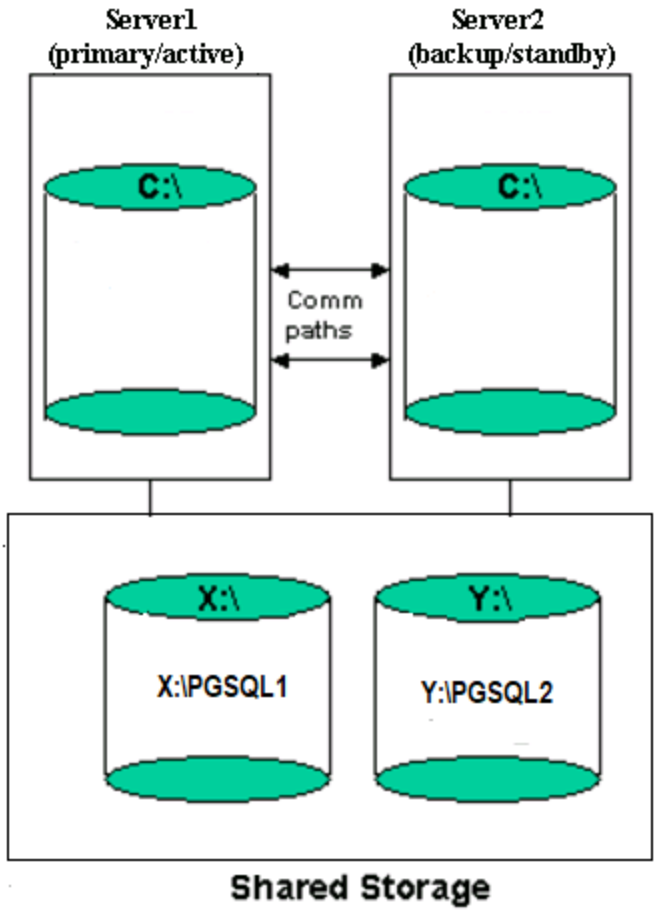


PostgreSQL Active/Active Configuration

Multiple PostgreSQL Server database clusters can be configured on any of the servers using initdb. SIOS Protection Suite can protect the multiple PostgreSQL database clusters in what is called an Active/Active configuration. SIOS Protection Suite identifies each instance by the port used for connections.

Each database cluster is protected in a single SIOS Protection Suite hierarchy.

The figure below depicts two PostgreSQL database clusters: PGSQL1 and PGSQL2.



Notes:

- In this configuration Server1 is the primary server for the PGSQL1 database cluster and Server2 is the primary server for the PGSQL2 database cluster.
- Each server can be the primary and backup server for multiple instances.
- It would be possible for Server1 or Server2 to be the primary server for both database clusters.

Creating the PostgreSQL Hierarchy

After you have completed the necessary setup tasks outlined in the SIOS Protection Suite for Windows Installation Guide, use the steps listed below to create the PostgreSQL Server hierarchy to protect your database cluster.

Important 	If you have an existing PosgreSQL database cluster installed, you may need to close any client applications (local or remote) that are accessing the PostgreSQL database cluster prior to completing this procedure. Closing all client connections is required if any of the following conditions exist: • The PostgreSQL database cluster data directory does not reside on a protected volume. To be highly available the PostgreSQL data directory must reside on a protected volume that can be switched between nodes in the LifeKeeper cluster. This will require manually moving the data directory prior to creating the resource hierarchy. Once the move is complete, restart the PostgreSQL database cluster services. • The PostgreSQL Server instance is not controlled by an existing Windows service. To facilitate the administration of the protected PostgreSQL Server, the hierarchy create will create a Windows service if one does not already exist. This requires stopping and restarting the PostgreSQL database cluster services. • The postmaster process is not running with the "-p port" option. See Configuring the Postmaster Port Argument for more information on how to verify and start PostgreSQL with the postmaster port argument.

1. From the LifeKeeper GUI menu, select **Edit** then **Server**. From the menu, select **Create Resource Hierarchy.**
2. The **Create Protected Application** dialog box will display. Select the **Primary** and **Backup** servers from the pull-down list. Select **Next** to continue.
3. The dialog box will appear with a drop down list box displaying all recognized recovery kits installed within the cluster. Select **PostgreSQL Server** and click **Next.**
4. You will be prompted to enter 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 an error requiring you to correct previously entered information. You may click **Cancel** at any time to cancel the entire creation process.

