Postgres Enterprise Manager

Release 7.14

PEM Linux Installation Guide

Mar 17, 2021

Contents

Postgres Enterprise Manager (PEM) is designed to assist database administrators, system architects, and performance analysts when administering, monitoring, and tuning PostgreSQL and Advanced Server database servers. PEM has been designed to manage and monitor a single server or multiple servers from a single console, allowing complete control over monitored databases.

This document provides step-by-step instructions to guide you through the installation of Postgres Enterprise Manager on a Linux host.

For information about the platforms and versions supported by PEM, visit the EnterpriseDB website at:

https://www.enterprisedb.com/services-support/edb-supported-products-and-platforms

Throughout this guide, the term *Postgres* refers to either a PostgreSQL or an Advanced Server installation, where either is appropriate.

Language pack installers contain supported languages that may be used with EDB Postgres Advanced Server and EnterpriseDB PostgreSQL database installers. The language pack installer allows you to install Perl, TCL/TK, and Python without installing supporting software from third party vendors. For more information about installing and using Language Pack, please see the *EDB Postgres Language Pack Guide*, available from the EnterpriseDB Website.

CHAPTER 1

What's New

The following features have been added to create Postgres Enterprise Manager 7.14:

- Search Object Functionality: This functionality helps you to search a database object (schema, index, table, etc.) within a database using a text string and wildcard, and locate them in the browser tree.
- SMTP emails and SNMP traps: Newer versions of PEM Agents will be able to send SMTP emails and SNMP traps using multiple PEM Agents simultaneously. The users will get the notification even when the PEM Agent on PEM Server is `down`.
- `Unmanaged` state on the Monitoring dashboard: The database servers registered with PEM for administration only are not bound to any PEM Agent. Those servers state will be identified as `Unmanaged` on the Monitoring dashboard instead of `Unknown`.
- 4096-bit key for generating the SSL certificates: The PEM configuration script, installer, and PEM Agent are modified to generate certificates using the 4096-bit key to improve security.
- Support for the SHA256 algorithm (sslutils v1.3): Signing certificate algorithm is changed to SHA256 from SHA1 to improve security.
- Other features and changes include:
  - Enhancements to the Schema Diff tool (Beta) add support for Packages, Sequences, Synonyms, Domain, Domain Constraints, Collation, FTS Configuration, FTS Dictionary, FTS Parser, FTS Template, Foreign Tables
  - Improved accessibility support in different third party libraries
  - Added support for Python 3.8
  - Added support for the `toast_tuple_target` and `parallel_workers` parameters
  - The ability to warn the user if an unsupported, deprecated or unknown browser is detected
  - Support for a new v3 version of REST API, which includes SNMP v3 support

The Package Deployment wizard and Streaming Replication wizard are no longer available from PEM v7.14 forward.

CHAPTER 2

Postgres Enterprise Manager - Overview

Postgres Enterprise Manager (PEM) consists of components that provide the management and analytical features of PEM:

- **PEM server:** The PEM server is used as the data repository for monitoring data and as a server to which both agents and clients connect. The PEM server consists of an instance of PostgreSQL and an associated database for storage of monitoring data, and a server that provides web services.
- **PEM web interface:** The PEM web interface allows you to manage and monitor Postgres servers and utilize PEM extended functionality. The web interface software is installed with the PEM server installer, and is accessed via your choice of web browser.
- **PEM agent:** The PEM agent is responsible for executing tasks and reporting statistics from the agent host and monitored Postgres instances to the PEM server. A single PEM agent can monitor multiple installed instances of Postgres that reside on one or many hosts.

  The PEM agent installer creates two executables: the PEM worker (`pemworker.exe`) and the PEM agent (`pemagent.exe`). Each PEM worker has a corresponding PEM agent that you can use to start or stop the PEM worker. The PEM agent will also restart the PEM worker should it terminate unexpectedly.

  The PEM worker log file contains information related to PEM worker activity (probe activities, heartbeat responses, etc.), and is stored in `/var/log/pem/worker.log`.
- **SQL Profiler plugin:** This plugin to the Postgres server is used to generate the monitoring data used by the SQL Profiler tool. Installation of the SQL Profiler plugin is optional, but the plugin must be installed into each instance of Postgres you wish to profile. The SQL Profiler may be used with any supported version of an EnterpriseDB distribution of a PostgreSQL server or an Advanced Server (not just those managed through the PEM server).See the PEM SQL Profiler User's Guide for details and supported versions.

The architectural diagram below illustrates the relationship between the various servers and workstations involved in a typical PEM installation.

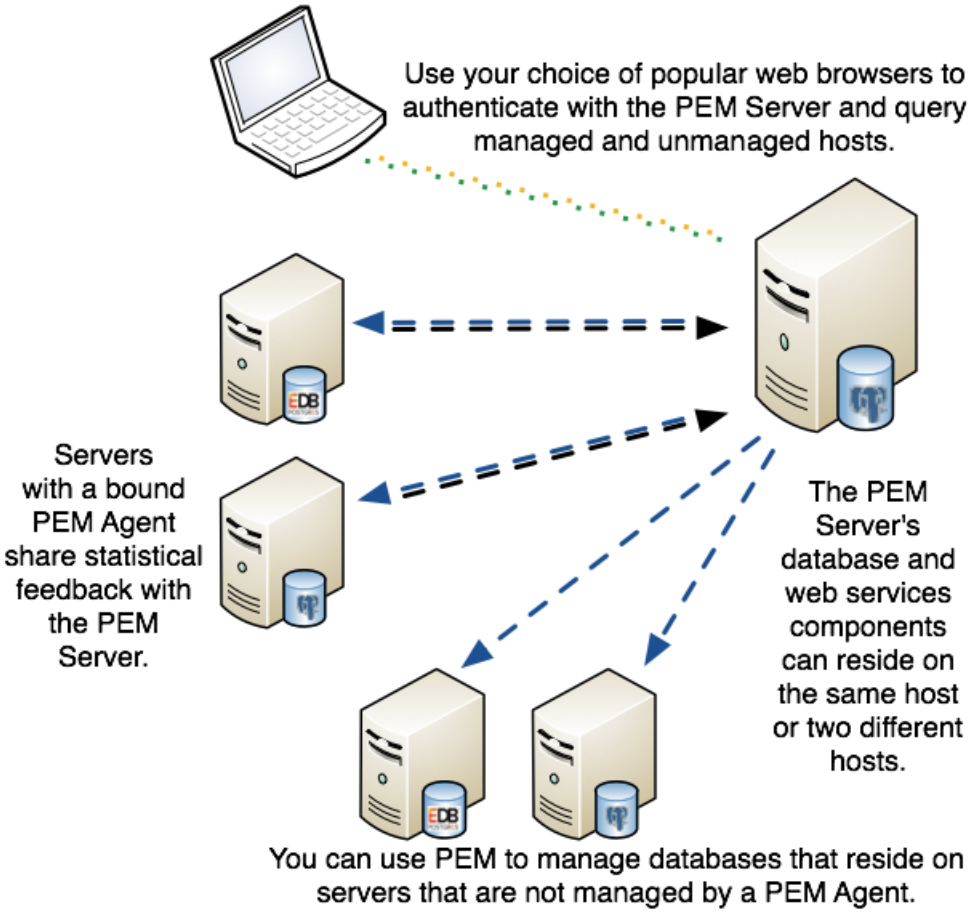

Fig. 1: *A typical PEM installation*

2.1 Hardware Prerequisites

For optimum speed when monitoring servers and rendering dashboards, we recommend installing PEM on a system with at least:

- 4 CPU cores
- 8 GB of RAM
- 100 GB of Storage

Additional disk space is required for data storage. Please note that resource usage will vary based on which probes are defined and enabled, and the activity level on the monitored databases. Monitoring server resources (as you use PEM) will let you know when you need to expand your initial system configuration.

2.2 Software Prerequisites

Platforms and Versions Support

For information about the platforms and versions supported by PEM, visit the EnterpriseDB website at:

https://www.enterprisedb.com/services-support/edb-supported-products-and-platforms

---

Note: PEM 7.14 is no longer supported on CentOS/RHEL/OEL 6.x platforms. It is strongly recommended that EDB products running on these platforms be migrated to a supported platform.

Modifying the pg_hba.conf File

The `pg_hba.conf` file manages connections for the Postgres server. You must ensure that the `pg_hba.conf` file on each monitored server allows connections from the PEM server, the monitoring PEM agent, and the host of the PEM-HTTPD server.

For information about modifying the `pg_hba.conf` file, see the *PEM Administrator's Guide* available at:

https://www.enterprisedb.com/edb-docs

Information about managing authentication is also available in the Postgres core documentation available at:

https://www.postgresql.org/docs/current/static/auth-pg-hba-conf.html

Firewall Restrictions

Please note that you must adjust your firewall to allow communication between PEM components.

Supported Locales

Currently, the PEM server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale, or a separator character other than a period may result in errors.

CHAPTER 3

Installing Postgres Enterprise Manager

The `edb-pem` package for Linux platforms installs the PEM Server, a PEM Agent, and the required software to connect to the PEM web interface with your choice of your browser.

The PEM server uses a Postgres installation and backing database to manage data. The `pem` backing database gets created while configuring PEM.

For detailed information about installing and configuring PEM Server see *Installing the PEM Server on Linux*

The PEM agent that is installed with the PEM server is capable of monitoring multiple servers that reside on the same host, or on remote hosts. Please note that the PEM functionality on servers monitored by a remote agent may be limited.

For detailed information about installing and configuring PEM Agent see *Installing the PEM Agent on Linux*

3.1 Installing the PEM Server on Linux

When installing a PEM server on a Linux host, you must first install a backing database and create the `pem` database cluster. The server's backing database may be installed via an RPM package for Linux. The database must be one of the following versions:

- EDB Postgres Advanced Server version 9.6 or above
- PostgreSQL version 9.6 or above

For detailed information about installing an Advanced Server or PostgreSQL database, please see the product documentation at the EnterpriseDB website.

The `pg_hba.conf` file on the backing database must be configured to use `trust` authentication for connections. For information about modifying the pg_hba.conf file, see the PostgreSQL core documentation.

If you are using a PostgreSQL database, you must also install the hstore contrib module.

If you are using a firewall, you must allow access to port `8443` on the PEM backing database; use the commands:

```
firewall-cmd --permanent --zone=public --add-port=8443/tcp
firewall-cmd --reload
```

3.1.1 Installing the PEM Server on a CentOS or RHEL Host

In addition to the above considerations, the following prerequisites are applicable if you are using a RHEL or CentOS host:

1. You must install the `epel-release` package:

   ```
   yum install epel-release
   ```

   Note: You may need to enable the `[extras]` repository definition in the `CentOS-Base.repo` file (located in `/etc/yum.repos.d`).

   If you are a Red Hat Network user you must also enable the `rhel-<x>-server-optional-rpms` repository to use EPEL packages, where *x* specifies the version of RHEL on the host. You can make the repository accessible by enabling the `RHEL optional subchannel` for `RHN-Classic`. If you have a certificate-based subscription, then you must also enable `rhel-<x>-server-eus-optional-rpms` repository to use EPEL packages or please see the `Red Hat Subscription Management Guide` for the required repository.

2. You must also enable the `rhel-<x>-server-extras-rpms` repository, where `x` specifies the version of the RHEL on the host.

3. You must also have credentials that allow access to the EnterpriseDB repository. To request credentials, visit:

   EnterpriseDB Repository Access Steps.

4. Create a repository configuration file; assume superuser privileges, and invoke the following command:

   ```
   yum -y install https://yum.enterprisedb.com/edb-repo-rpms/
   edb-repo-latest.noarch.rpm
   ```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

5. After creating the `edb.repo` file, use your choice of editor to ensure that the value of the enabled parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EnterpriseDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:<password>@yum.enterprisedb.com/edb/redhat/rhel-
↪$releasever-$basearch
```

(continues on next page)

(continued from previous page)

```
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

7. Use yum to install the PEM server:

   ```
   yum install edb-pem
   ```

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

If you want to install PEM server on a machine that is in isolated network, you must first create PEM repository on that machine. For more information about creating PEM repository on an isolated network, see *Creating a PEM repository in an Isolated Network*.

3.1.2 Installing the PEM Server on a Debian or Ubuntu Host

To install PEM on a Debian or Ubuntu host, you must have credentials that allow access to the EnterpriseDB repository. To request credentials for the repository, contact EnterpriseDB .

The following steps will walk you through using the EnterpriseDB apt repository to install a Debian package. When using the commands, replace the username and password with the credentials provided by EnterpriseDB.

