



Postgres Enterprise Manager - Administration Guide

リリース *9.1.0*

EnterpriseDB Corporation

2023 年 03 月 17 日

目次

第 1 章	Changing the default port	3
1.1	RHEL および CentOS の場合	3
1.2	Debian および Ubuntu の場合	4
第 2 章	Registering a Postgres server	5
2.1	データベースサーバーの手動登録	5
2.2	自動サーバー検出	12
2.3	pemworker ユーティリティを使用してサーバーを登録する	13
2.3.1	pemworker ユーティリティを使用してサーバーを登録解除する	14
2.4	接続とバインディングの検証	15
2.5	AWS に Postgres クラスターを登録する	15
2.5.1	AWS EC2 で実行されている Postgres クラスターの監視	16
2.5.2	AWS RDS で実行されている Postgres クラスターのモニタリング	16
第 3 章	Registering a PEM agent	17
3.1	例	19
第 4 章	Register the PEM agent using command line. Assign the	21
第 5 章	SSL certificates and key files using the -o option.	23
第 6 章	Managing a PEM server	27
6.1	PEM サーバーとエージェントの起動と停止	27
6.2	Linux での PEM サーバーまたは PEM エージェントの制御	28
6.3	Windows での PEM サーバーまたは PEM エージェントの制御	28
6.4	HTTPD サーバーの制御	29
6.5	PEM 認証の管理	29
6.6	PEM サーバー構成の編集	30
6.7	セキュリティの管理	30
6.7.1	ログインの役割	30
6.7.2	グループの役割	31
6.7.3	PEM の定義済みロールを使用して PEM 機能へのアクセスを管理する	31
6.7.4	チームの役割を使用する	34
6.7.5	オブジェクトの権限	34
6.8	サーバー構成パラメーター - リファレンス	35
第 7 章	Managing Postgres servers	51
7.1	監視対象データベース サーバーの起動と停止	51
7.2	pg_hba.conf ファイルの変更	51
7.3	データベースとオブジェクトの作成と維持	53
7.4	Linux サービス スクリプトのテンプレート	54

第 8 章	Managing a PEM agent	55
8.1	ジョブ通知の管理	55
8.1.1	ジョブレベルでのジョブ通知の設定	55
8.1.2	エージェントレベルでのジョブ通知の設定	55
8.1.3	サーバーレベルでのジョブ通知の設定	56
8.2	PEM スケジュール済みジョブの管理	56
8.3	エージェント権限	59
8.4	エージェント構成	61
8.4.1	アラートスレッドについて	65
8.5	エージェントのプロパティ	65
第 9 章	Managing certificates	67
9.1	SSL 証明書の置き換え	67
9.2	エージェント SSL 証明書の更新	70
第 10 章	Managing configuration settings	73
第 11 章	Troubleshooting agent issues	75
11.1	削除された PEM エージェントの復元	75
11.2	コマンドラインを使用して、ダウンまたは不明なステータスの PEM エージェントを削除する	75

Postgres Enterprise Manager (PEM) へようこそ。PEM は、EDB Postgres Advanced Server または PostgreSQL データベースの管理および分析機能を提供するコンポーネントで構成されています。PEM は、オープンソースの pgAdmin 4 プロジェクトに基づいています。

PEM は、包括的なデータベース設計および管理システムです。PEM は、初心者と経験豊富な Postgres ユーザーの両方のニーズを満たすように設計されており、データベースオブジェクトの作成、保守、および使用を簡素化する強力なグラフィカルインターフェイスを提供します。

このガイドでは、次のことについて説明します。

- デフォルトポートの変更
- データベースサーバーの登録
- 登録エージェント
- PEM サーバーの管理
- PEM エージェントの管理
- 認証の管理
- 構成の管理
- トラブルシューティング エージェント

第1章 Changing the default port

デフォルトでは、PEM サーバーの構成時に 8443 ポートが Web サービスに割り当てられます。Apache 構成ファイルのいくつかのパラメーターを変更することで、構成後にポートを変更できます。Apache 構成ファイルの名前と場所は、プラットフォーム固有です。

1.1 RHEL および CentOS の場合

1. ファイル `/etc/httpd/conf.d/edb-ssl-pem.conf` を編集し、次のパラメーターで 8443 をポート番号に置き換えます。

```
Listen 8443
<VirtualHost _default_:8443>
ServerName localhost:8443
RewriteRule ^(.*)$ https://%{HTTP_HOST}:8443%{REQUEST_URI} [L,R=301]
```

1. `semanage` がインストールされていない場合は、インストールします。

```
sudo yum -y install policycoreutils-python
```

1. ポートがリストされているかどうかを確認します。

```
semanage port -l | grep http
```

1. ポートがリストに表示されない場合は、次のコマンドを実行します。

```
sudo semanage port -a -t http_port_t -p tcp <your_port_number>
```

1. `httpd` サービスを再起動します。

```
sudo systemctl restart httpd
```

ポートを使用して PEM Web インターフェイスにアクセスできるようになりました。詳細については、[Accessing the web interface](#) を参照してください。

1.2 Debian および Ubuntu の場合

1. ファイル `/etc/apache2/sites-available/edb-ssl-pem.conf` を編集し、次のパラメーターで 8443 をポート番号に置き換えます。

```
Listen 8443
<VirtualHost _default_:8443>
ServerName HTTPD-EDBPEM-SERVER-v8:8443
```

1. httpd サービスを再起動します。

```
sudo systemctl restart apache2
```

ポートを使用して PEM Web インターフェイスにアクセスできるようになりました。詳細については、[Accessing the web interface](#) を参照してください。

第2章 Registering a Postgres server

PEM でサーバーを管理または監視するには、サーバーを PEM に登録してエージェントをバインドする必要があります。サーバーをリモートエージェント (別のホストに存在するエージェント) にバインドできます。ただし、エージェントが同じホストに存在しない場合、インスタンスに関するすべての統計情報にアクセスすることはできません。

2.1 データベースサーバーの手動登録

PEM を使用してデータベースサーバーを管理または監視するには、次のことを行う必要があります:

- EDB Postgres Advanced Server または PostgreSQL サーバーを PEM サーバーに登録します。
- データベース サーバーを PEM エージェントにバインドします。

[サーバーの作成] ダイアログボックスを使用して、サーバーの登録情報を提供し、PEM エージェントをバインドし、サーバーを PEM クライアントツリーに表示できます。[サーバーの作成] ダイアログ ボックスを開くには、[オブジェクト]>[作成]>[サーバー] を選択します。

注釈: 接続を試みる前に、登録している Postgres サーバーの *pg_hba.conf* ファイルが PEM クライアントのホストからの接続を許可していることを確認する必要があります。

[全般] タブを使用して、サーバーの全般的なプロパティを記述します。

- **名前** フィールドを使用して、サーバーの名前を指定します。この名前は、PEM ブラウザ ツリーでサーバーを識別します。
- グループを使用して、ツリー内のサーバーとエージェントを整理できます。グループを使用すると、多数のサーバーをより簡単に管理できます。たとえば、本番グループ、テストグループ、または LAN 固有のグループを作成できます。 **Group** リストを使用して、新しいサーバーを表示するサーバーグループを選択します。
- **Team** フィールドを使用して、PostgreSQL ロール名を指定します。このロールのメンバーである PEM ユーザー、最初にサーバーを作成した PEM ユーザー、または PEM サーバーでスーパーユーザー権限を持つ PEM ユーザーのみが、PEM にログオンしたときにこのサーバーを表示します。このフィールドを空白のままにすると、デフォルトですべての PEM ユーザーにサーバーが表示されます。[サーバー構成] ダイアログボックスで `show_objects_with_no_team` パラメーターを使用して、動作を変更できます。`show_objects_with_no_team` が `false` に設定されている場合、チームのないサーバーは他のすべてのユーザーに表示されません。
- **背景** カラーセクターを使用して、サーバーに格納されているデータベースオブジェクトの背後にある PEM ツリーに表示する色を選択します。

- **Foreground** カラーセレクターを使用して、サーバーに格納されているオブジェクトの PEM ツリー内のラベルのフォントの色を選択します。
- [今すぐ接続しますか?] を選択して、[保存] を選択したときにサーバー接続を試みます。後で接続が試行されるまで、指定された接続パラメーターを PEM クライアントに検証させたくない場合は、**今すぐ接続しますか?** をオフにします。
- サーバーに関するメモを **コメント** フィールドに記入してください。

[接続] タブを使用して、サーバーの接続の詳細を指定します。

- サーバー ホストの IP アドレスまたは完全修飾ドメイン名を [**ホスト名/アドレス**] フィールドに指定します。Unix ベースのシステムでは、アドレス フィールドを空白のままにして、ローカルマシンのデフォルトの PostgreSQL Unix ドメインソケットを使用します。または、PostgreSQL ソケットを含む代替パスを設定できます。パスを入力する場合、パスはスラッシュ (/) で始まる必要があります。
- **Port** フィールドにホストのポート番号を指定します。
- **Maintenancedatabase** フィールドを使用して、インストールされている pgAgent スキーマと adminpack オブジェクトが含まれていると予想される PEM が接続する初期データベースの名前を指定します (両方ともオプション)。PostgreSQL バージョン 8.1 以降では、メンテナンス DB は通常 **postgres** と呼ばれます。以前のバージョンでは **template1** がよく使用されますが、テンプレートデータベースが乱雑になるのを避けるために **postgres** データベースを作成することをお勧めします。
- **Kerberos 認証** を **はい** に設定して、監視対象サーバーに Kerberos 認証を使用します。デフォルトでは、監視対象サーバーは PEM サーバーと同じ認証方法を使用します。監視対象サーバーが Kerberos 認証を使用したくない場合は、**config_local.py** ファイルで **ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS** を **TRUE** に設定します。
- サーバーで認証するときに使用する名前を **Username** フィールドに指定します。
- 指定したユーザーに関連付けられたパスワードを **Password** フィールドに入力します。
- **パスワードを保存しますか?** を選択して、パスワードを暗号化された形式で PEM バックエンドデータベースに保存し、後で再利用できるようにします。各パスワードは、ユーザーごと、サーバーごとに保存され、他のチームメンバーとは共有されません。PEM は保存されたパスワードを使用して、次回データベースサーバーに接続します。保存したパスワードを削除するには、まずデータベースサーバーを切断してから、[**オブジェクト**] > [**保存したパスワードをクリア**] を選択します。
- **Role** フィールドを使用して、クライアントがサーバーに接続した後に使用する権限が割り当てられている役割の名前を指定します。この値を使用すると、1つのロールとして接続し、接続が確立されたときに別のロール (このフィールドで指定したロール) の権限を引き受けることができます。接続ロールは、指定されたロールのメンバーである必要があります。

SSL タブを使用して SSL を構成します。

- **SSL モード** フィールドで、サーバーが使用する SSL 接続のタイプを選択します。SSL 暗号化の使用に関する詳細については、[the PostgreSQL documentation](#) を参照してください。

プラットフォーム固有のファイル マネージャー ダイアログボックスを使用して、SSL 暗号化をサポートするファイルをサーバーにアップロードできます。ファイルマネージャーを開くには、次の各フィールドの右側にあるアイコンを選択します。