Field	Tips
PostgreSQL Service Name	Enter the Windows Service Name for this instance. If the name entered is an existing PostgreSQL service, then the PostgreSQL Recovery Kit will use the service information to pull information for the resource creation. If the name entered is not an existing PostgreSQL service, then the create process will require additional inputs in order to create a Windows Service using the entered name. **Note**: If a Windows Service is created during the PostgreSQL resource create, then a delete of the PostgreSQL resource will remove the PostgreSQL service. If the PostgreSQL service already exists at create time, then a delete of PostgreSQL resource will not remove the service.
PostgreSQL Executable Location	Select the location of the PostgreSQL executables (directory where pg_ctl.exe and psql.exe reside).
PostgreSQL Data Directory	Enter the path to the data directory for the PostgreSQL database cluster to be protected. If the PostgreSQL Service name entered exists, then it will default to the data directory defined for that service.
PostgreSQL Port	Select the port to be used for the PostgreSQL database cluster. This field is used to specify the TCP/IP port number on which the postmaster daemon is listening for connections from client applications. The default choice is obtained from the running PostgreSQL Server.
Enter Database Administrator User	Enter the name of the PostgreSQL database cluster administrative user. This user must have connection and administrative privileges.
The following fields only appear if the PostgreSQL Service Name entered does not already exist as a Windows service.	
PostgreSQL Service Logon Account	Enter the user account to be used for the logon credentials for the new PostgreSQL Windows Service. This account must have privileges to start and stop the PostgreSQL server instance being protected. This user account must also have full control file permissions for all files in the PostgreSQL Data Directory.
PostgreSQL Service Logon Password	Enter the user password to be used for the logon credentials for the new PostgreSQL Windows Service. **Note:** If built-in system accounts (e.g. "Local System" or "Network Service") are used, enter any non-blank value into this field (blank is not allowed). For built-in accounts, the password is ignored.

5. After you click **Create**, the **Wizard** will create your PostgreSQL resource. SIOS Protection Suite will validate the data entered. If SIOS Protection Suite detects a problem, an error message will appear in the information box.

6. Another information box will appear indicating that you have successfully created a PostgreSQL resource hierarchy, and you must **Extend** that hierarchy to another server in your cluster in order to achieve failover protection. Click **Next**.
7. After you click **Next**, SIOS Protection Suite will launch the **Pre-Extend Wizard**.

Extending a PostgreSQL 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. From the **Edit** menu, select **Resource** then **Extend Resource Hierarchy**. The **Pre-Extend Wizard** appears. If you are unfamiliar with the **Extend** operation, click **Next**.
2. The **Pre-Extend Wizard** will prompt you to enter the following information. **Note:** These first two fields appear only if you initiated the **Extend** from the **Edit** menu.

Field	Tips
Primary Server	Select a server where a resource hierarchy to be extended is currently defined and in service.
Resource Hierarchy to Extend	Select the resource hierarchy to extend.
Backup Server	Select a server to be the backup server for the resource hierarchy.

3. After receiving the message that the pre-extend checks were successful, click **Next**.
4. The Extend Wizard will prompt you to enter the following information. **Note:** The first two fields appear only if a Windows Service for the PostgreSQL Server does not exist on the backup server (the service name entered on the primary server during create is used for checking on the backup server).

Field	Tips
PostgreSQL Service Logon Account	Enter the user account to be used for the logon credentials for the PostgreSQL Windows Service on the backup server. This account must have privileges to start and stop the PostgreSQL server instance being protected.
PostgreSQL Service Logon Password	Enter the user password to be used for the logon credentials for the new PostgreSQL Windows Service.

Field	Tips
PostgreSQL Executable Location	Select the location of the PostgreSQL executables (directory where pg_ctl.exe and psql.exe reside).
Backup Priority	Enter a number between 1 and 999 to specify the template server's priority in the cascading failover sequence for this resource. A lower number means a higher priority. SIOS Protection Suite assigns the number "1" to the server on which the hierarchy was created. No two servers can have the same priority for a given resource.