1. Go to https://apt.enterprisedb.com/ and log in as root:

   ```
   sudo su -
   ```

2. Configure the EnterpriseDB repository:

   ```
   sh -c 'echo "deb https://username:password@apt.enterprisedb.com
   $(lsb_release - cs)-edb/ $(lsb_release -cs) main" > /etc/apt/
   sources.list.d/edb-$(lsb_release -cs).list'
   ```

3. Add support to your system for secure APT repositories:

   ```
   apt-get install apt-transport-https
   ```

4. Add the EBD signing key:

   ```
   wget -q -O -https://username:password@apt.enterprisedb.com/edb-deb.
   gpg.key | apt-key add -
   ```

5. Update the repository metadata:

   ```
   apt-get update
   ```

6. Use the following command to install the Debian package for the PEM server:

   ```
   apt-get install edb-pem
   ```

3.1.3 Installing PEM Server on a SLES Host

For detailed information about installing Advanced Server and supporting components on a SLES host, please consult the *EDB Postgres Advanced Server Installation Guide*,available at:

https://www.enterprisedb.com/edb-docs/p/edb-postgres-enterprise-manager

SLES packages are available from:

https://zypp.enterprisedb.com

Before installing PEM, you must install prerequisite packages. Invoke the following commands, replacing *sp_no* with the service pack that you are using (i.e. SP4):

```
SUSEConnect -p sle-module-legacy/12/x86_64

SUSEConnect -p sle-sdk/12/x86_64

zypper addrepo https://download.opensuse.org/repositories/
Apache:Modules/SLE_12_<sp_no>/Apache:Modules.repo

zypper addrepo http://download.opensuse.org/repositories/
Cloud:/OpenStack:/Newton:/cisco-apic:/2.3.1/SLE_12_<sp_no>/
pem_opensuse_boost

zypper refresh

zypper install edb-pem
```

3.1.4 Configuring the PEM Server

Before configuring the PEM server, ensure that the `sslutils` extension is installed for your backing database.

- For an Advanced Server backing database, `sslutils` extension is by default installed along with Advanced Server.
- If you are using a PostgreSQL backing database, ensure you have access to the PostgreSQL community repository, and use the command:

```
yum install sslutils_<x> postgresql<X>-contrib
```

Where, `x` is the server version.

The PEM server installer includes a script (`configure-pem-server.sh`) to help automate the configuration process for RPM installations. The script is installed in the `/usr/edb/pem/bin` directory. To invoke the script, use the command:

```
/usr/edb/pem/bin/configure-pem-server.sh
```

When invoking the script, you can include command line options to specify configuration properties; the script will prompt you for values that you omit on the command line. The accepted options are:

Option	Description
-acp	Defines PEM Agent certificate path. The default is /root/.pem.
-ci	CIDR formatted network address range that agents will connect to the server from, to be added to the server's pg_hba.conf file. For example, 192.168.1.0/24. The default is 0.0.0.0/0.
-dbi	The directory for the database server installation. For example, /usr/edb/as10 for Advanced Server or /usr/pgsql-10 for PostgreSQL.
-ds	The unit file name of the PEM database server. For Advanced Server, the default file name is edb-as-10; for PostgreSQL, it is postgresql-10.
-ho	The host address of the PEM database server.
-p	The port number of the PEM database server.
-ps	The service name of the pemagent; the default value is pemagent.
-sp	The superuser password of the PEM database server. This value is required.
-su	The superuser name of the PEM database server.
-t	The installation type: Specify 1 if the configuration is for web services and backing database, 2 if you are configuring web services, or 3 if you are configuring the backing database. If you specify 3, please note that the database must reside on the local host.

If you do not provide configuration properties on the command line, you will be prompted for values by the script. When you invoke the script, choose from:

1. Web Services and Database - Select this option if the web server and database both reside on the same host as the PEM server.

2. Web Services - Select this option if the web server resides on a different host than the PEM server.

3. Database - Select this option to configure the PEM backing database for use by the PEM server. Please note that the specified database must reside on the local host.

Note: If the web server and the backing database reside on separate hosts, configure the database server first (option 3), and then web services (option 2). The script will exit if the backing database is not configured before web services.

After selecting a configuration option, the script will proceed to prompt you for configuration properties. When the script completes, it will create the objects required by the PEM server, or perform the configuration steps required.

To view script-related help, use the command:

```
/usr/edb/pem/bin/configure-pem-server.sh -help
```

After configuring the PEM server, you can access the PEM web interface in your browser. Navigate to:

```
https://<ip_address_of_PEM_server>:8443/pem
```

3.2 Creating a Repository in an Isolated Network

You can create a local repository to act as a host for RPM packages if the server on which you wish to install PEM cannot directly access the EnterpriseDB repository. Please note that this is a high-level overview of the steps required; you may need to modify the process for your individual network. To create and use a local repository, you must:

1. Use a system with Internet access to download all the dependencies required:

```
yum install yum-plugin-downloadonly

mkdir /tmp/<pem_dir>

yum install --downloadonly --downloaddir=/tmp/<pem_dir>/ edb-pem

mkdir /tmp/<epel_dir>

yum install --downloadonly --downloaddir=/tmp/<epel_dir>/ epel-release*
```

Where `<pem_dir>` and `<epel_dir>` are the local directories that you create for downloading the RPMs.

2. Copy the `/tmp/<pem_dir>` and `/tmp/<epel_dir>` directories to the machine that is on the isolated network.

3. Create the repositories:

```
yum install createrepo

createrepo /tmp/<pem_dir>

createrepo /tmp/<epel_dir>
```

5. Create a repository configuration file called `/etc/yum.repos.d/pem.repo` with connection information that specifies:

```
[pemrepo]
name=PEM Repository
baseurl=file:///tmp//<pem_dir>/
enabled=1
gpgcheck=0
```

6. Create a repository configuration file called `/etc/yum.repos.d/epel.repo` with connection information that specifies:

```
[epelrepo]
name=epel Repository
baseurl=file:///tmp/<epel_dir>/
enabled=1
gpgcheck=0
```

After specifying the location and connection information for your local repository, you can use yum commands to install or upgrade PEM server:

- To install PEM server:

  ```
  yum install edb-pem
  ```

- To upgrade PEM server:

  ```
  yum upgrade edb-pem
  ```

For more information about creating a local yum repository, visit: https://wiki.centos.org/HowTos/CreateLocalRepos

3.3 Installing a PEM Agent on Linux

A PEM agent may monitor one or more servers on one or more hosts. For comprehensive information about managing a PEM agent, see the PEM Agent User Guide.

3.3.1 Installing a PEM agent on a CentOS or RHEL host

On a Linux system, you can use the `yum` package manager to install a PEM agent. Please note that before using a package manager to install the PEM agent on a host, you must:

- Install the `epel-release` package on the host by running any one of the following commands:
- `yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm`
- `yum install epel-release`

Note: You may need to enable the `[extras]` repository definition in the `CentOS-Base.repo` file (located in `/etc/yum.repos.d`).

- You must also have credentials that allow access to the EnterpriseDB repository. For information about requesting credentials, visit:

  EnterpriseDB Repository Access Steps.

After receiving your repository credentials you can:

1. Create the repository configuration file.
2. Modify the file, providing your user name and password.
3. Install the `edb-pem-agent` package.

Creating a Repository Configuration File

To create the repository configuration file, assume superuser privileges, and invoke the following command:

```
yum -y install https://yum.enterprisedb.com/edb-repo-rpms/edb-repo-latest.noarch.rpm
```

The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.