- **クライアント証明書** フィールドを使用して、クライアント SSL 証明書を含むファイルを指定します。このファイルは、PEM がデスクトップモードでインストールされている場合はデフォルトの `~/.postgresql/postgresql.crt` ファイルを置き換え、PEM が Web モードでインストールされている場合は `<STORAGE_DIR>/<USERNAME>/.postgresql/postgresql.crt` ファイルを置き換えます。SSL 接続が確立されていない場合、このパラメーターは無視されます。
- **Clientcertificatekey** フィールドを使用して、クライアント証明書に使用される秘密鍵を含むファイルを指定します。このファイルは、PEM がデスクトップモードでインストールされている場合はデフォルトの `~/.postgresql/postgresql.key` を置き換え、PEM が Web モードでインストールされている場合は `<STORAGE_DIR>/<USERNAME>/.postgresql/postgresql.key` を置き換えます。SSL 接続が確立されていない場合、このパラメーターは無視されます。
- **ルート証明書** フィールドを使用して、SSL 認証局を含むファイルを指定します。このファイルは、デフォルトの `~/.postgresql/root.crt` ファイルを置き換えます。SSL 接続が確立されていない場合、このパラメーターは無視されます。
- **証明書失効リスト** フィールドを使用して、SSL 証明書失効リストを含むファイルを指定します。このリストは、`~/.postgresql/root.crl` にあるデフォルトのリストを置き換えます。SSL 接続が確立されていない場合、このパラメーターは無視されます。
- **SSL 圧縮?** が **True** に設定されている場合、SSL 接続を介して送信されるデータは圧縮されます。デフォルト値は **False** (圧縮は無効) です。SSL 接続が確立されていない場合、このパラメーターは無視されます。

警告: 証明書、秘密鍵、失効リストは、サーバー上のユーザーごとのファイル ストレージ領域に格納されます。これは、PEM サーバー プロセスが実行されるユーザー アカウントによって所有されます。これは、サーバーの管理者がそれらのファイルにアクセスできる可能性があることを意味します。この機能を使用する前に注意してください。

SSH トンネル タブを使用して、SSH トンネリングを構成します。トンネルを使用して、クライアントが直接接続できない可能性があるネットワーク上に存在するサーバーに、中間プロキシホストを介してデータベース サーバーを接続できます。

- **UseSSHtunneling** を **Yes** に設定して、指定したサーバーへの接続時に SSH トンネルを使用します。
- SSH ホスト (クライアント接続が転送される) の名前または IP アドレスを [**トンネルホスト**] フィールドに指定します。
- クライアント接続が転送される SSH ホストのポートを [**トンネルポート**] フィールドに指定します。
- **Username** フィールドに、SSH ホストへのログイン権限を持つユーザーの名前を指定します。
- **Authentication** フィールドで、SSH ホストに接続するときに使用する認証の種類を指定します。
- **パスワード** を選択して、SSH ホストへの認証にパスワードを使用します。これがデフォルトです。
- 接続時に秘密鍵ファイルを使用するには、**ID ファイル** を選択します。
- SSH ホストが認証用の秘密鍵ファイルを予期している場合は、**ID ファイル** フィールドを使用して鍵ファイルの場所を指定します。

- SSH ホストがパスワードを予期している場合は、パスワードフィールドを使用してパスワードを指定します。ID ファイルが使用されている場合は、パスフレーズを指定します。

[詳細] タブを使用して、サーバーの管理に使用する詳細を指定します。

- [**ホストアドレス**] フィールドにサーバー ホストの IP アドレスを指定します。
- **DB 制限** フィールドを使用して、`pg_database` テーブルに対して使用する SQL 制限を指定し、ツリーに表示されるデータベースを制限します。たとえば、`'live_db'`、`'test_db'` と入力すると、`live_db` および `test_db` データベースのみが表示されます。`pg_namespace` に対する制限を入力して、データベースの [プロパティ] ダイアログボックスからデータベースに表示されるスキーマを制限することもできます。
- **パスワードファイル** フィールドを使用して、パスワード ファイル (`.pgpass`) の場所を指定します。`.pgpass` ファイルを使用すると、ユーザーは接続時にパスワードを入力せずにログインできます。ファイルは PEM サーバーに存在する必要があります。詳細については、[Postgres documentation](#) を参照してください。
- **サービス ID** フィールドを使用して、データベース サービスプロセスにパラメータを指定します。Enterprise Manager ディレクトリに格納されているサーバーの場合は、サービス ID を入力します。Windows マシンでは、これは Windows サービスの識別子です。Linux マシンでは、サーバーの起動に使用される `init` スクリプトの名前は `/etc/init.d` で、サーバーの起動に使用される `systemd` スクリプトの名前は `systemctl` です。たとえば、EDB Postgres Advanced Server 11 サービスの名前は `edb-as-11` です。ローカルサーバーの場合、設定はオペレーティングシステムによって異なります。
- PEM クライアントが Windows マシンで実行されている場合、十分なアクセス権があれば `postmaster` サービスを制御できます。サービスの名前を入力します。リモートサーバーの場合は、先頭にマシン名 (`PSE1\pgsql-8.0` など) を追加します。PEM は、ローカルマシンで実行されているサービスを自動的に検出します。- PEM クライアントが Linux マシンで実行されている場合、十分なアクセス権があれば、ローカルマシンで実行されているプロセスを制御できます。`pg_ctl` プログラムにアクセスするためのフルパスと必要なオプションを指定します。サービス制御機能を実行するとき、PEM は `status/start/stop` キーワードをこれに追加します。例えば：

```
sudo /usr/pgsql-x/bin/pg_ctl -D /var/lib/pgsql/x/data
```

ここで、`x` は PostgreSQL データベース サーバーのバージョンです。

- サーバーが Failover Manager クラスターのメンバーである場合、PEM を使用してクラスターの状態を監視し、必要に応じてプライマリノードを交換できます。PEM が Failover Manager を監視できるようにするには、**EFM クラスター名** フィールドを使用してクラスター名を指定します。クラスター名は、Failover Manager クラスター プロパティファイルの名前のプレフィックスです。たとえば、クラスター プロパティファイルの名前が `efm.properties` の場合、クラスター名は `efm` です。
- PEM を使用して Failover Manager クラスターのステータスを監視している場合は、**EFM インストールパス** フィールドを使用して、Failover Manager バイナリファイルの場所を指定します。デフォルトでは、フェールオーバーマネージャーのバイナリ ファイルは `/usr/edb/efm-x.x/bin` にインストールされます。ここで、`x.x` はフェールオーバーマネージャーのバージョンを指定します。

PEMAgent タブを使用して、PEM エージェントの接続の詳細を指定します。

[接続パラメーター] タブで:

- **Remotemonitoring?** を **Yes** に設定して、PEM エージェントが監視対象サーバーと同じホストに存在しないことを示します。リモート監視が有効になっている場合、監視対象サーバーのエージェントレベルの統計をカスタム グラフやダッシュボードで使用できず、一部の PEM ユーティリティ (Audit Manager、Capacity Manager、Log Manager、Postgres Expert など) からリモートサーバーにアクセスできません。、およびチューニング ウィザード)。
- **バインドされたエージェント** リストを使用して、PEM エージェントを選択します。1 つのエージェントで複数の Postgres サーバーを監視できます。
- **[ホスト]** フィールドに、エージェントがデータベースサーバーに接続するときに使用する IP アドレスまたはソケットパスを入力します。デフォルトでは、エージェントは **General** タブに表示されるホスト アドレスを使用します。Unix サーバーでは、`/tmp` などのソケットパスを指定する必要がある場合があります。
- エージェントがサーバーに接続するときに使用するポート番号を入力します。デフォルトでは、エージェントは **Properties** タブで定義されたポートを使用します。
- **SSL** フィールドを使用して、SSL 操作モードを指定します。**require**、**prefer**、**allow**、**disable**、**verify-ca**、または **verify-full** を指定します。

Mode	Description
require	サーバーとエージェント間のトランザクションに SSL 暗号化を要求します。
prefer	SSL 暗号化が使用可能な場合は、サーバーとエージェントの間で SSL 暗号化を使用します。
allow	サーバーで必要な場合は、接続で SSL を使用できるようにします。
disable	エージェントとサーバー間の SSL 暗号化を無効にします。
verify-ca	SSL 暗号化を要求し、サーバーが認証局によって登録された証明書を使用して認証されることを要求します。
verify-full	SSL 暗号化を要求し、サーバーが信頼できる認証局によって登録された証明書を使用して認証されることを要求します。

SSL 暗号化の使用に関する詳細については、[PostgreSQL documentation](#) を参照してください。

- **Database** フィールドを使用して、エージェントが最初に接続する Postgres データベースの名前を指定します。EDB Postgres 分散データベースノードを登録している場合は、このフィールドに EDB Postgres 分散対応データベース名を指定します。
- **[ユーザー名]** フィールドで、エージェントがサーバーに接続するときに使用する役割の名前を指定します。すべての機能が期待どおりに動作するには、指定されたロールがデータベーススーパーユーザーである必要があります。指定されたロールがデータベーススーパーユーザーでない場合に機能しない機能のリストについては、

[エージェント権限](#) を参照してください。Kerberos

認証方式を使用している場合は、`pgd_monitor` または `pgd_superuser` ロールを持つユーザーをこのフィールドに指定します。

Postgres バージョン 10 以降を使用している場合は、`pg_monitor` ロールを使用して、スーパーユーザー以外に必要な権限を付与できます。`pg_monitor` ロールの詳細については、[Default Roles](#) を参照してください。

- **[ユーザー名]** フィールドに、サーバーへの接続時にエージェントが使用するユーザーの名前を指定します。すべての機能が期待どおりに動作するには、指定されたロールがデータベーススーパーユーザーである必要があります。Postgres バージョン 10 以降を使用している場合は、pg_monitor ロールを使用して、必要な権限を非スーパーユーザーに付与できます。pg_monitor ロールの詳細については、[Default Roles](#) を参照してください。
- サーバーへの接続時にエージェントが使用するパスワードを **パスワード** フィールドと **パスワードの確認** フィールドに指定します。パスワードを指定しない場合は、エージェントの認証を手動で構成する必要があります。たとえば、.pgpass ファイルを使用できます。このファイルは、エージェントがインストールされているシステムに存在し、アクセスできる必要があります。

[詳細] タブ:

- **Allowtakeover?** を **Yes** に設定して、別のエージェントがサーバーを引き継ぐことができるように指定します。この機能を使用すると、高可用性フェイルオーバープロセスの一環としてサーバーが別のホストに移動した場合などに、エージェントがデータベースサーバーの監視を担当できます。
- プラス記号 (+) を使用して、PEM 監視から除外するデータベースを追加します。 **PEMAgent** タブの **ConnectionParameters** タブに記載されているデータベースを除外することはできません。