5. Click **Extend**.

Unextending a PostgreSQL Hierarchy

To remove a resource hierarchy from a single server in the SIOS Protection Suite 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 SQL resource. It cannot be the server where the PostgreSQL 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 SQL 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.

Deleting a PostgreSQL Hiearchy

Before deleting a PostgreSQL hierarchy or instance, make sure that the hierarchy is active (green) on its primary server. You may also wish to remove the dependencies before deleting the hierarchy; otherwise, the dependencies will also be deleted.

To delete a resource hierarchy from all the servers in your SIOS Protection Suite environment, complete the following steps:

1. On the **Edit** menu, select **Resource**, then **Delete Resource Hierarchy**.
2. Select the **Target Server** where you will be deleting your PostgreSQL resource hierarchy and click **Next**. (This dialog will not appear if you selected the **Delete Resource** task by right-clicking on a resource instance in either pane.)
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 **Delete**.

5. Another information box appears confirming that the PostgreSQL resource was deleted successfully.
6. Click **Done** to exit.

Testing Your PostgreSQL Resource Hierarchy

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

Select **Edit**, then **Resource**, then **In Service**. For example, an In Service request executed on a backup server causes the application hierarchy to be taken out of service on the primary server and placed in service on the backup server. At this point, the original backup server is now the primary server and the original primary server has now become the backup server.

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

PostgreSQL Server Hierarchy Administration

Follow these guidelines when administering your PostgreSQL Server.

Access Via Protected Communication Paths

Reserve Volumes For Exclusive PostgreSQL Use

Start and Stop PostgreSQL Server Only Through SIOS Protection Suite

Creating and Protecting Additional PostgreSQL Database Clusters

PostgreSQL Administrative Login

Configuration for Unattended Connections

Configuring the Postmaster Port Argument

Monitoring Your PostgreSQL Hierarchy

Access Via Protected Communication Paths

All remote access of the service should be done through the hierarchy's LifeKeeper network resources. This will ensure that users can access the PostgreSQL database cluster regardless of which server it is currently running on.

Note: Currently the PostgreSQL recovery kit does not automatically include an IP communication resource as a dependent resource in the hierarchy. To ensure remote access regardless of which server is currently running the PostgreSQL database cluster, an IP resource will need to be created and added as a child to the PostgreSQL resource. Additionally, the PostgreSQL database cluster must be configured to listen on this address.

Reserve Volumes For Exclusive PostgreSQL Use

The volumes containing the protected PostgreSQL files should be reserved for use by PostgreSQL exclusively.

A SIOS Protection Suite protected volume may fail to switch over if it is accessed by an another application, process or remote user.

Start and Stop PostgreSQL Server Only Through SIOS Protection Suite

Although most of the administrative tasks for the PostgreSQL Server are done through the PostgreSQL tools, starting and stopping of the PostgreSQL Server should not be one of them:

1. **Consistent state** - When SIOS Protection Suite stops and starts the PostgreSQL Server, it maintains a consistent state for the protected Microsoft Service. Performing start and stop requests via the command line using the PostgreSQL tools such as pg_ctl.exe creates an inconsistent state for the Microsoft Service, as it is unable to detect the state as running or stopped. This can result in failures to detect and correct issues for the PostgreSQL Server.
2. **Protected PostgreSQL services** should be set to **Manual** startup mode through the **Control Panel "Services"** tool. **Note:** When creating a PostgreSQL resource hierarchy the protected service will automatically be set to **Manual** startup mode.

Creating and Protecting Additional PostgreSQL Database Clusters

As your environment grows, you may need to add new PostgreSQL Server database clusters on existing or new shared or replicated volumes.

To add and protect a new database cluster and the associated volume follow these steps:

1. **Create the volume resource.** On the server where the PostgreSQL database cluster will be placed, create and extend a volume resource.
2. **Create the database cluster.** Run `initdb` to create the new database cluster. Be sure to locate the data directory on the volume resource created above.
3. **Set access permissions.** In order to start and stop the PostgreSQL database cluster, the data directory and all files and sub-directories must have access rights that are recognized by all nodes in the cluster. By default the data directory and all files and sub-directories will have access rights based on the user running `initdb`. If the user is the local administrator, then attempting to start the database cluster on any other server will fail as that server will not have access rights. Either running `initdb` while logged on as a domain user or adding access rights for **NT AUTHORITY\NetworkService** is recommended. The account is used to provide access across all servers and should be the same account used for the Windows service logon credentials when creating the PostgreSQL resource.