Modifying the file, providing your user name and password

After creating the `edb.repo` file, use your choice of editor to ensure that the value of the enabled parameter is `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EnterpriseDB user.

```
[edb]
name=EnterpriseDB RPMs $releasever - $basearch
baseurl=https://<username>:<password>@yum.enterprisedb.com/edb/redhat/rhel-
↪$releasever-$basearch
```

(continues on next page)

(continued from previous page)

```
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

If you want to install PEM Agent on a machine that is in isolated network, you must first create PEM repository on that machine. For more information about creating PEM repository on an isolated network, see *Creating a PEM repository in an Isolated Network*.

Installing the PEM Agent

After saving your changes to the configuration file, you can use the `yum install` command to install `edb-pem-agent`:

```
yum install edb-pem-agent
```

When the installation is complete, `yum` will display a list of the installed packages and dependencies.

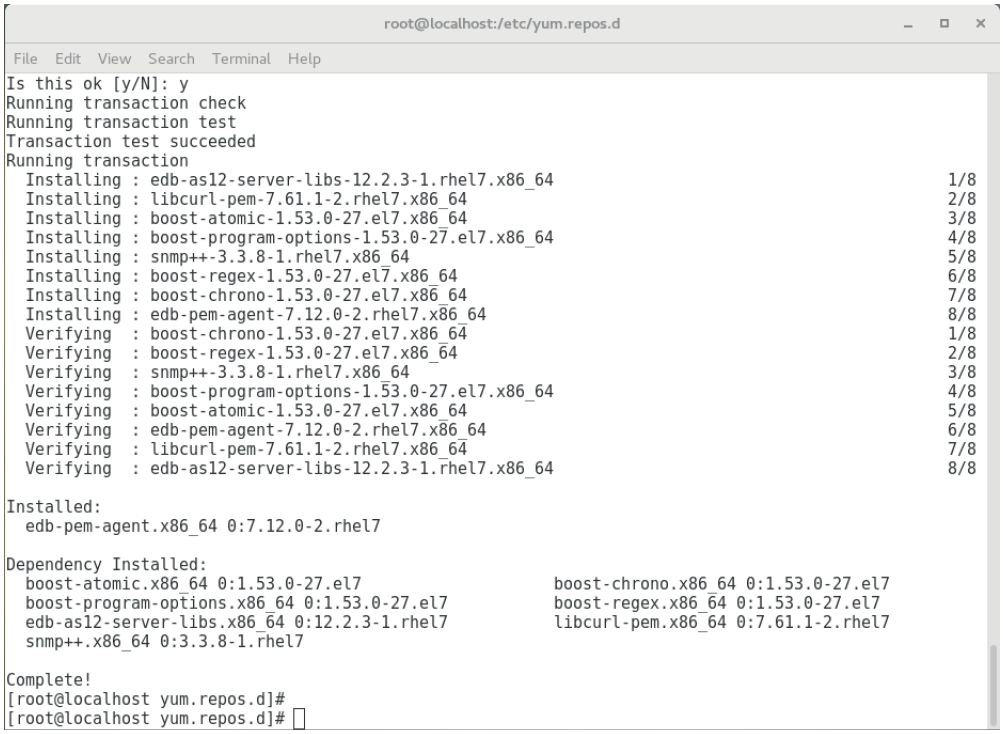

Fig. 1: *Using an RPM package to install the PEM agent*

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

3.3.2 Installing a PEM Agent on a Debian or Ubuntu Host

To install PEM on a Debian or Ubuntu host, you must have credentials that allow access to the EnterpriseDB repository. To request credentials for the repository, contact EnterpriseDB .

The following steps will walk you through using the EnterpriseDB apt repository to install a Debian package. When using the commands, replace the username and password with the credentials provided by EnterpriseDB.

1. Go to https://apt.enterprisedb.com/ and log in as root:

   ```
   sudo su -
   ```

2. Configure the EnterpriseDB repository:

   ```
   sh -c 'echo "deb https://username:password@apt.enterprisedb.com/
   $(lsb_release - cs)-edb/ $(lsb_release -cs) main" > /etc/apt/
   sources.list.d/edb-$(lsb_release -cs).list'
   ```

3. Add support to your system for secure APT repositories:

   ```
   apt-get install apt-transport-https
   ```

4. Add the EBD signing key:

   ```
   wget -q -O -https://username:password@apt.enterprisedb.com/edb-deb.
   gpg.key | apt-key add -
   ```

5. Update the repository metadata:

   ```
   apt-get update
   ```

6. Use the following command to install the Debian package for the PEM agent:

   ```
   apt-get install edb-pem-agent
   ```

3.3.3 Installing a PEM Agent on a SLES Host

For detailed information about installing Advanced Server and supporting components on a SLES host, please consult the

EDB Postgres Advanced Server Installation Guide

SLES packages are available from:

> https://zypp.enterprisedb.com

Before installing PEM, you must install prerequisite packages. Use the following commands replacing *sp_no* with the service pack that you are using (i.e. SP2 or SP3):

```
SUSEConnect -p sle-module-legacy/12/x86_64

SUSEConnect -p sle-sdk/12/x86_64

zypper addrepo https://download.opensuse.org/repositories/Apache:Modules/SLE_
↪12_<sp_no>/Apache:Modules.repo

zypper addrepo http://download.opensuse.org/repositories/Cloud:/OpenStack:/
↪Newton:/cisco-apic:/2.3.1/SLE_12_<sp_no>/ pem_opensuse_boost

zypper refresh

zypper install edb-pem-agent
```

3.3.4 Registering an Agent

Each PEM agent must be *registered* with the PEM server. The registration process provides the PEM server with the information it needs to communicate with the agent. The PEM agent graphical installer supports agent self-registration, but you can use the `pemworker` utility to register the agent if you skip PEM agent registration during a graphical installation or use an RPM package to install a PEM agent.

The RPM installer places the PEM agent in the `/usr/edb/pem/agent/bin` directory. To register an agent, include the `--register-agent` keywords along with registration details when invoking the `pemworker` utility:

```
pemworker --register-agent
```

Append command line options to the command string when invoking the pemworker utility. Each option should be followed by a corresponding value:

Option	Description
`--pem-server`	Specifies the IP address of the PEM backend database server. This parameter is required.
`--pem-port`	Specifies the port of the PEM backend database server. The default value is 5432.
`--pem-user`	Specifies the name of the Database user (having superuser privileges) of the PEM backend database server. This parameter is required.
`--pem-agent-user`	Specifies the agent user to connect the PEM server backend database server.
`--cert-path`	Specifies the complete path to the directory in which certificates will be created. If you do not provide a path, certificates will be created in: On Linux, ~/.pem On Windows, %APPDATA%/pem
`--config-dir`	Specifies the directory path where configuration file can be found. The default is the <pemworker path>/../etc.
`--display-name`	Specifies a user-friendly name for the agent that will be displayed in the PEM Browser tree control. The default is the system hostname.
`--force-registration`	Include the force_registration clause to instruct the PEM server to register the agent with the arguments provided; this clause is useful if you are overriding an existing agent configuration. The default value is Yes.
`--group`	The name of the group in which the agent will be displayed.
`--team`	The name of the database role, on the PEM backend database server, that should have access to the monitored database server.

continues on next page

Table 1 – continued from previous page

Option	Description
`--owner`	The name of the database user, on the PEM back-end database server, who will own the agent.
`--allow_server_restart`	Enable the allow-server_restart parameter to allow PEM to restart the monitored server. The default value is True.
`--allow-batch-probes`	Enable the allow-batch-probes parameter to allow PEM to run batch probes on this agent. The default value is False.
`--batch-script-user`	Specifies the operating system user that should be used for executing the batch/shell scripts. The default value is none; the scripts will not be executed if you leave this parameter blank or the specified user does not exist.
`--enable-heartbeat-connection`	Enable the enable-heartbeat-connection parameter to create a dedicated heartbeat connection between PEM Agent and server to update the active status. The default value is False.
`--enable-smtp`	When set to true for multiple PEM Agents (7.13 or lesser) then it may send more duplicate emails. Whereas for PEM Agents (7.14 or higher) then it may send lesser duplicate emails.
`--enable-snmp`	When set to true for multiple PEM Agents (7.13 or lesser) then it may send more duplicate traps. Whereas for PEM Agents (7.14 or higher) then it may send lesser duplicate traps.
`-o`	Specify if you want to override the configuration file options.

If you want to use any PEM feature for which database server restart is required by the pemagent such as Audit Manager, Log Manager, or Tuning Wizard, then you must set the value for `allow_server_restart` as `true` in the `agent.cfg` file.

Note: When configuring a shell/batch script run by a PEM agent that has PEM 7.11 or later version installed, the user for the `batch_script_user parameter` must be specified. It is strongly recommended that a non-root user is used to run the scripts. Using the root user may result in compromising the data security and operating system security. However, if you want to restore the pemagent to its original settings using `root` user to run the scripts, then the `batch_script_user` parameter value must be set to `root`.

Before any changes are made on the PEM database, the connecting agent is authenticated with the PEM database server. When invoking the pemworker utility, you must provide the password associated with the PEM server administrative user role (`postgres`). There are three ways to specify the administrative password; you can:

- set the `PEM_MONITORED_SERVER_PASSWORD` environment variable.

- provide the password on the command line with the `PGPASSWORD` keyword.
- create an entry in the `.pgpass` file.

Failure to provide the password will result in a password authentication error; you will be prompted for any other required but omitted information. When the registration is complete, the server will confirm that the agent has been successfully registered.

Setting PEM Agent Configuration Parameters

The PEM agent RPM installer creates a sample configuration file named `agent.cfg.sample` in the `/usr/edb/pem/agent/etc` directory. When you register the PEM agent, the `pemworker` program creates the actual agent configuration file (named `agent.cfg`). You must modify the `agent.cfg` file, adding the following configuration parameter:

```
heartbeat_connection = true
```

You must also add the location of the `ca-bundle.crt` file (the certificate authority). By default, the installer creates a `ca-bundle.crt` file in the location specified in your `agent.cfg.sample` file. You can copy the default parameter value from the sample file, or, if you use a `ca-bundle.crt` file that is stored in a different location, specify that value in the `ca_file` parameter:

```
ca_file=/usr/libexec/libcurl-pem7/share/certs/ca-bundle.crt
```

Then, use a platform-specific command to start the PEM agent service; the service is named `pemagent`. For example, on a CentOS or RHEL 7.x or 8.x host, use systemctl to start the service:

```
systemctl start pemagent
```

The service will confirm that it is starting the agent; when the agent is registered and started, it will be displayed on the `Global Overview` and in the `Object browser` of the PEM web interface.

For information about using the `pemworker` utility to register a server, please see the PEM Administrator's Guide

CHAPTER 4

The PEM Web Interface

After installing a PEM server and agent, you can configure PEM to start monitoring and managing PostgreSQL or Advanced Server instances. The PEM server installer installs the PEM web interface. You can use the interface to review information about objects that reside on monitored servers, or to review statistical information gathered by the PEM server.

After installing and configuring PEM, you can use your browser to access the PEM web interface. Open your browser, and navigate to:

```
https://<ip_address_of_PEM_host>:8443/pem
```

Where *ip_address_of_PEM_host* specifies the IP address of the host of the PEM server. The `Postgres Enterprise Manager Web Login` window opens:

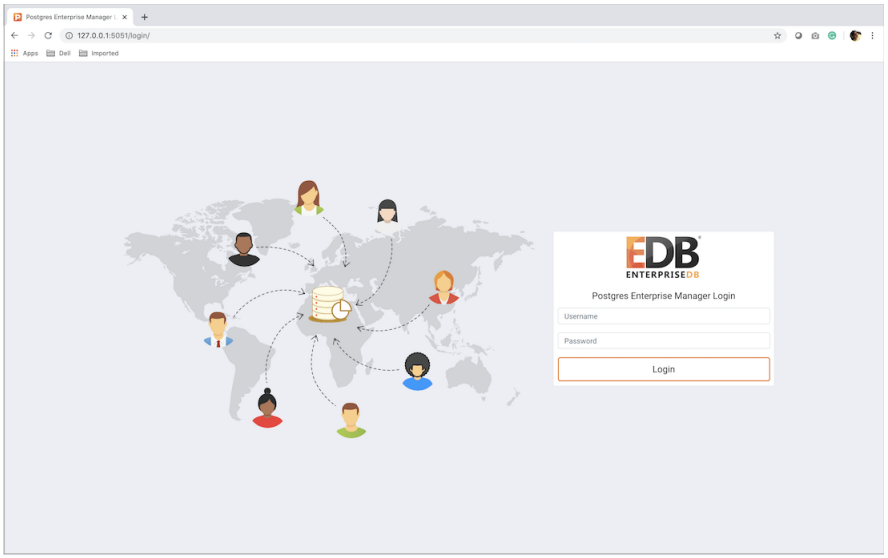

Fig. 1: *The PEM Web Login page*

Use the fields on the `Postgres Enterprise Manager Login` window to authenticate yourself with the PEM server:

- Provide the name of a `pem` database user in the `Username` field. For the first user connecting, this will be the name provided when installing the PEM server.
- Provide the password associated with the user in the `Password` field.

Click the `Login` button to connect to the PEM server.

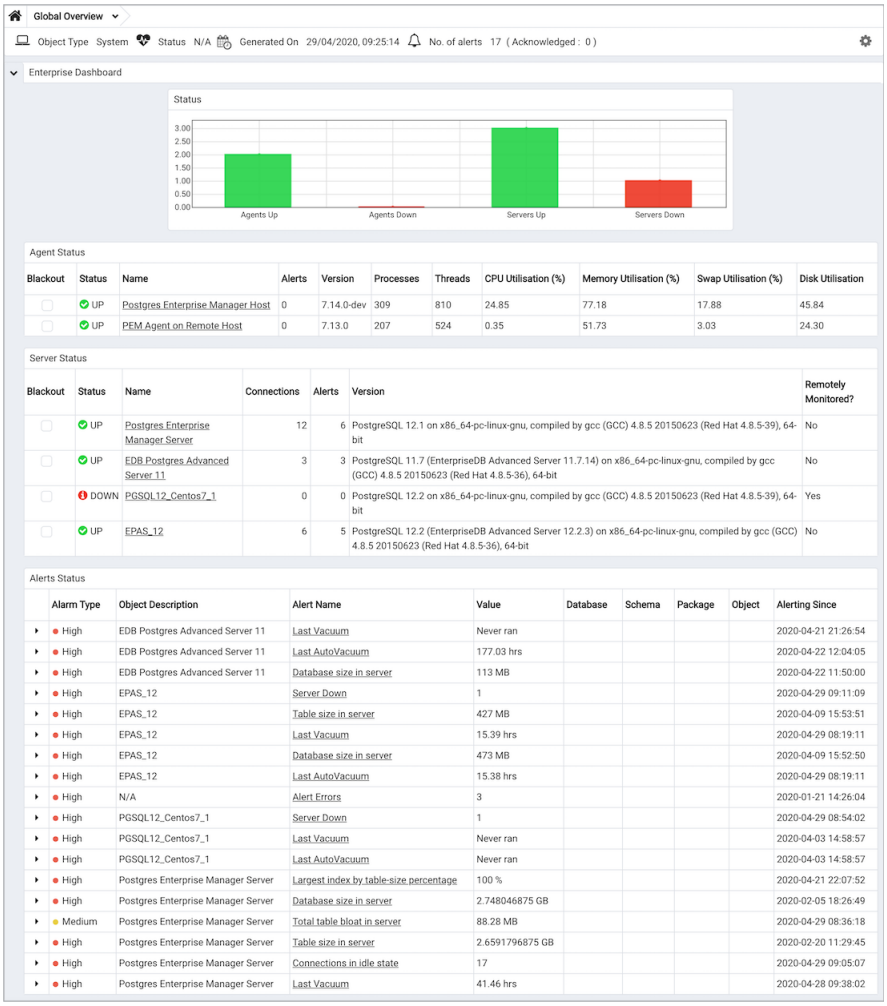

Fig. 2: *The Global Overview Dashboard*

Before you can use the PEM web interface to manage or monitor a database server, you must *register* the server with the PEM server. When you register a server, you describe the connection to the server, provide authentication information for the connection, and specify any management preferences (optionally binding an agent).

A server may be managed or unmanaged:

- A `managed` server is bound to a PEM agent. The PEM agent will monitor the server to which it is bound, and perform tasks or report statistics for display on the PEM dashboards. A managed server has access to extended PEM functionality such as Package Management or Custom Alerting; when registering a server, you can also allow a managed server to be restarted by PEM as required.
- An `unmanaged` server is not bound to a PEM agent; you can create database objects on an unmanaged server, but extended PEM functionality (such as Package Management or Custom Alerting) is not supported on an unmanaged server.

You must also ensure the `pg_hba.conf` file of the server that you are registering allows connections from the host of the PEM web interface.

To access online help information about the PEM web interface, select `Help` from the menu bar. Additional information is available in .pdf and .html format from the EnterpriseDB website

CHAPTER 5

Uninstalling Postgres Enterprise Manager™ Components

The process of uninstalling the PEM server or agent is platform-specific. The name of the package for PEM server is `edb-pem` and for PEM agent is `edb-pem-agent`.

If you uninstall the PEM server package from a host, the PEM agent package installed on the same host doesn't get uninstalled. But if you uninstall the PEM agent package, then the PEM server package installed on the same host also gets uninstalled.

5.1 Uninstalling PEM components from CentOS or RHEL hosts

You can use variations of the `rpm`, `yum remove`, or `yum erase` commands to remove the installed packages. Note that removing a package does not damage the PEM data directory.

- Include the `-e` option when invoking the rpm command to remove an installed package; the command syntax is:

  ```
  rpm -e <package_name>
  ```

- You can use the `yum remove` command to remove the pem server or agent package installed by yum. To remove a package, open a terminal window, assume superuser privileges, and enter the command:

  ```
  yum remove <package_name>
  ```

- You can use the `yum erase` command to remove the pem server or agent package along with the `edb-pem` and `edb-pem-docs` dependencies. To remove a package, open a terminal window, assume superuser privileges, and enter the command:

  ```
  yum erase <package_name>
  ```

Where *package_name* is the name of the package that you would like to remove.

5.2 Uninstalling PEM components from Debian or Ubuntu hosts

You can use `apt-get remove` or `apt-get purge` command to uninstall the PEM server or agent package from a Debian or Ubuntu host:

- To uninstall PEM server or agent from a Debian or Ubuntu host without impacting the configuration files and data directories, invoke the following command:

  ```
  apt-get remove <package_name>
  ```

- To uninstall PEM server or agent along with the configuration files and data directory, invoke the following command:

  ```
  apt-get purge <package_name>
  ```

Where *package_name* is the name of the package that you would like to remove.

5.3 Uninstalling PEM components from SLES hosts

To uninstall PEM server or agent from a SLES host, invoke the follwoing command:

```
zypper remove <package_name>
```

Where *package_name* is the name of the package that you would like to remove.

CHAPTER 6

Reference - Linux Service Script

The Postgres server on which the PEM server resides must contain a service script. Postgres installers generated by EnterpriseDB create a service script for you; if you are using a Postgres server from another source, you must provide a service script.

You can use the following example of a linux service script as a starting point when developing a script for a Postgres installation that was installed or built from a source that does not provide one. Please ensure (if you copy and paste from this example) that the line breaks are copied correctly.