注釈: データベース レベルのプロブは除外されたデータベースに対して実行されませんが、サーバー レベルのプロブはデータベース統計を収集できます。

BART タブの下の **General** タブを使用して、PEM サーバーにマップされる BART サーバーの一般的なプロパティを記述します。

- **BART サーバー** フィールドを使用して、BART サーバー名を選択します。PEM コンソールで構成されたすべての BART サーバーが一覧表示されます。
- **サーバー名** フィールドを使用して、BART サーバーを使用してバックアップするデータベースサーバーの名前を指定します。この名前は、BART 構成ファイルに保存されます。
- **バックアップ名** フィールドを使用して、データベースサーバーのバックアップに割り当てるユーザー定義名のテンプレートを指定します。バックアップ名テンプレートを指定しない場合は、BART によって割り当てられた整数のバックアップ識別子を使用してのみ、BART サブコマンドでバックアップを参照できます。
- **ホストアドレス** フィールドを使用して、バックアップ用に構成するデータベース サーバーの IP アドレスを指定します。
- **Port** フィールドを使用して、バックアップするデータベースに使用するポートを指定します。
- **User** フィールドを使用して、PEM コンソールから BART を使用してバックアップするデータベースのユーザーを指定します。このデータベースサーバーの増分バックアップを有効にする場合、ユーザーはスーパーユーザーである必要があります。
- **パスワード** フィールドを使用して、バックアップするデータベースのユーザーのパスワードを指定します。
- **ClusterOwner** フィールドを使用して、データベースクラスターを所有する Linux オペレーティングシステムのユーザーアカウントを指定します。これは通常、Oracle データベース互換モードでインストールされた EDB Postgres Advanced Server データベース クラスターの enterprisedb です。これは通

常、PostgreSQL データベース クラスターの場合は postgres であり、PostgreSQL データベース互換モードでインストールされた EDB Postgres Advanced Server データベース クラスターの場合です。

- **Archivecommand** フィールドを使用して、bart.cfg ファイルで使用するアーカイブ コマンド文字列の目的の形式を指定します。Archive コマンドに入力すると、データベース サーバーの Postgresql.conf ファイルが上書きされます。サーバーが追加されると、データベースサーバーが再起動するか、データベース構成が再ロードされます。
- **増分バックアップを許可しますか?** を使用して、このデータベースサーバーの増分バックアップを有効にするかどうかを指定します。
- **パスワードなしの SSH をセットアップしますか?** を使用して、データベース サーバーと BART サーバー間のパスワードなしのログインを許可する SSH 証明書を作成するかどうかを指定します。パスワードなしの SSH 認証を設定する前に、必ず PEM エージェントをバインドしてください。パスワードなしの SSH は、PEM エージェントによってリモートで監視されているデータベースサーバーでは機能しません。

BART タブの下の **Misc** タブを使用して、BART サーバーのその他のプロパティを記述します。

- **既定の構成を上書きしますか?** を使用して、BART サーバー構成を特定のデータベース サーバー構成で上書きします。
- **Xlog** メソッドを使用して、pg_basebackup の実行中にトランザクションログを収集する方法を指定します。
- **Retentionpolicy** フィールドを使用して、バックアップの保持ポリシーを指定します。これにより、アクティブなバックアップを古いものとしてマークし、削除の候補にする時期が決定されます。保持ポリシーは、バックアップの数または期間 (日、週、または月) で指定できます。
- **WAL 圧縮** を使用して、アーカイブされた Xlog/WAL ファイルを gzip 形式で圧縮するかどうかを指定します。WAL 圧縮を有効にするには、gzip 圧縮プログラムが BART ユーザー アカウントの PATH に存在する必要があります。増分バックアップを作成する必要があるデータベースサーバーでは、**WAL 圧縮** 設定を有効にしないでください。
- **CopyWALsduringrestore** フィールドを使用して、RESTORE 操作を呼び出すときにアーカイブされた WAL ファイルを収集する方法を指定します。**Enabled** に設定すると、データベース サーバーのアーカイブリカバリの前に、アーカイブされた WAL ファイルが BART バックアップカタログから <restore_path>/archived_wals ディレクトリにコピーされます。**無効** に設定すると、データベースサーバーのアーカイブリカバリ中に、アーカイブされた WAL ファイルを BART バックアップ カタログから直接取得できます。
- **スレッド数** フィールドを使用して、ブロックをコピーするスレッドの数を指定します。pg_basebackup ユーティリティを使用してバックアップを作成する場合は、**スレッドカウント** を **1** に設定する必要があります。
- **バッチサイズ** フィールドを使用して、変更されたブロックのコピーに使用されるメモリのブロック数を指定します。このオプションは、増分バックアップにのみ適用されます。
- **スキャン間隔** フィールドを使用して、新しい WAL ファイルをスキャンする前に WAL スキャナが待機する秒数を指定します。
- **MBM スキャンタイムアウト** フィールドを使用して、タイムアウトする前に MBM ファイルを待機する秒数を指定します。このオプションは、増分バックアップにのみ適用されます。

サーバーのプロパティを表示するには、PEM クライアントツリーでサーバー名を右クリックし、コンテキストメニューから [プロパティ] を選択します。サーバーのプロパティを変更するには、[プロパティ] ダイアログボックスを開く前にサーバーから切断します。

2.2 自動サーバー検出

監視するサーバーが監視エージェントと同じホストにある場合は、[自動検出] ダイアログボックスを使用して、登録とバインドのプロセスを簡略化できます。

特定のエージェントの自動検出を有効にするには、サーバー自動検出プローブを有効にする必要があります。これを行うには、PEM クライアント ツリーで PEM エージェントを選択し、**Management>ManageProbes** を選択します。**ManageProbes** タブが開いたら、**Enabled?** 列のスライダーが **Yes** に設定されていることを確認します。

[自動検出] ダイアログボックスを開くには、PEM クライアント ツリーで PEM エージェントを選択し、[管理]>[自動検出] を選択します。

[自動検出] ダイアログボックスが開くと、**検出されたデータベースサーバー** ボックスに、現在 PEM エージェントによって監視されていないサーバーのリストが表示されます。サーバー名の横にあるボックスを選択して、サーバーに関する情報を [サーバー接続の詳細] ボックスに表示し、エージェントの接続プロパティを [エージェント接続の詳細] ボックスに表示します。

CheckAll を使用して、表示されているすべてのサーバーの横にあるボックスを選択するか、**UncheckAll** を使用して、サーバー名の左側にあるすべてのボックスをオフにします。

ServerConnectionDetails ボックスのフィールドは、PEM が監視するサーバーに関する情報を提供します。

- **[名前]** フィールドで監視対象サーバーの名前を受け入れるか、変更します。指定した名前が PEM クライアントのツリーに表示されます。
- **サーバーグループ** リストを使用して、サーバーが PEM クライアントツリーに表示されるサーバーグループを選択します。
- **ホスト名/アドレス** フィールドを使用して、監視対象サーバーの IP アドレスを指定します。
- **Port** フィールドには、サーバーによって監視されているポートが表示されます。このフィールドは変更できません。
- **サービス ID** フィールドにサービスの名前を入力します。一部の PEM 機能を有効にするには、サービス名を指定する必要があります。
- デフォルトでは、**メンテナンスデータベース** フィールドは、選択したサーバーが Postgres メンテナンスデータベースを使用することを示します。インストールの **Maintenancedatabase** フィールドの内容をカスタマイズします。

AgentConnectionDetails ボックスのフィールドは、サーバーへの接続時に使用する PEM エージェントのプロパティを指定します。

- **Host** フィールドには、PEM エージェントのバインドに使用される IP アドレスが表示されます。
- **ユーザー名** フィールドには、選択したサーバーに接続するときに PEM エージェントが使用する名前が表示されます。

- パスワードフィールドには、指定したユーザー名に関連付けられたパスワードが表示されます。
- **SSL モード** フィールドを使用して、SSL 接続設定を指定します。

監視のためにバインドするサーバーの接続プロパティの指定が完了したら、[**OK**] を選択してサーバーを登録します。

OK を選択すると、新しく登録されたサーバーが PEM ツリーに表示され、PEM サーバーによって監視されます。

2.3 pemworker ユーティリティを使用してサーバーを登録する

pemworker ユーティリティを使用して、PEM サーバーによる監視用にサーバーを登録したり、データベースサーバーを登録解除したりできます。登録時に、pemworker ユーティリティは、登録コマンドを呼び出したシステムに常駐するエージェントに新しいサーバーをバインドします。サーバーを登録するには:

Linux ホストでは、次のコマンドを使用します。

```
pemworker --register-server
```

Windows ホストでは、次のコマンドを使用します。

```
pemworker.exe REGISTER-SERVICE
```

pemworker ユーティリティを呼び出すときに、コマンドラインオプションをコマンド文字列に追加します。各オプションの後に対応する値を指定します。

Option	Description
<code>--pem-user</code>	PEM 管理ユーザーの名前を指定します。必要。
<code>--server-addr</code>	サーバー ホストの IP アドレスまたは完全修飾ドメイン名を指定します。Unix ベースのシステムでは、アドレス フィールドを空白のままにして、ローカルマシンのデフォルトの PostgreSQL Unix ドメイン ソケットを使用できます。または、PostgreSQL ソケットを含む代替パスに設定することもできます。パスを入力する場合、パスはスラッシュ (/) で始まる必要があります。必要。
<code>--server-port</code>	ホストのポート番号を指定します。必要。
<code>--server-database</code>	サーバーが接続するデータベースの名前を指定します。必要。
<code>--server-user</code>	サーバーのモニター時にエージェントが使用するユーザーの名前を指定します。必要。
<code>--server-service-name</code>	登録されているサーバー上の操作 (STOP、START、RESTART など) を制御するデータベース サービスの名前を指定します。オプション。
<code>--remote-monitoring</code>	<code>--remote-monitoring</code> 句と <code>no</code> の値 (デフォルト) を含めて、サーバーが PEM エージェントと同じマシンにインストールされていることを示します。リモート監視が有効になっている場合 (<code>yes</code>)、監視対象サーバーのエージェント レベルの統計はカスタム グラフとダッシュボードで使用できず、一部の PEM ユーティリティ (Audit Manager、Capacity Manager、Log Manager、Postgres エキスパート、およびチューニングウィザード)。必要。

次のページに続く

表 2 – 前のページからの続き

Option	Description
<code>--efm-cluster-name</code>	サーバーを監視する Failover Manager クラスターの名前を指定します (該当する場合)。オプション。
<code>--efm-install-path</code>	該当する場合、Failover Manager のインストール ディレクトリへの完全なパスを指定します。オプション。
<code>--asb-host-name</code>	エージェントが接続しているホストの名前を指定します。
<code>--asb-host-port</code>	データベースへの接続時にエージェントが使用するポート番号を指定します。
<code>--asb-host-db</code>	エージェントが接続するデータベースの名前を指定します。
<code>--asb-host-user</code>	データベースでの認証時にエージェントが提供するデータベース ユーザー名を指定します。
<code>--asb-ssl-mode</code>	接続に使用する SSL 認証のタイプを指定します。サポートされている値には、 prefer 、 require 、 disable 、 verify-CA 、 verify-full が含まれます。
<code>--group</code>	サーバーが表示されるグループの名前を指定します。
<code>--team</code>	サーバーへのアクセスが許可されているグループ ロールの名前を指定します。
<code>--owner</code>	監視対象サーバーを所有する役割の名前を指定します。
<code>--display-name <name></code>	監視対象データベース サーバーの表示名を指定します。デフォルトはシステムのホスト名です。
<code>--config-dir</code>	エージェント構成ファイルのディレクトリ パスを指定します。デフォルトは <code>"<pemworker path>/../etc"</code> です。