4. **Configure for unattended connections.** The database cluster must be configured to allow connections without requiring a password. See Configuration for Unattended Connections for more information.
5. **Start the database cluster.** The database cluster must be running to create the PostgreSQL resource. Start the database cluster via pg_ctl.exe with the –o "-p port" argument. See Configuring the Postmaster Port Argument for more information on how to verify and start PostgreSQL with the postmaster port argument.
6. **Create the PostgreSQL resource.** On the server that the new database cluster was created on, create and extend a PostgreSQL resource. During the resource create you will be prompted for a Windows service name along with the logon credentials (user account and password). The Windows service name can be anything and will be used to create a Service account for administration (starting, stopping …) of the database cluster. The logon credentials used should be the ones setup in step 3 to ensure access rights on all servers.

PostgreSQL Administrative Login

During the creation of a SIOS Protection Suite PostgreSQL resource the user must enter a PostgreSQL administrative username for that database cluster. This administrative username is used for client connections through the *psql* utility.

The username:

1. Must have the ability to connect to the database (template1), as well as obtain the listing of defined databases for the instance.
2. Must have the ability to view system tables and make generalized queries.
3. Must allow unattended (non-terminal or scripted) connections. See Configuration for Unattended Connections for details.

Configuration for Unattended Connections

During the creation of a SIOS Protection Suite PostgreSQL resource, the user must enter a PostgreSQL administrative username for that database cluster. No password is requested for this username by the PostgreSQL recovery kit. Therefore, for the connection tests performed during health checking to be successful one of two configuration methods must be used:

- Trust configuration (no password)
- Credentials supplied via pgpass.conf

Trust Configuration

To configure trusted connections for the PostgreSQL database cluster modifications to the pg_hba.conf file are required. The authentication method for the administrative user for the database cluster must be set to

'trust'. The following is an example entry for the administrative user pgsql:

# TYPE	DATABASE	USER	ADDRESS	METHOD
# IPv4 local connections:				
host	all	pgsql	127.0.0.1/32	trust
# IPv6 local connections:				
host	all	pgsql	::1/128	trust

Credentials via pgpass.conf

To supply credentials for the PostgreSQL database cluster administrative user and password, creation of a pgpass.conf file for the logon account specified for the LifeKeeper service is required. This file must be created in the user's %APPDATA% folder. Login as the user account used to start LifeKeeper, and follow these steps:

- Change directories to %appdata%
- Create the directory postgresql if it does not exist
- Change directories to postgresql
- Create the file pgpass.conf with the following format:

  hostname:port:database:user:password

If the PostgreSQL database cluster administrative user password changes, then the pgpass.conf file will need to be updated with the password setting.

If a non-login user account (such as the built-in accounts "Network Service" or "Local System") is used to start LifeKeeper on a node, the %APPDATA% folder can be determined by running the following LifeKeeper command on a different node in the LifeKeeper cluster.

```
lcdremexec -d <node> -- echo $APPDATA
```

The pgpass.conf file can be created by a system administrator in that folder. Be sure to add read permissions for the LifeKeeper login account user to the pgpass.conf file that is created.

Configuring the Postmaster Port Argument

To properly manage the PostgreSQL database cluster the PostgreSQL Recovery Kit requires the postmaster process to be running with the port argument: "-p port". The port is required to create a PostgreSQL resource hierarchy and for monitoring once the hierarchy is created. To view the current argument list for the postmaster process see the postmaster.opts file located in the data directory for PostgreSQL database cluster. The following is an example of the contents of this file:

C:/Program Files/PostgreSQL/9.6/bin/postgres.exe "-D" "E:\PGSQL1" "-p" "5432"

In this example the postmaster process is running with the port argument. If the postmaster process is not running with the port argument it will need to be added via one of the two methods listed below:

- Adding the port argument to an existing Windows Service Instance
- Adding the port argument to a non-Windows Service Instance

Adding the port argument to an existing Windows Service Instance

If the PostgreSQL database cluster to be protected by the PostgreSQL recovery kit is running via a Windows Service, such as the postgresql-x64-9.6 service created by the initial install of the PostgreSQL software v9.6, then the startup command line for the service will need to be modified. This can be done via the following steps:

1. Stop the existing PostgreSQL database cluster instance. This can be done using several different methods:

    a. Use the Windows Service Interface (services.msc) and select to stop the service.

    b. Use the command line utility "sc", e.g. sc `stop service name`.

    c. Use the command "net", e.g `net stop service_name`.