```
| #!/bin/bash
| # chkconfig: 2345 85 15
| # description: Starts and stops the PostgreSQL/Postgres Plus Advanced
Server database server
| # PostgreSQL/Postgres Plus Advanced Server Service script template for
Linux
| # Please modify the values accordingly
| DB_DESC="Database Server - PostgreSQL 9.6"
| DB_INSTALL_DIR=/opt/PostgreSQL/9.6
| DB_BIN_DIR=${DB_INSTALL_DIR}/bin
| DB_LIB_DIR=${DB_INSTALL_DIR}/lib
| DB_DATA_DIR=${DB_INSTALL_DIR}/data
| DB_HBA_FILE=${DB_DATA_DIR}/pg_hba.conf
| DB_CONF_FILE=${DB_DATA_DIR}/postgresql.conf
| DB_PID_FILE=${DB_DATA_DIR}/postmaster.pid
| DB_STARTUP_LOG=${DB_DATA_DIR}/pg_log/startup.log
| DB_SERVICE_USER=postgres
| _die()
| {
|   echo ""
|   echo "FATAL ERROR: $*"
|   echo ""
|   exit 1
```

(continues on next page)

(continued from previous page)

```
| }
| if [ `id -u` != 0 ]; then
|       _die "You must run this script as the root."
| fi

| # Source function library.
| if [ -f /etc/rc.d/functions ];
| then
|       . /etc/init.d/functions
| fi
start()
| {
|   STARTDBSERVER=0
|   if [ -e "${DB_PID_FILE}" ]
|   then
|       PIDOFDB=`head -n 1 "${DB_PID_FILE}"`
|       PIDALIVEDB=""
|       if [ -n "${DB_PID_FILE}" ]; then
|            PIDALIVEDB=`ps -p "${PIDOFDB}" | grep
"${PIDOFDB}"`
|         fi
|         if [ -n "${PIDALIVEDB}" ]
|         then
|               echo "The '${DB_DESC}' is already running.
|                         PID(${PIDALIVEDB})."
|                                       exit
|                         else
|                                       STARTDBSERVER=1
|                         fi
|                          else
|                         STARTDBSERVER=1
|                             fi
| if [ "${STARTDBSERVER}" != 0 ]
| then
| echo "Starting ${DB_DESC}..."
| su - "${DB_SERVICE_USER}" -c
"LD_LIBRARY_PATH=\"${DB_LIB_DIR}:\$LD_LIBRARY_PATH\"
\"${DB_BIN_DIR}/pg_ctl\" -w start -D \"${DB_DATA_DIR}\" -l
\"${DB_STARTUP_LOG}\" -o \"${DB_STARTUP_OPTIONS}\""
| if [ $? -eq 0 ];
| then
| echo "${DB_DESC} started successfully."
| exit 0
| else
| echo "${DB_DESC} did not start in a timely fashion, please see
'${DB_STARTUP_LOG}' for details."
| exit 1
| fi
| fi
| }
| stop()
| {
```

(continues on next page)

(continued from previous page)

```
| if [ -e "${DB_PID_FILE}" ]
| then
| PIDOFDB=`head -n 1 "${DB_PID_FILE}"`
| PIDALIVEDB=""
| if [ -n "${DB_PID_FILE}" ]; then
| PIDALIVEDB=`ps -p "${PIDOFDB}" | grep "${PIDOFDB}"`
| fi
| if [ -n "${PIDALIVEDB}" ]
| then
| echo "Stopping ${DB_DESC}..."
| su - "${DB_SERVICE_USER}" -c
"LD_LIBRARY_PATH=\"${DB_LIB_DIR}:\$LD_LIBRARY_PATH\"
\"${DB_BIN_DIR}/pg_ctl\" stop -m fast -D \"${DB_DATA_DIR}\"
-l \"${DB_STARTUP_LOG}\" -o \"${DB_STARTUP_OPTIONS}\""
| else
| echo "The '${DB_DESC}' is not running."
| fi
| else
| echo "The '${DB_DESC}' is not running."
| fi
| }
| reload()
| {
| echo "Reloading '${DB_DESC}'..."
| su - "${DB_SERVICE_USER}" -c
"LD_LIBRARY_PATH=\"${DB_LIB_DIR}:\$LD_LIBRARY_PATH\"
\"${DB_BIN_DIR}/pg_ctl\" reload -D \"${DB_DATA_DIR}\" -l
\"${DB_STARTUP_LOG}\" -o \"${DB_STARTUP_OPTIONS}\""
| }
| restart()
| {
| echo "Restarting '${DB_DESC}'..."
| su - "${DB_SERVICE_USER}" -c
"LD_LIBRARY_PATH=\"${DB_LIB_DIR}:\$LD_LIBRARY_PATH\"
\"${DB_BIN_DIR}/pg_ctl\" restart -m fast -w -D
\"${DB_DATA_DIR}\" -l \"${DB_STARTUP_LOG}\" -o
\"${DB_STARTUP_OPTIONS}\""
| if [ $? -eq 0 ];
| then
| echo "'${DB_DESC}' restarted successfully."
| exit 0
| else
| echo "'${DB_DESC}' did not start in a timely fashion, please see
'${DB_STARTUP_LOG}' for details."
| exit 1
| fi
| }
| _die_incomplete_requirement()
| {
| echo "One or more required configuration variables are not set:"
| _die $*
| }
```

(continues on next page)

(continued from previous page)

```
| _validate_script()
| {
| if [ -z "${DB_INSTALL_DIR}" ]; then
| _die_incomplete_requirement "Missing installation directory";
| fi
| if [ ! -d "${DB_INSTALL_DIR}" ]; then
| _die_incomplete_requirement "The specified - '${DB_INSTALL_DIR}'
is not a valid installation directory. It is not present on the
system.";
| fi
| if [ -z "${DB_BIN_DIR}" ]; then
DB_BIN_DIR=${DB_INSTALL_DIR}/bin; fi
| if [ ! -d "${DB_BIN_DIR}" ]; then
| _die_incomplete_requirement "The specified - '${DB_BIN_DIR}' is
not a valid bin directory. It is not present on the system.";
| fi
| if [ ! -f "${DB_BIN_DIR}/pg_config" -o ! -f
"${DB_BIN_DIR}/pg_ctl" ]; then
| _die_incomplete_requirement "The specified - '${DB_BIN_DIR}' does
not contain the database server binaries.";
| fi
| if [ -z "${DB_LIB_DIR}" ]; then
DB_LIB_DIR=${DB_INSTALL_DIR}/lib; fi
| if [ -z "${DB_DESC}" ]; then DB_DESC=`${DB_BIN_DIR}/pg_config
--version`; fi
| if [ -z "${DB_DATA_DIR}" ]; then
| _die_incomplete_requirement "Missing data directory settings in the
script. Please set 'DB_DATA_DIR' variable in the script.";
| fi
| if [ ! -d "${DB_DATA_DIR}" ]; then
| _die_incomplete_requirement "The specified - '${DB_DATA_DIR}' is
not a valid. It is not present on the system.";
| fi
| if [ ! -f "${DB_DATA_DIR}/PG_VERSION" -o ! -d
"${DB_DATA_DIR}/base" -o ! -d "${DB_DATA_DIR}/global" ]; then
| _die_incomplete_requirement "The directory - '${DB_DATA_DIR}'
does not look like a valid PostgreSQL/Postgres Plus Advanced Server
data directory."
| fi
| if [ -z "${DB_SERVICE_USER}" ]; then
| _die_incomplete_requirement "The service-user is not specified in
the service script. Please set 'DB_SERVICE_USER' variable in the
script."
| fi
| DB_VALID_SERVICE_USER=`cat /etc/passwd | grep
"^${DB_SERVICE_USER}:"`
| if [ -z "${DB_VALID_SERVICE_USER}" ]; then
| _die_incomplete_requirement "The service-user
'${DB_SERVICE_USER}' is not present on the system. Please specify
the correct information."
| fi
| DB_DATA_DIR_OWNER=`ls -l ${DB_DATA_DIR}/PG_VERSION | awk
```

(continues on next page)

(continued from previous page)