環境変数 `PEM_SERVER_PASSWORD` を設定して PEM サーバーのパスワードを提供し、pemworker が PEM 管理者ユーザーとして接続できるようにします。

環境変数 `PEM_MONITORED_SERVER_PASSWORD` を設定して、pemagent によって登録および監視されているデータベースサーバーのパスワードを提供します。

パスワードを指定しないと、パスワード認証エラーが発生します。PEM サーバーは、サーバーが正しく登録されたことを確認します。

2.3.1 pemworker ユーティリティを使用してサーバーを登録解除する

pemworker ユーティリティを使用して、データベースサーバーを登録解除できます。サーバーを登録解除するには、pemworker ユーティリティを呼び出します。

Linux ホストでは、次のコマンドを使用します。

```
pemworker --unregister-server
```

Windows ホストでは、次のコマンドを使用します。

```
pemworker.exe UNREGISTER-SERVICE
```

pemworker ユーティリティを呼び出すときに、コマンド ラインオプションをコマンド文字列に追加します。各オプションの後に、対応する値を続けます。

Option	Description
<code>--pem-user</code>	PEM 管理ユーザーの名前を指定します。必要。
<code>--server-addr</code>	サーバー ホストの IP アドレスまたは完全修飾ドメイン名を指定します。Unix ベースのシステムでは、アドレス フィールドを空白のままにして、ローカル マシンのデフォルトの PostgreSQL Unix ドメイン ソケットを使用できます。または、PostgreSQL ソケットを含む代替パスに設定することもできます。パスを入力する場合、パスはスラッシュ (/) で始まる必要があります。必要。
<code>--server-port</code>	ホストのポート番号を指定します。必要。
<code>--config-dir</code>	エージェント構成ファイルのディレクトリ パスを指定します。デフォルトは <code>"<pemworker path>/../etc"</code> です。

PEM_SERVER_PASSWORD 環境変数を使用して PEM サーバーのパスワードを指定し、pemworker が PEM 管理ユーザーとして接続できるようにします。

パスワードを指定しないと、パスワード認証エラーが発生します。PEM サーバーは、サーバーが登録されていないことを確認します。

2.4 接続とバインディングの検証

登録されると、新しいサーバーが PEM ブラウザーツリーに追加され、グローバル概要ダッシュボードに表示されます。

新しくバインドされたサーバーに初めて接続すると、グローバル概要ダッシュボードのサーバーリストに新しいサーバーが「不明」のステータスで表示されることがあります。サーバーを認識する前に、バインドされたエージェントは多くのプローブを実行してサーバーを調べる必要があります。ネットワークの可用性によっては、これらのプローブが完了するまでに数分かかる場合があります。

数分後、グローバル概要ダッシュボードの棒グラフに、エージェントが正常に接続されたことが示されます。新しいサーバーは **PostgresServerStatus** リストに含まれています。

5 分経ってもグローバル概要ダッシュボードに新しいサーバーが一覧表示されない場合は、監視エージェントのログファイルでエラーを確認してください。ツリーでエージェントの名前を右クリックし、コンテキストメニューから **Dashboards>ProbeLogAnalysis** を選択します。

2.5 AWS に Postgres クラスターを登録する

PEM を使用して AWS ホスト上の Postgres クラスターを監視できるシナリオは 2 つあります。以下を監視できます。

- AWS EC2 で実行される Postgre クラスター
- AWS RDS で実行されている Postgre クラスター

2.5.1 AWS EC2 で実行されている Postgres クラスターの監視

AWS EC2 で Postgres クラスターを作成したら、PEM サーバーを使用してクラスターを登録および監視できます。現在、次のシナリオがサポートされています。

- 同じ AWS EC2 で実行されている Postgres クラスターと PEM エージェント、およびローカル マシンで実行されている PEM サーバー
- 同じローカル マシンで実行されている Postgres クラスターと PEM エージェント、および AWS EC2 で実行されている PEM サーバー
- 同じ AWS EC2 で実行されている Postgres クラスターと PEM エージェント、および別の AWS EC2 で実行されている PEM サーバー

注釈: 最初の 2 つのシナリオでは、AWS EC2 インスタンスが *pem* データベースにアクセスできるように、AWS EC2 で VPN を構成する必要があります。必要に応じて、ネットワーク管理者に連絡して VPN をセットアップしてください。

PEM エージェントは PEM サーバーとは異なるホスト上にあるため、最初に PEM エージェントを PEM サーバーに登録します。また、PEM エージェントを PEM サーバーに登録する際には、AWS EC2 インスタンスのホスト名を必ず使用してください。PEM エージェントの登録の詳細については、[Registering an agent](#) を参照してください。

次の操作を行った後、PEM を使用して Postgres クラスターを監視できます。

- PEM エージェントを PEM サーバーに登録します
- PEM サーバーへの追加中に、PEM エージェントを Postgres クラスターにバインドします

2.5.2 AWS RDS で実行されている Postgres クラスターのモニタリング

AWS RDS データベースの作成中に、エンジンオプションのプロンプトが表示されたら、**PostgreSQL** を選択します。次の手順については、[Remote monitoring](#) を参照してください。

第3章 Registering a PEM agent

別のホストにインストールされている各 PEM エージェントを PEM サーバーに登録する必要があります。(PEM エージェントは PEM サーバー・ホストとは異なります。)登録プロセスは、エージェントとの通信に必要な情報を PEM サーバーに提供します。Windows 用の PEM エージェントのグラフィカルインストーラは、エージェントの自己登録をサポートしています。Linux ホストでは、`pemworker` ユーティリティを使用してエージェントに登録する必要があります。

PEM エージェント パッケージ インストーラーは、PEM エージェントを `/usr/edb/pem/agent/bin` ディレクトリに配置します。エージェントに登録するには、`pemworker` ユーティリティを呼び出すときに、登録の詳細とともに `--register-agent` キーワードを含めます。

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

注釈:

- PEM サーバー ホストにインストールされた PEM エージェントは、デフォルトで PEM サーバーに登録されます。- PEM エージェントをアップグレードした後、再起動する必要があります。登録は必要ありません。

`pemworker` ユーティリティを呼び出すときに、コマンド ラインオプションをコマンド文字列に追加します。各オプションの後に対応する値を指定します。

Option	Description
<code>--pem-server</code>	PEM バックエンド データベース サーバーの IP アドレス。このパラメーターは必須です。
<code>--pem-port</code>	PEM バックエンド データベース サーバーのポート。デフォルト値は 5432 です。
<code>--pem-user</code>	PEM バックエンド データベース サーバーのスーパーユーザー権限を持つデータベース ユーザーの名前。このパラメーターは必須です。
<code>--pem-agent-user</code>	PEM サーバー バックエンド データベース サーバーに接続するためのエージェント ユーザー。
<code>--cert-path</code>	証明書が作成されるディレクトリへの完全なパス。パスを指定しない場合、証明書は Linux では <code>~/pem</code> に、Windows では <code>%APPDATA%/pem</code> に作成されます。
<code>--config-dir</code>	構成ファイルのディレクトリ パス。デフォルトは <code><pemworker path>/../etc`</code> です。

次のページに続く

表 1 – 前のページからの続き

Option	Description
<code>--display-name</code>	PEM ブラウザ ツリーに表示するエージェントのわかりやすい名前。デフォルトはシステムのホスト名です。
<code>--force-registration</code>	<code>force_registration</code> 句を含めて、提供された引数でエージェントを登録します。この句は、既存のエージェント構成をオーバーライドする場合に役立ちます。デフォルト値は <i>Yes</i> です。
<code>--group</code>	エージェントが表示されるグループの名前。
<code>--team</code>	監視対象のデータベース サーバーにアクセスできる PEM バックエンド データベース サーバー上のデータベース ロールの名前。
<code>--owner</code>	エージェントを所有する PEM バックエンド データベース サーバー上のデータベース ユーザーの名前。
<code>--allow_server_restart</code>	<code>allow-server_restart</code> パラメータを有効にして、PEM が監視対象サーバーを再起動できるようにします。デフォルト値は <i>True</i> です。
<code>--allow-batch-probes</code>	<code>allow-batch-probes</code> パラメータを有効にして、PEM がこのエージェントでバッチ プロブを実行できるようにします。デフォルト値は <i>False</i> です。
<code>--batch-script-user</code>	パッチ/シェル スクリプトの実行に使用するオペレーティングシステム ユーザー。デフォルト値はなしです。このパラメーターを空白のままにするか、指定したユーザーが存在しない場合、スクリプトは実行されません。
<code>--enable-heartbeat-connection</code>	<code>enable-heartbeat-connection</code> パラメータを有効にして、PEM エージェントとサーバー間に専用のハートビート接続を作成し、アクティブなステータスを更新します。デフォルト値は <i>False</i> です。
<code>--enable-smtp</code>	<code>enable-smtp parameter</code> を有効にして、PEM エージェントが PEM サーバーに代わって電子メールを送信できるようにします。デフォルト値は <i>False</i> です。
<code>--enable-snmp</code>	<code>enable-snmp parameter</code> を有効にして、PEM エージェントが PEM サーバーに代わって SNMP トラップを送信できるようにします。デフォルト値は <i>False</i> です。
<code>-o</code>	構成ファイルのオプションをオーバーライドするかどうかを指定します。

pemagent (Audit Manager、Log Manager、Tuning Wizard など) でデータベースサーバーの再起動が必要な PEM 機能を使用する場合は、`agent.cfg` ファイルで `allow_server_restart` の値を `true` に設定する必要があります。

注釈: PEM バージョン 7.11 以降がインストールされている PEM エージェントによって実行されるシェル/バッチ スクリプトを構成する場合は、`batch_script_user` パラメーターにユーザーを指定する必要があります。root 以外のユーザーを使用してスクリプトを実行することを強くお勧めします。root ユーザーを使用すると、データのセキュリティとオペレーティングシステムのセキュリティが損なわれる可能性があります。ただし、root ユーザーを使用して pemagent を元の設定に復元してスクリプトを実行する場合は、`batch_script_user` パラメータを `root` に設定する必要があります。

PEM データベースに変更が加えられる前に、接続エージェントは PEM データベースサーバーで認証されます。pemworker ユーティリティを呼び出すときは、PEM サーバーの管理ユーザーロール (`postgres`) に関連付けられているパスワードを指定する必要があります。管理パスワードは、次の 3 つの方法で指定できます。

- PEM_MONITORED_SERVER_PASSWORD 環境変数を設定します。
- コマンドラインで PGPASSWORD キーワードを使用してパスワードを指定します。
- .pgpass ファイルにエントリを作成します。

パスワードを指定しないと、パスワード認証エラーが発生します。不足しているその他の必須情報の入力を求めるプロンプトが表示されます。登録が完了すると、サーバーはエージェントが正常に登録されたことを確認します。

登録された各 PEM エージェントには、一意のエージェント ID が必要です。9.0 以降の場合:

- 値 `max(id)+1` は、各エージェント ID に割り当てられます。
- コマンドラインを使用して、エージェント ID を手動で上書きできます。

注釈: 8.6 以前では、シーケンス `pem.agent_id_seq` が生成され、PEM エージェントに一意の ID が与えられます。

3.1 例

この例は、バージョン 9.0 以降でデフォルト設定を上書きする PEM エージェントを登録する方法を示しています。-o オプションを使用します。

コマンドラインを使用して PEM エージェントを登録します。-o オプションを使用して agent_id 値 8 を割り当てます。

```
shell{promptUser: root}{promptHost: pem}{outputLines: 2-8} /usr/edb/pem/agent/
bin/pemworker --register-agent \ - -pem-server pemserver \ - -pem-user postgres
\ - -pem-port 5432 \ - -display-name agent8 \ - o agent_id=8 __OUTPUT__ Postgres
Enterprise Manager Agent registered successfully!
```

8 の agent_id が使用可能であるため、PEM エージェントは正常に登録されます。指定された ID が既存のエージェントによって既に使用されている場合、エラーがスローされます。

コマンドラインを使用して PEM エージェントを登録します。バージョン 9.0 以降で特定のエージェント ID に対して新しい SSL 証明書とキーファイルを生成しないようにするには、既存の SSL 証明書とキーファイルを割り当てます。SSL 証明書とキーファイルは、データベースユーザー agent<ID> に対して有効である必要があります。ここで、<ID> は、コマンドラインを使用して提供されたものと同じである必要があります。-o オプションを使用します。

```
```shell{promptUser: root}{promptHost: pem}{outputLines: 1,3-7,9-18} # List the location of valid SSL certifi-
cates and key files.
```

```
ls -l /root/.pem/agent5.* - rw—— - 1 root root 2192 Nov 7 11:27 /root/.pem/agent5.crt - rw—— - 1 root root 3244
Nov 7 11:27 /root/.pem/agent5.key
```



## **第4章   Register the PEM agent using command line. Assign the**



## 第5章 SSL certificates and key files using the -o option.

```
/usr/edb/pem/agent/bin/pemworker --register-agent
```

```
- -pem-server pemserver
- -pem-user postgres
- -pem-port 5432
- -config-dir /tmp/pem-config
- -display-name agent5
- o agent_id=5
- o agent_ssl_cert=/root/.pem/agent5.crt
- o agent_ssl_key=/root/.pem/agent5.key OUTPUT Postgres Enterprise Manager Agent registered successfully!
```

有効な SSL 証明書とキーファイルが適切なアクセス許可を持つ指定された場所で利用できるため、PEM エージェントは正常に登録されます。証明書またはキーファイルが有効でない場合、または適切なアクセス許可がない場合、エラーがスローされます。

pemworker ユーティリティを使用して、PEM エージェントを登録解除できます。エージェントを登録解除するには、pemworker ユーティリティを呼び出すときに、`--unregister-agent` キーワードと詳細を含めます。

```
pemworker --unregister-agent
```

pemworker ユーティリティを呼び出すときに、コマンド ラインオプションをコマンド文字列に追加します。各オプションの後に対応する値を指定します。

Option	Description
<code>--pem-user &lt;username&gt;</code>	PEM バックエンド データベース サーバーのデータベース ユーザー (pem_admin ロールのメンバー) の名前を指定します。このパラメーターは必須です。
<code>--config-dir</code>	構成ファイルのディレクトリ パスを指定します。デフォルトは <code>"&lt;pemworker path&gt;/../etc"</code> です。

PEM エージェント RPM インストーラーは、`/usr/edb/pem/agent/etc` ディレクトリに `agent.cfg.sample` という名前のサンプル構成ファイルを作成します。PEM エージェントを登録すると、pemworker プログラムは `agent.cfg` という名前の実際のエージェント構成ファイルを作成します。次の構成パラメーターを追加して、`agent.cfg` ファイルを変更する必要があります。

```
heartbeat_connection = true
```

ca-bundle.crt 認証局ファイルの場所も追加する必要があります。デフォルトでは、インストーラーは agent.cfg.sample ファイルで指定された場所に ca-bundle.crt ファイルを作成します。サンプルファイルから既定のパラメーター値をコピーするか、別の場所に保存されている ca-bundle.crt ファイルを使用する場合は、ca\_file パラメーターでその値を指定します。

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

次に、プラットフォーム固有のコマンドを使用して、PEM エージェントサービスを開始します。サービスの名前は pemagent です。

RHEL または CentOS 7.x または 8.x ホストで、systemctl を使用してサービスを開始します。

```
systemctl start pemagent
```

サービスは、エージェントを開始していることを確認します。エージェントが登録されて開始されると、グローバル概要ダッシュボードと PEM Web インターフェイスのオブジェクト ブラウザ ツリーに表示されます。

**pemworker ユーティリティを使用してサーバーを登録する方法については、**

**[データベースサーバーの手動登録](#) を参照してください。**

root 以外のユーザー アカウントを使用して PEM エージェントを登録するには、最初に PEM エージェントを root ユーザーとしてインストールする必要があります。インストール後、edb などの非ルート ユーザーの ID を想定します。それから：

1. edb としてログインします。pem および logs ディレクトリを作成し、読み取り、書き込み、および実行権限を割り当てます。

```
$ mkdir /home/edb/pem
$ mkdir /home/edb/pem/logs
$ chmod 700 /home/edb/pem
$ chmod 700 /home/edb/pem/logs
```

2. エージェントを PEM サーバーに登録します。

```
$ export PEM_SERVER_PASSWORD=edb

Use the following command to create agent certificates and an agent
configuration file (agent.cfg) in the `/home/edb/pem` directory.
$ /usr/edb/pem/agent/bin/pemworker --register-agent --pem-server <172.19.11.230> --
→pem-user postgres --pem-port 5432 --display-name non_root_pem_agent --cert-path /
→home/edb/pem --config-dir /home/edb/pem

Use the following command to assign read and write permissions to
these files:
$ chmod -R 600 /home/edb/pem/agent*
```

3. agent.cfg ファイルのパラメーターを変更します。

```
$ vi /home/edb/pem/agent.cfg
agent_ssl_key=/home/edb/pem/agent<id>.key
agent_ssl_cert=/home/edb/pem/agent<id>.crt
log_location=/home/edb/pem/worker.log
agent_log_location=/home/edb/pem/agent.log
```

<id> は、割り当てられた PEM エージェント ID です。

4. tmp ディレクトリを作成し、環境変数を設定して、エージェントを開始します。

```
$ mkdir /home/edb/pem/tmp

Create a script file, add the environment variable, give permissions, and execute:
$ vi /home/edb/pem/run_pemagent.sh
#!/bin/bash
export TEMP=/home/edb/agent/tmp
/usr/edb/pem/agent/bin/pemagent -c /home/edb/agent/agent.cfg
$ chmod a+x /home/edb/pem/run_pemagent.sh
$ cd /home/edb/pem
$./run_pemagent.sh
```

これで、PEM エージェントが登録され、edb ユーザーで開始されました。マシンが再起動しても、このエージェントは自動的に再起動しません。前のコマンドを使用して手動で開始する必要があります。

5. 必要に応じて、次のようにマシンの再起動時にこのエージェントを自動的に開始するルートユーザーとして、この PEM エージェントのサービスを作成できます。

を。構成ファイルパスの値と pemagent サービスファイル内のユーザーをスーパーユーザーとして更新します。

```
$ sudo vi /usr/lib/systemd/system/pemagent.service
[Service]
Type=forking
WorkingDirectory=/home/edb/pem
Environment=LD_LIBRARY_PATH=/usr/edb/pem/agent/lib:/usr/libexec/edb-snmpp++/lib
Environment=TEMP=/home/edb/pem/tmp
ExecStart=/usr/edb/pem/agent/bin/pemagent -c /home/edb/pem/agent.cfg
```

- b. 実行中のエージェント プロセスを停止し、エージェントサービスを再起動します。

```
Find the process id of the running pem agent and pem worker process and kill that
→process
$ ps -ax | grep pemagent
$ kill -9 <process_id_of_pemagent>
$ ps -ax | grep pemworker
$ kill -9 <process_id_of_pemworker>
Enable and start pemagent service
```

(次のページに続く)

(前のページからの続き)

```
$ sudo systemctl enable pemagent
$ sudo systemctl start pemagent
$ sudo systemctl status pemagent
```

6. PEM ダッシュボードでエージェントのステータスを確認します。

---

**注釈:** root 権限または別のユーザー (enterprisedb など) が所有するファイルへのアクセスを必要とするすべてのプローブとジョブは失敗します。

---

## 第6章 Managing a PEM server

PEM サーバーの管理に関連するタスクには、次のものがあります:

- PEM サーバーとエージェントの再起動
- PEM サーバーまたは PEM エージェントの制御
- Linux および Windows での HTTPD サービスの制御
- HTTPD サーバーの制御
- PEM 認証とセキュリティの管理
- `pg_hba.conf` ファイルの修正
- プロキシ サーバーを使用するように PEM を変更する

### 6.1 PEM サーバーとエージェントの起動と停止

PEM サーバーは、それが存在する Postgres サーバーインスタンスが起動、停止、または再起動すると、起動、停止、および再起動します。Postgres サーバーの制御に使用するのと同じコマンドを使用して、PEM サーバーを制御します。Linux プラットフォームでは、サービススクリプトを停止および開始するコマンドは、プラットフォームおよび OS のバージョンによって異なります。

PEM エージェントは、`pemagent` という名前のサービスによって制御されます。

Windows オペレーティングシステムには、サーバーの状態を表示し、対話型のサーバーコントロールを提供するグラフィカル サービスコントローラーが含まれています。サービス ユーティリティには、Windows コントロールパネルからアクセスできます。ユーティリティが開いたら、一覧表示されたサービスをナビゲートし、サービス名を選択します。

**停止、一時停止、開始、または再起動**を選択して、サービスの状態を制御します。

サービスを停止すると、Postgres サーバーに接続されているすべてのユーザーまたはクライアントアプリケーションが突然切断されます。サービスの制御の詳細については、EDB Postgres Advanced Server を参照してください。

## 6.2 Linux での PEM サーバーまたは PEM エージェントの制御

Linux プラットフォームでは、以下を制御するサービス スクリプトの名前。

- EDB Postgres Advanced Server 上の PEM サーバーは `edb-as-<x>` または `ppas-<x>` です。
- PostgreSQL 上の PEM サーバーは `postgresql-<x>` です。
- PEM エージェントは `pemagent` です。

`x` はサーバーのバージョン番号を示します。

サービス スクリプトを使用して、サービスを制御できます。

- RHEL または CentOS バージョン 7.x または 8.x でサービスを制御するには、コマンドプロンプトでスーパーユーザー権限を取得し、次のコマンドを発行します。