2. Modify the startup command line for the service. This can be done via the command line using the "sc" utility or by editing the startup command line in the registry. **Note:** The startup command line for the PostgreSQL service uses the pg_ctl.exe utility. To pass arguments to the postmaster process from the pg_ctl utility requires the use of the "-o" argument. The "-o" argument takes a quoted list of postmaster startup options such as the "-p port" as used in the modifications shown below.

    To edit the startup command line using the "sc" command line utility follow these steps. This example uses the postgresql-x64-9.6 service:

    a. Retrieve the binPath (startup command line) for the service, e.g "`sc qc postgresql-x64-9.6`" it will return something like the following (for this example only the output line with the startup command line is shown):

    BINARY_PATH_NAME : "C:\Program Files\PostgreSQL\9.6\bin\pg_ctl.exe" runservice -N "postgresql-x64-9.6" -D "E:\PGSQL1" –w

    b. Update the binPath. **Note:** There is a space after the "=" for the binPath argument and the value must be in double quotes therefore it requires escaping the imbedded double quotes.

    sc config postgresql-x64-9.6 binPath= "\"C:\Program Files\PostgreSQL\9.6\bin\pg_ctl.exe\" runservice -N \"postgresql-x64-9.6\" -D \"E:\PGSQL1\" –w –o \"-p 5432\" "

    To edit the startup command line in registry requires the use of a registry edit tool such as regedit.

    a. Using regedit find the service ImagePath registry value: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\postgresql-x86-9.6\ImagePath

    b. Modify the ImagePath value by adding the port argument at the end of the existing command line: -o "-p 5432"

3. Restart the existing PostgreSQL database cluster instance. This can be done using several different methods:

a. Use the Windows Service Interface (services.msc) and select to start the service.

b. Use the command line utility “sc”, e.g. `sc start service_name`.

c. Use the command “net”, e.g `net start service_name`.

4. Verify the postmaster process is running with port argument by checking the postmaster.opts file as described above.

Adding the port argument to a non- Windows Service Instance

If a Windows Service does not exist for the PostgreSQL database cluster then stopping and restarting the PostgreSQL database cluster via the PostgreSQL utility pg_ctl.exe will be required. This can be done via the following steps:

1. Stop the PostgreSQL database cluster.

```
pg_ctl stop –D F:\PGSQL2
```

2. Start the PostgreSQL database cluster.

```
pg_ctl start –D F:\PGSQL2 –w –o "-p 5433"
```

3. Verify the postmaster process is running with port argument by checking the postmaster.opts file as described above.

Monitoring Your PostgreSQL Hierarchy

SIOS Protection Suite monitors the PostgreSQL database cluster services for every PostgreSQL resource. If the services stop, the monitoring process associated with the PostgreSQL resource will detect this and attempt to restart the services on the local server if Local Recovery is enabled. If Local Recovery is disabled, the resource will fail over to the backup server.

SIOS Protection Suite will also perform a query to test the connection to the protected PostgreSQL database cluster. If the query fails, the monitoring process associated with the PostgreSQL resource will detect this and attempt to restart the services if Local Recovery is enabled. If Local Recovery is disabled, the resource will fail over to the backup server.

Troubleshooting

This section provides suggestions and insights into occurrences that are not specifically related to the SIOS Protection Suite software but have a relationship with the total environment.

Create Fails

Restore Fails after Switchover

Create Fails

Symptom:

The Create of a PostgreSQL Server resource will fail if the database cluster data directory is located on a volume and that volume is not already protected by SIOS Protection Suite.

Suggested Action:

Create the volume resource for the data directory.

Symptom:

The Create of a PostgreSQL Server resource will fail if the postmaster process for PostgreSQL database cluster is not running with the “-p port” option.

Suggested Action:

To ensure access to the correct PostgreSQL database cluster the postmaster process must be running with the “-p port” argument. See Configuring the Postmaster Port Argument for more information on how to verify and start PostgreSQL with the postmaster port argument.