```
'{print $3}'`
| if [ x"${DB_DATA_DIR_OWNER}" != x"${DB_SERVICE_USER}" ]; then
| _die_incomplete_requirement "The specified user -
'${DB_SERVICE_USER}' does not own the data directory -
'${DB_DATA_DIR}'. The data directory is owned by the user -
'${DB_DATA_DIR_OWNER}'."
| fi
| if [ -z "${DB_HBA_FILE}" ]; then
DB_HBA_FILE=${DB_DATA_DIR}/pg_hba.conf; fi
| if [ ! -f "${DB_HBA_FILE}" ]; then
| _die_incomplete_requirement "The hba-file - '${DB_HBA_FILE}' does
not exist."
| fi
| if [ -z "${DB_CONF_FILE}" ]; then
DB_CONF_FILE=${DB_DATA_DIR}/postgresql.conf; fi
| if [ ! -f "${DB_CONF_FILE}" ]; then
| _die_incomplete_requirement "The config-file - '${DB_CONF_FILE}'
does not exist."
| fi
| if [ -z "${DB_PID_FILE}" ]; then
DB_PID_FILE=${DB_DATA_DIR}/postmaster.pid; fi
| if [ -z "${DB_STARTUP_LOG}" ]; then
DB_STARTUP_LOG=${DB_DATA_DIR}/pg_log/startup.log; fi
| DB_STARTUP_OPTIONS=""
| if [ x"${DB_CONF_FILE}" != x"${DB_DATA_DIR}/postgresql.conf" ];
then
| DB_STARTUP_OPTIONS="-c 'config_file=${DB_CONF_FILE}'"
| fi
| if [ x"${DB_HBA_FILE}" != x"${DB_DATA_DIR}/pg_hba.conf" ]; then
| DB_STARTUP_OPTIONS="${DB_STARTUP_OPTIONS} -c
'hba_file=${DB_HBA_FILE}'"

fi

if [ x"${DB_PID_FILE}" != x"${DB_DATA_DIR}/postmaster.pid" ]; then

| DB_STARTUP_OPTIONS="${DB_STARTUP_OPTIONS} -c
'external_pid_file=${DB_PID_FILE}'"
| fi
| if [ x"${DEBUG_VALIDATION}" = x"1" ]; then
| echo "Using these values in the scripts:"
| echo ""
| echo "DB_DESC : ${DB_DESC}"
| echo ""
| echo "DB_INSTALL_DIR : ${DB_INSTALL_DIR}"
| echo "DB_BIN_DIR : ${DB_BIN_DIR}"
| echo "DB_LIB_DIR : ${DB_LIB_DIR}"
| echo ""
| echo "DB_DATA_DIR : ${DB_DATA_DIR}"
| echo "DB_HBA_FILE : ${DB_HBA_FILE}"
| echo "DB_CONF_FILE : ${DB_CONF_FILE}"
| echo "DB_PID_FILE : ${DB_PID_FILE}"
```

(continues on next page)

(continued from previous page)

```
| echo "DB_STARTUP_LOG : ${DB_STARTUP_LOG}"
| echo ""
| echo "DB_SERVICE_USER : ${DB_SERVICE_USER}"
| echo "DB_STARTUP_OPTIONS : ${DB_STARTUP_OPTIONS}"
| echo ""
| fi
| }
| DEBUG_VALIDATION=0
| # See how we were called.
| case "$1" in
| start)
| _validate_script
| start
| ;;
| stop)
| _validate_script
| stop
| ;;
| reload)
| _validate_script
| reload
| ;;
| restart)
| _validate_script
| restart
| ;;
| condrestart)
| _validate_script

if [ -e "${DB_PID_FILE}" ]
then
PIDOFDB=`head -n 1 "${DB_PID_FILE}"`
PIDALIVEDB=""
if [ -n "${DB_PID_FILE}" ]; then
PIDALIVEDB=`ps -p "${PIDOFDB}" | grep
"${PIDOFDB}"`
fi
if [ -n "${PIDALIVEDB}" ]
then
restart
else
echo "The '${DB_DESC}' is not running."
fi
else
echo "The '${DB_DESC}' is not running."
fi
;;
status)
_validate_script
su - "${DB_SERVICE_USER}" -c
"LD_LIBRARY_PATH=\"${DB_LIB_DIR}:\$LD_LIBRARY_PATH\"
\"${DB_BIN_DIR}/pg_ctl\" status -D \"${DB_DATA_DIR}\" -l
```

(continues on next page)

(continued from previous page)

```
\"${DB_STARTUP_LOG}\" -o \"${DB_STARTUP_OPTIONS}\""
| ;;
| validate)
| DEBUG_VALIDATION=1
| _validate_script
| exit 0
| ;;
| *)
| echo "Usage: $0
{start|stop|restart|condrestart|reload|status|validate}"
| exit 1
| esac
```

CHAPTER 7

Conclusion

EDB Postgres Enterprise Manager Installation Guide for Linux

EnterpriseDB® Corporation 34 Crosby Drive, Suite 201, Bedford, MA 01730, USA

T +1 781 357 3390 F +1 978 467 1307 E info@enterprisedb.com www.enterprisedb.com

- EDB designs, establishes coding best practices, reviews, and verifies input validation for the logon UI for EDB Postgres Enterprise Manager where present. EDB follows the same approach for additional input components, however the nature of the product may require that it accepts freeform SQL, WMI or other strings to be entered and submitted by trusted users for which limited validation is possible. In such cases it is not possible to prevent users from entering incorrect or otherwise dangerous inputs.
- EDB reserves the right to add features to products that accept freeform SQL, WMI or other potentially dangerous inputs from authenticated, trusted users in the future, but will ensure all such features are designed and tested to ensure they provide the minimum possible risk, and where possible, require superuser or equivalent privileges.
- EDB does not that warrant that we can or will anticipate all potential threats and therefore our process cannot fully guarantee that all potential vulnerabilities have been addressed or considered.

Index