```
systemctl <service_name> <action>
```

どこ：

`service_name` はサービスの名前です。

`action` は、サービスによって実行されるアクションを指定します。特定：

- サービスを開始するための `start`。
- サービスを停止するための `stop`。
- サービスを停止してから開始する `restart`。
- サービスのステータスを確認する `status`。

## 6.3 Windows での PEM サーバーまたは PEM エージェントの制御

Windows オペレーティングシステムには、サーバーの状態を表示し、対話型のサーバーコントロールを提供するグラフィカル サービスコントローラーが含まれています。以下を制御するサービスの登録名:

- PostgreSQL 上の PEM サーバー ホストは `postgresql-<x>` です。
- Advanced Server の PEM サーバー ホストは `edb-as-<x>` または `ppas-<x>` です。
- PEM エージェントは Postgres Enterprise Manager - `pemAgent` です。

`x` はサーバーのバージョン番号を示します。

Windows コントロール パネルで、サービスユーティリティを開いて、サービスのリストを表示します。

サービスを停止するには、[ **サービスを停止する** ] を選択します。サービスを停止すると、サーバーに接続されているすべてのユーザーまたはクライアントアプリケーションが突然切断されます。

[ **サービスを一時停止** ] を選択して、サービスの構成パラメーターを再読み込みします。**サービスの一時停止** は、多くの構成パラメーターのユーザーセッションを中断せずにパラメーターをリセットする効果的な方法です。

[ サービスを開始する ] を選択して、サービスを開始します。

## 6.4 HTTPD サーバーの制御

Linux では、コマンド プロンプトで次のコマンドを入力して、PEM-HTTPD サービスのステータスを確認できます。

```
ps -ef | grep httpd
```

Linux が次のような応答を返す場合、httpd は実行されていません。

```
user 13321 13267 0 07:37 pts/1 00:00:00 grep httpd
```

CentOS または RHEL 7.x または 8.x システムでサービスを開始するには、次のコマンドを使用します。

```
systemctl start httpd
```

Windows では、サービス アプレットを使用して、PEM HTTPD サービスのステータスを確認できます。サービスアプレットを開いた後、リストから PEM HTTPD サービスを選択します。

[ステータス] 列には、サーバーの現在の状態が表示されます。サービスが実行されていない場合は、**Start** を選択して PEM HTTPD を開始します。

## 6.5 PEM 認証の管理

Postgres は多くの認証方法をサポートしています。

- 安全なパスワード (md5)
- GSSAPI
- SSPI
- ケルベロス
- ID
- LDAP
- ラディウス
- 証明書 (SSL)
- パム

Postgres および PEM 認証は、`pg_hba.conf` 構成ファイルによって制御されます。構成ファイルのエントリは次を指定します。

- 特定のデータベースに接続できるユーザー
- そのユーザーが接続を許可される前に必要な認証の種類

postgres という名前のユーザーが、安全なパスワード (md5) 認証接続を使用してローカル ホスト (127.0.0.1/32) からすべてのデータベースに接続できるようにする `pg_hba.conf` ファイルの一般的なエントリは、次の形式を取ります。

```
host all postgres 127.0.0.1/32 md5
```

システム構成によっては、PEM エージェントがサーバーへの接続に使用するユーザー アカウントのパスワードファイルを作成する必要がある場合もあります。このファイルにより、エージェントはサーバーの認証要求に適切に応答できます。1safepwd のパスワードを持つ、postgres という名前のユーザーのパスワードファイル内のエントリは、次の形式を取ります。