Symptom:

The Create of a PostgreSQL Server resource will hang after entering the administrative user if the unattended connection configuration is incorrect or non-existent.

Suggested Action:

Correctly configure unattended connections. See Configuration for Unattended Connections.

Restore Fails

Symptom:

The Restore of a PostgreSQL Server resource will fail after switchover if the database cluster data directory access permissions are configured incorrectly.

Suggested Action:

Verify the access permissions are configured correctly. See step 3 in Creating and Protecting Additional PostgreSQL Database Clusters.

Symptom:

The Restore of a PostgreSQL Server resource will fail if the database cluster instance is started via **pg_ctl.exe start** and not via an in service action in LifeKeeper or via a service start via Windows APIs. Using pg_ctl.exe to start the database cluster creates an inconsistency in the Windows Service state causing a LifeKeeper restore to fail on the attempt to start an already running instance.

When attempting to start an already running instance, PostgreSQL will log the following messages:

> FATAL: lock file "postmaster.pid" already exists
>
> HINT: Is another postmaster (PID 3488) running in data directory "E:/PGSQL1"?

Suggested Action:

To correct this condition the database cluster must be stopped via **pg_ctl stop**. Once the stop completes the LifeKeeper in service action should be successful.

Symptom:

The Restore of a PostgreSQL Server resource can fail if the database cluster did not shut down cleanly because of server crash or the PostgreSQL service was hung when the shutdown occurred (windbg was used to simulate a hang). The inability to shutdown cleanly will force a database cluster recovery action on the next startup. This recovery action can cause the Window's Service start action to fail placing the service in an inconsistent state with the database cluster state. During startup after a unclean shutdown, PostgreSQL may log the following messages (along with a number of others):

> Waiting for server start up
>
> LOG: database system was interrupted; last known up at 2017-07-25 16:12:10 EDT
>
> FATAL: the database system is starting up
>
> LOG: database system was not properly shut down; automatic recovery in progress

Once the recovery is complete the PostgreSQL database cluster processes are running but the Window's Service state is "Stopped" and the LifeKeeper PostgreSQL resource is in the failed state. If a LifeKeeper restore action is attempted with the database cluster up and running, PostgreSQL will log the following messages:

> FATAL: lock file "postmaster.pid" already exists
>
> HINT: Is another postmaster (PID 3488) running in data directory "E:/PGSQL1"?

Suggested Action:

To correct this condition the database cluster must be stopped via **pg_ctl stop** once the recovery is complete. Once the stop completes the LifeKeeper in service action should be successful.

Tunable Settings for the PostgreSQL Recovery Kit

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 %LKROOT%\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_START_RETRIES**

  This tunable controls the number of times the PostgreSQL Recovery Kit will loop waiting for the database cluster to start before giving up with a failed status. There is a 5 second wait between each retry. By default the Recovery Kit will retry 12 times resulting is a 60 second wait for the database cluster to startup. The minimum value for this tunable setting is 12.

- **LKPGSQL_STOP_RETRIES**

  This tunable controls the number of times the PostgreSQL Recovery Kit will loop waiting for the database cluster to stop before giving up with a failed status. There is a 5 second wait between each retry. By default the Recovery Kit will retry 12 times resulting is a 60 second wait for the database cluster to stop. The minimum value for this tunable setting is 12.

- **LKPGSQL_RESTORE_CONNECT_RETRIES**

  This tunable controls the number of times during a restore action that the PostgreSQL Recovery Kit will loop waiting for the database cluster to respond to a connection request. If the number of retries is reached the restore action will be failed. There is a 5 second wait between each retry. By default the Recovery Kit will retry 2 times resulting is a 10 second wait for the database cluster to respond to the connection request. The minimum value for this tunable setting is 2.

Tunable Settings for the PostgreSQL Database Cluster

To check the health of the PostgreSQL database cluster, the Recovery Kit's deepchk script will attempt to connect to the database cluster's template1 db. This connection operation can fail if all the available connections are in use. This can be prevented via the PostgreSQL super user connection tunable.

- superuser_reserved_connections
  This tunable in the database cluster postgresql.conf file controls the number of super user connections allowed. By default this setting is commented out. This setting should be uncommented. For the change to take effect, the PostgreSQL database cluster will need to be stopped and restarted.