```
localhost:5432:*:postgres:1safepwd
```

パスワード ファイルの名前は通常、Linux システムでは `~root/.pgpass`、Windows では `%APPDATA%\postgresql\pgpass.conf` です。パスワードファイルの構成の詳細については、[PostgreSQL website](#) を参照してください。

Postgres でサポートされている認証方法の詳細については、[PostgreSQL core documentation](#) を参照してください。

## 6.6 PEM サーバー構成の編集

PEM クライアントを使用して PEM サーバーの構成パラメータを管理し、機能を有効にしたり、デフォルト設定を変更したりできます。[サーバー構成] ダイアログ ボックスを開くには、[管理]>[サーバー構成] を選択します。次に、必要に応じてパラメーター値を編集します。

## 6.7 セキュリティの管理

PEM は、Postgres の役割とサーバーを管理するためのインターフェイスを提供します。

### 6.7.1 ログインの役割

PEM サーバーに接続するときは、PEM サーバーがデータを保存するデータベースへのアクセスを許可するロール資格情報を提供する必要があります。デフォルトでは、サーバーへの最初の接続には postgres スーパーユーザーアカウントが使用されますが、(セキュリティと監査の両方の目的で) 接続ユーザーごとに個別のロールを作成することを強くお勧めします。PEM クエリ ツール、PEM Web インターフェイス **Create – Login/Group Role** ダイアログ、またはコマンド ライン クライアント (psql など) を使用してロールを作成できます。

[Create – Login/Group Role] ダイアログボックスを使用してロールを作成するには:

1. PEM ツリーで、役割を作成するサーバーのノードを展開します。
2. **Login/GroupRoles** ノードを右クリックします。
3. コンテキスト メニューから、[作成]>[ログイン/グループロール] を選択します。
4. [Create – Login/Group Role] ダイアログボックスのタブを使用して、役割を定義します。

5. 新しい役割の定義が完了したら、**保存** を選択して役割を作成します。

既存のログイン ロールのプロパティを変更するには、ツリーでログインロールの名前を右クリックし、コンテキスト メニューから [プロパティ] を選択します。ログインロールを削除するには、ロールの名前を右クリックし、コンテキストメニューから [削除/ドロップ] を選択します。

ロールの作成と管理の詳細については、[PostgreSQL online documentation](#) を参照してください。

## 6.7.2 グループの役割

グループロールはコンテナーとして機能し、データベースの作成などのシステム権限と、テーブルへのデータの挿入などのオブジェクト権限を分配するために使用されます。グループロールの主な目的は、DBA がシステムおよびオブジェクト権限の一括管理を容易にすることです。多数の異なるログインアカウントに個別に権限を割り当てたり変更したりするのではなく、1つのロールの権限を割り当てたり変更したりしてから、そのロールを一度に多くのログインロールに付与できます。

**GroupRoles** ノード (PEM ツリーの各登録済みサーバーの名前の下にあります) を使用して、グループの役割を作成および管理します。コンテキストメニューのオプションを使用すると、新しいロールを作成したり、既存のロールのプロパティを変更したりできるダイアログボックスにアクセスできます。[PostgreSQL documentation](#) でロールの作成に関する詳細情報を見つけることができます。

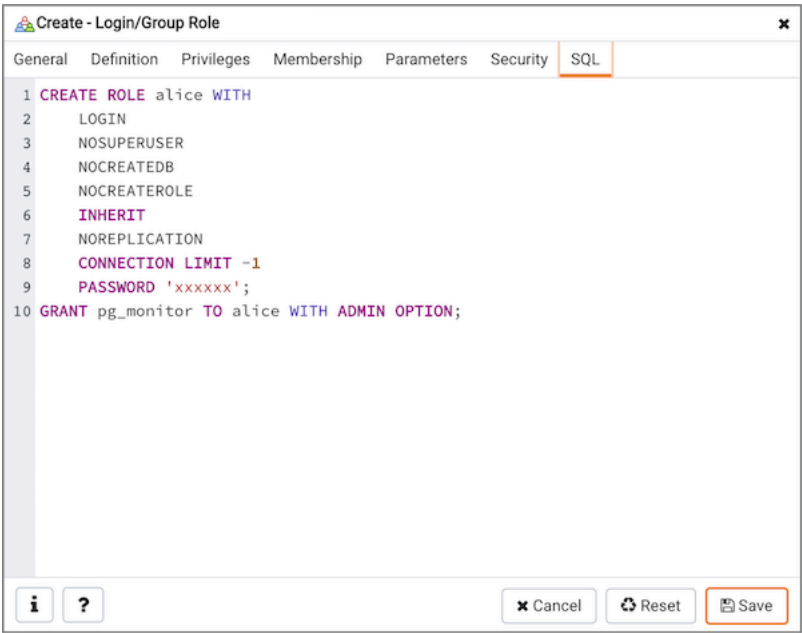
## 6.7.3 PEM の定義済みロールを使用して PEM 機能へのアクセスを管理する

[ログイン/グループ ロール] ダイアログボックスを使用して、権限が制限されたロールが監査マネージャー、キャパシティマネージャー、SQL プロファイラーなどの PEM 機能にアクセスできるようにすることができます。PEM の定義済みロールにより、PEM 機能へのアクセスが許可されます。これらのロールのメンバーシップが割り当てられたロールは、関連付けられた機能にアクセスできます。

ユーザーを定義するときは、**Membership** タブを使用して、新しいユーザーがメンバーであるロールを指定します。新しいユーザーは、メンバーである各ロールに関連付けられた権限を共有します。ユーザーが PEM 拡張機能にアクセスできるようにするには、役割が `pem_user` 役割のメンバーであり、機能へのアクセスを許可する定義済みの役割である必要があります。**Roles** フィールドを使用して、リストから定義済みの役割名を選択します。

**SQL** タブには、**Save** を選択したときにサーバーが実行する SQL コマンドが表示されます。

この例では、Audit Manager にアクセスできる `acctg_clerk` という名前のログインロールを作成します。このロールは、サーバーに対していつでも無制限に接続できます。



PEM の定義済みロールを使用して、表にリストされている機能へのアクセスを許可できます。

Value	Parent role	Description
pem_super_admin		PEM 上のすべてを管理/構成する役割
pem_admin	pem_super_admin	すべての可視エージェント/サーバーおよび監視対象オブジェクトの管理/管理/構成の役割
pem_user		pem_user ロールを持つユーザーに表示されるすべてのエージェント、サーバー、または監視対象オブジェクトへの読み取り専用アクセス権を持つロール。 pem_user ロールを持つユーザーは、サーバーのプロパティの [チーム] フィールドでこのロールが言及されているオブジェクトのみを表示できます。
pem_config	pem_admin	PEM の構成管理の役割
pem_component	pem_admin	すべてのウィザード/ダイアログ ボックス ベースのコンポーネントを実行/実行する役割
pem_rest_api	pem_admin	REST API にアクセスするロール

[次のページに続く](#)

表 1 – 前のページからの続き

Value	Parent role	Description
pem_server_service_ma	pem_admin	監視対象データベース サーバーの再起動/リロードを許可するロール (サーバー ID が指定されている場合)
pem_manage_schedule_	pem_admin	スケジュール タスクを構成する役割
pem_manage_chart	pem_admin	カスタム グラフを管理/構成する役割。
pem_manage_alert	pem_admin	アラートとそのテンプレートを管理/構成する役割
pem_config_alert	pem_config, pem_manage_alert	監視対象オブジェクトのアラートを構成する役割
pem_manage_probe	pem_admin	カスタム プローブを作成、更新、および削除し、カスタム プローブ構成を変更する役割
pem_config_probe	pem_config, pem_manage_probe	すべての表示可能な監視対象オブジェクトに対するプローブ構成 (履歴保持、実行頻度、プローブの有効化/無効化) の役割
pem_database_server_re	pem_admin	データベース サーバーを登録するロール
pem_comp_postgres_exj	pem_component	Postgres Expert を実行するロール
pem_comp_auto_discover	pem_component	データベース サーバー ダイアログ ボックスの自動検出を実行する役割
pem_comp_log_analysis	pem_component	Log Analysis Expert を実行する役割
pem_comp_sqlprofiler	pem_component	SQL プロファイラーを実行する役割
pem_manage_efm	pem_admin	Failover Manager を管理する役割
pem_comp_capacity_manage	pem_component	キャパシティ マネージャーを実行するロール
pem_comp_log_manage	pem_component	ログ マネージャーを実行する役割
pem_comp_audit_manage	pem_component	監査マネージャーを実行する役割
pem_comp_tuning_wizard	pem_component	チューニング ウィザードを実行するロール

次のページに続く

表 1 – 前のページからの続き

Value	Parent role	Description
pem_comp_bart	pem_component	BART サーバーを構成および管理する役割。
pem_comp_performance	pem_component	パフォーマンス診断を実行する役割。

#### 6.7.4 チームの役割を使用する

PEM による監視用にサーバーを登録すると、サーバーに関連付けるチームを指定できます。チームは、1つ以上の監視対象サーバーへのアクセスをロールメンバーの限られたグループに許可または制限するために使用できるグループロールです。PEM クライアントは、指定されたチームを持つサーバーを次のユーザーにのみ表示します:

- チームロールのメンバー
- サーバーを作成した役割
- PEM サーバーに対するスーパーユーザー権限を持つ役割

チームの役割を作成するには:

1. PEM ツリーで、役割を作成するサーバーのノードを展開します。
2. **Login/GroupRoles** ノードを右クリックします。
3. コンテキストメニューから、**[作成]>[ログイン/グループロール]**を選択します。
4. **[作成 - ログイン/グループロール]** ダイアログボックスで、フィールドを使用してチームロールのプロパティを指定します。

#### 6.7.5 オブジェクトの権限

データベースオブジェクトにアクセス、実行、または作成する前に、ロールに十分な権限を付与する必要があります。PEM を使用すると、PEM クライアント インターフェイスを使用して、グループロールまたはログインアカウントにオブジェクト権限を割り当て (**GRANT**) および削除 (**REVOKE**) できます。

オブジェクト権限は、特定のオブジェクトごとにグラフィカル オブジェクトエディタで管理されます。たとえば、データベーステーブルにアクセスする権限を割り当てるには、ツリーでテーブル名を右クリックし、コンテキストメニューから **[プロパティ]** を選択します。 **Privileges** タブを使用して、テーブルに特権を割り当てます。

PEM クライアントには、一度に多くのオブジェクト権限を管理できる Grant ウィザード (**Tools** メニューからアクセス) も含まれています。

## 6.8 サーバー構成パラメーター - リファレンス

グローバル設定オプションを使用して、PEM サーバーの動作の側面を変更できます。構成パラメーターのリストは変更される可能性があります。

Parameter name	Value and unit	Description
au-dit_log_retention_time	30 days	監査ログが PEM サーバーに保持される日数を指定します。
auto_create_agent_alerts	true	エージェントの登録時にデフォルトのエージェントレベルのアラートを自動的に作成するかどうかを指定します。
auto_create_server_alert	true	サーバーがエージェントにバインドされたときに、既定のサーバーレベルのアラートを自動的に作成するかどうかを指定します。
chart_disable_bullets	false	ダッシュボードおよびキャパシティ マネージャー レポートの折れ線グラフの黒丸を有効または無効にします。
cm_data_points_per_rep	50	Capacity Manager レポートのグラフにプロットするデータ ポイントの数を指定します。
cm_max_end_date_in_y	5 years	Capacity Manager がデータを推定する最大時間を指定します。レポートのしきい値ベースの終了日が無期限に推測されないようにします。
dash_alerts_timeout	60 seconds	アラート ダッシュボードのコンポーネントが更新されるまでの秒数を指定します。
dash_db_comrol_span	7 days	データベース分析およびサーバー分析ダッシュボードのコミット/ロールバック分析グラフにプロットするデータの日数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_db_comrol_timeou	1800 seconds	データベース分析ダッシュボードおよびサーバー分析ダッシュボードでコミット/ロールバック折れ線グラフが更新されるまでの秒数を指定します。
dash_db_connovervw_ti	300 seconds	データベース分析ダッシュボードで接続概要円グラフが更新されるまでの秒数を指定します。
dash_db_eventlag_span	7 days	データベース分析ダッシュボードの slony レプリケーションの Number of Events Lag グラフにプロットするデータの日数を指定します。
dash_db_eventlag_timec	1800 seconds	データベース分析ダッシュボードで、slony レプリケーションの Number of Events Lag 折れ線グラフが更新されるまでの秒数を指定します。
dash_db_hottable_rows	25 rows	データベース分析ダッシュボードの HOT テーブル分析テーブルに表示する行数を指定します。
dash_db_hottable_timeo	300 seconds	データベース分析ダッシュボードで Hot Tables テーブルが更新されるまでの秒数を指定します。
dash_db_io_span	7 days	データベース分析および I/O 分析ダッシュボードのデータベース I/O 分析グラフにプロットするデータの日数を指定します。
dash_db_io_timeout	1800 seconds	データベース分析および I/O 分析ダッシュボードでデータベース I/O 折れ線グラフが更新されるまでの秒数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_db_rowact_span	7 days	データベース分析、I/O 分析、およびサーバー分析ダッシュボードの行アクティビティ分析グラフにプロットするデータの日数を指定します。
dash_db_rowact_timeou	1800 seconds	データベース分析、I/O 分析、およびサーバー分析のダッシュボードで行アクティビティの折れ線グラフが更新されるまでの秒数を指定します。
dash_db_storage_timeou	300 seconds	データベース分析ダッシュボードでストレージの棒グラフが更新されるまでの秒数を指定します。
dash_db_timelag_span	7 days	データベース分析ダッシュボードの Slony レプリケーションのタイムラグチャートにプロットするデータの日数を指定します。
dash_db_timelag_timeou	1800 seconds	Slony レプリケーションの Time Lag 折れ線グラフがデータベース分析ダッシュボードで更新されるまでの秒数を指定します。
dash_db_useract_span	7 days	データベース分析ダッシュボードのユーザーアクティビティ分析グラフにプロットするデータの日数を指定します。
dash_db_useract_timeou	1800 seconds	データベース分析ダッシュボードでユーザーアクティビティの折れ線グラフが更新されるまでの秒数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_efm_timeout	300 seconds	フェールオーバー マネージャー ノード ステータスとフェールオーバー マネージャー クラスター情報の折れ線グラフがストリーミングレプリケーション ダッシュボードで更新されるまでの秒数を指定します。
dash_global_overview_t	30 seconds	グローバル概要ダッシュボードのコンポーネントが更新されるまでの秒数を指定します。
dash_header_timeout	60 seconds	すべてのダッシュボードのヘッダーの情報が更新されるまでの秒数を指定します。
dash_io_chkpt_span	7 days	I/O 分析ダッシュボードのチェックポイント グラフにプロットするデータの日数を指定します。
dash_io_chkpt_timeout	1800 seconds	I/O 分析ダッシュボードでチェックポイント折れ線グラフが更新されるまでの秒数を指定します。
dash_io_hotindx_timeou	300 seconds	I/O 分析ダッシュボードでホット インデックスの棒グラフが更新されるまでの秒数を指定します。
dash_io_hottbl_timeout	300 seconds	I/O 分析ダッシュボードでホット テーブルの棒グラフが更新されるまでの秒数を指定します。
dash_io_index_objectio_	25 rows	I/O 分析およびオブジェクト アクティビティ分析ダッシュボードのインデックス アクティビティ テーブルに表示される行数を指定します。
dash_io_index_objectio_	60 seconds	I/O 分析ダッシュボードおよびオブジェクト アクティビティ分析ダッシュボードでインデックス アクティビティ テーブルが更新されるまでの秒数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_io_objectio_rows	25 rows	I/O 分析およびオブジェクトアクティビティ分析ダッシュボードのオブジェクト I/O 詳細テーブルに表示される行数を指定します。
dash_io_objectio_timeout	300 seconds	I/O 分析およびオブジェクトアクティビティ分析ダッシュボードでオブジェクト I/O の詳細テーブルが更新されるまでの秒数を指定します。
dash_memory_hostmem	7 days	メモリ分析ダッシュボードのホスト メモリ アクティビティ分析グラフにプロットするデータの日数を指定します。
dash_memory_hostmem	1800 seconds	メモリ分析ダッシュボードでホスト メモリ アクティビティの折れ線グラフが更新されるまでの秒数を指定します。
dash_memory_hostmem	300 seconds	メモリ分析およびサーバー分析ダッシュボードでホスト メモリ構成の円グラフが更新されるまでの秒数を指定します。
dash_memory_servmem	7 days	メモリ分析ダッシュボードのサーバー メモリ アクティビティ分析グラフにプロットするデータの日数を指定します。
dash_memory_servmem	1800 seconds	メモリ分析ダッシュボードでサーバー メモリ アクティビティの折れ線グラフが更新されるまでの秒数を指定します。
dash_memory_servmem	300 seconds	メモリ分析ダッシュボードでサーバー メモリ構成の円グラフが更新されるまでの秒数を指定します。
dash_objectact_objstorage	15 rows	Object Activity Analysis ダッシュボードの Object Storage テーブルに表示する行数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_objectact_objstorage	300 seconds	Object Activity Analysis ダッシュボードで Object Storage テーブルがリフレッシュされるまでの秒数を指定します。
dash_objectact_objtopin	300 seconds	オブジェクト アクティビティ分析ダッシュボードで上位 5 つの最大インデックス棒グラフが更新されるまでの秒数を指定します。
dash_objectact_objtopal	300 seconds	オブジェクト アクティビティ分析ダッシュボードで上位 5 つの最大テーブルの棒グラフが更新されるまでの秒数を指定します。
dash_os_cpu_span	7 days	オペレーティング システム分析ダッシュボードの CPU グラフにプロットするデータの日数を指定します。
dash_os_cpu_timeout	1800 seconds	オペレーティング システム分析ダッシュボードで CPU 折れ線グラフが更新されるまでの秒数を指定します。
dash_os_data_span	7 days	オペレーティング システム分析ダッシュボードの I/O 折れ線グラフにプロットするデータの日数を指定します。
dash_os_disk_span	7 days	オペレーティング システム分析ダッシュボードの使用率グラフにプロットするデータの日数を指定します。
dash_os_hostfs_timeout	1800 seconds	オペレーティング システム分析ダッシュボードでホスト ファイル システムの詳細テーブルが更新されるまでの秒数を指定します。
dash_os_io_timeout	1800 seconds	オペレーティング システム分析ダッシュボードで I/O 折れ線グラフが更新されるまでの秒数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_os_memory_span	7 days	オペレーティング システム分析ダッシュボードのメモリ グラフにプロットするデータの日数を指定します。
dash_os_memory_timeo	1800 seconds	オペレーティング システム分析ダッシュボードでメモリの折れ線グラフが更新されるまでの秒数を指定します。
dash_os_packet_span	7 days	オペレーティング システム分析ダッシュボードのパケット グラフにプロットするデータの日数を指定します。
dash_os_packet_timeout	1800 seconds	オペレーティング システム分析ダッシュボードでネットワーク パケットの折れ線グラフが更新されるまでの秒数を指定します。
dash_os_process_span	7 days	オペレーティング システム分析ダッシュボードのプロセス グラフにプロットするデータの日数を指定します。
dash_os_process_timeou	1800 seconds	オペレーティング システム分析ダッシュボードでプロセスの折れ線グラフが更新されるまでの秒数を指定します。
dash_os_storage_timeou	1800 seconds	オペレーティング システム分析ダッシュボードでストレージ円グラフが更新されるまでの秒数を指定します。
dash_os_traffic_span	7 days	オペレーティング システム分析ダッシュボードのトラフィック グラフにプロットするデータの日数を指定します。
dash_os_traffic_timeout	1800 seconds	オペレーティング システム分析ダッシュボードでトラフィックの折れ線グラフが更新されるまでの秒数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_os_util_timeout	1800 seconds	オペレーティング システム 分析ダッシュボードで使用率折れ線グラフが更新されるまでの秒数を指定します。
dash_probe_log_timeout	300 seconds	プローブ ログ テーブルがリフレッシュされるまでの秒数を指定します。
dash_replication_archive	7 days	ストリーミングレプリケーション分析ダッシュボードの WAL アーカイブ ステータス グラフにプロットするデータの日数を指定します。
dash_replication_archive	1800 seconds	ストリーミングレプリケーション ダッシュボードで WAL アーカイブ ステータスの折れ線グラフが更新されるまでの秒数を指定します。
dash_replication_pagela	7 days	ストリーミングレプリケーション ダッシュボードの WAL ラグ ページ グラフにプロットするデータの日数を指定します。
dash_replication_pagela	1800 seconds	ストリーミングレプリケーション ダッシュボードで WAL ラグ ページの折れ線グラフが更新されるまでの秒数を指定します。
dash_replication_segme	7 days	ストリーミングレプリケーション ダッシュボードの WAL ラグ セグメント グラフにプロットするデータの日数を指定します。
dash_replication_segme	1800 seconds	ストリーミングレプリケーション ダッシュボードで WAL ラグ セグメントの折れ線グラフが更新されるまでの秒数を指定します。
dash_replication_timela	7 days	ストリーミングレプリケーション ダッシュボードのレプリケーション ラグ タイム グラフにプロットするデータの日数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_replication_timelag	1800 seconds	ストリーミングレプリケーション ダッシュボードでレプリケーション ラグ タイムの折れ線グラフが更新されるまでの秒数を指定します。
dash_server_buffers_writes	168 hours	サーバー分析ダッシュボードのバックグラウンドライター統計グラフにプロットするデータの日数を指定します。
dash_server_buffers_writes	300 seconds	サーバー分析ダッシュボードでバックグラウンドライター統計の折れ線グラフが更新されるまでの秒数を指定します。
dash_server_connoverview	300 seconds	サーバー分析ダッシュボードで接続概要円グラフが更新されるまでの秒数を指定します。
dash_server_database_time	300 seconds	サーバー分析ダッシュボードでデータベース テーブルが更新されるまでの秒数を指定します。
dash_server_dbsize_space	7 days	サーバー分析ダッシュボードのデータベース サイズ分析にプロットするデータの日数を指定します。
dash_server_dbsize_time	1800 seconds	サーバー分析ダッシュボードでデータベース サイズの折れ線グラフが更新されるまでの秒数を指定します。
dash_server_disk_timeouts	1800 seconds	サーバー分析ダッシュボードでディスクの折れ線グラフが更新されるまでの秒数を指定します。
dash_server_global_space	7 days	サーバー分析ダッシュボードのディスク ライン チャートにプロットするデータの日数を指定します。
dash_server_sharedbuffer	7 days	サーバー分析ダッシュボードの共有バッファ グラフにプロットするデータの日数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_server_sharedbuff_	1800 seconds	サーバー分析ダッシュボードで共有バッファの折れ線グラフが更新されるまでの秒数を指定します。
dash_server_tabspacesiz	7 days	サーバー分析ダッシュボードのテーブルスペース サイズ グラフにプロットするデータの日数を指定します。
dash_server_tabspacesiz	1800 seconds	サーバー分析ダッシュボードでテーブルスペース サイズの折れ線グラフが更新されるまでの秒数を指定します。
dash_server_useract_spa	7 days	サーバー分析ダッシュボードのユーザー アクティビティ グラフにプロットするデータの日数を指定します。
dash_server_useract_tim	1800 seconds	サーバー分析ダッシュボードでユーザー アクティビティの折れ線グラフが更新されるまでの秒数を指定します。
dash_sessact_lockact_tir	300 seconds	セッション アクティビティ分析ダッシュボードでセッション ロック アクティビティ テーブルが更新されるまでの秒数を指定します。
dash_sessact_workload_	300 seconds	セッション アクティビティ分析ダッシュボードでセッション ワークロード テーブルが更新されるまでの秒数を指定します。
dash_sess_waits_timewa	300 seconds	[Session Waits Analysis] ダッシュボードで [Session Waits By Time Waited] 円グラフが更新されるまでの秒数を指定します。
dash_sess_waits_waitdtl	300 seconds	セッション待機分析ダッシュボードでセッション待機の詳細テーブルが更新されるまでの秒数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
dash_storage_dbdtls_tim	300 seconds	ストレージ分析ダッシュボードでデータベースの詳細テーブルが更新されるまでの秒数を指定します。
dash_storage_dbovervw	300 seconds	ストレージ分析ダッシュボードでデータベース概要円グラフが更新されるまでの秒数を指定します。
dash_storage_hostdtls_ti	300 seconds	ホストの詳細テーブルが更新されるまでの秒数を指定します。
dash_storage_hostoverv	300 seconds	ストレージ分析ダッシュボードでホスト概要の円グラフが更新されるまでの秒数を指定します。
dash_storage_tblspcdtls	300 seconds	Storage Analysis ダッシュボードで Tablespace Details テーブルが更新されるまでの秒数を指定します。
dash_storage_tblspcover	300 seconds	ストレージ分析ダッシュボードで表領域概要の円グラフが更新されるまでの秒数を指定します。
dash_sys_waits_nowaits	300 seconds	システム待機分析ダッシュボードの待機回数別システム待機円グラフが更新されるまでの秒数を指定します。
dash_sys_waits_timewai	300 seconds	System Waits Analysis ダッシュボードで System Waits By Time Waited 円グラフが更新されるまでの秒数を指定します。
dash_sys_waits_waitdtl	300 seconds	システム待機分析ダッシュボードでシステム待機の詳細テーブルが更新されるまでの秒数を指定します。
deleted_charts_retention	7 days	カスタム グラフ (ユーザー定義のダッシュボードに表示される) を保存する日数を指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
deleted_probes_retention	7 days	カスタム プローブ (ユーザー定義のダッシュボードに表示される) が保存される日数を指定します。
download_chart_format	jpeg	ダウンロードしたチャートを保存する形式を指定します。 jpeg または png にすることができます。
flapping_detection_state_change	3	特定のアラートをフラッピングとして定義するために、指定された間隔内で検出された状態変化の数を指定します。 $N$ 以上の状態変化が $[N + 1 * (\min(\text{probe\_interval}) * 2)]$ 分にわたって発生し、細かい状態が None でない場合、フラッピングが開始されます。 $N$ のデフォルト値は 2 または 3 で、 $\min(\text{probe\_interval})$ はアラートで使用されるすべてのプローブの最小間隔です。フラッピングは、 $[2N * \min(\text{probe\_interval})]$ 分にわたって ZERO 状態の変化が発生したときに終了します。
job_retention_time	30 days	非定期的なスケジュールされたタスクとそれに関連するジョブが保持される日数を指定します。
long_running_transaction_timeout	5 minutes	長時間実行されていると見なされる前にクエリを実行する分数を指定します。
notification_command_file_name	<file_name>	パッシブサービス チェックの結果が送信される nagios コマンド ファイルを指定します。
nagios_enabled	t	アラート通知を nagios に送信するかどうかを指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
na- gios_medium_alert_as_c	f	中レベルの PEM アラートが nagios でクリティカルと見なされるかどうかを指定します。
na- gios_spool_retention_tin	7 days	Nagios メッセージが破棄されるまでスプール テーブルに保持する日数を指定します。
probe_log_retention_tim	30 days	プローブ ログ レコードが保持される日数を指定します。
re- minder_notification_inte	24 hours	アラートがクリアされなかった場合にリマインダーメールが送信されるまでの時間数を指定します。
server_log_retention_tin	30 days	サーバーログが PEM サーバーに保持される日数を指定します。
show_data_tab_on_grap	false	<i>true</i> の場合、 <b>Data</b> タブが各グラフに追加されます。 <b>[データ]</b> タブを選択して、グラフにプロットされたデータを確認します。
smtp_authentication	false	SMTP 経由の認証を有効または無効にするかどうかを指定します。
smtp_enabled	true	メールの送信を有効または無効にするかどうかを指定します。
smtp_encryption	false	暗号化された接続を使用して SMTP 電子メールを送信するかどうかを指定します。
smtp_password		SMTP サーバーへの接続に使用するパスワードを指定します。
smtp_port	25	電子メールの送信に使用する SMTP サーバー ポートを指定します。
smtp_server	127.0.0.1	電子メールの送信に使用する SMTP サーバーのホストアドレスを指定します。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
smtp_spool_retention_ti	7 days	送信された電子メール メッセージが破棄されるまでスプール テーブルに保持される日数を指定します。
smtp_username		SMTP サーバーへの接続に使用するユーザー名を指定します。
snmp_community	public	トラップを送信するときに使用する SNMP コミュニティを指定します。SNMPv1 および SNMPv2 でのみ使用されます。
snmp_enabled	true	SNMP トラップの送信を有効または無効にするかどうかを指定します。
snmp_port	162	SNMP トラップの送信に使用する SNMP サーバー ポートを指定します。
snmp_server	127.0.0.1	SNMP トラップの送信に使用する SNMP サーバー ホストアドレスを指定します。
snmp_spool_retention_ti	7 days	送信されたトラップが破棄されるまでスプール テーブルに保持される日数を指定します。
snmp_security_name		SNMP トラップを送信するためのユーザー名またはセキュリティ名を指定します。SNMPv3 でのみ使用されます。
snmp_security_engine_i		SNMP サーバー上の SNMP エージェントのエンジン ID を指定します。SNMPv3 でのみ使用されます。
snmp_security_level	NOAUTH_NOPRIV	セキュリティ レベルを指定します。可能な値は次のとおりです。AUTH_NOPRIV - 認証、プライバシーなし、または AUTH_PRIV - 認証、プライバシー、または NOAUTH_NOPRIV - 認証なし、プライバシーなし。SNMPv3 でのみ使用されます。

次のページに続く

表 2 – 前のページからの続き

Parameter name	Value and unit	Description
snmp_context_name		SNMP トラップを送信するときの MIB オブジェクトの識別子であるコンテキスト名を指定します。SNMPv3 でのみ使用されます。
snmp_context_engine_id		SNMP トラップを送信するときの MIB オブジェクトの識別子であるコンテキスト エンジン ID を指定します。指定しない場合、snmp_security_engine_id が使用されます。SNMPv3 でのみ使用されます。
snmp_authentication_protocol	NONE	SNMP トラップの認証タイプを指定します。可能な値は、NONE、HMACMD5、および HMACSHA です。SNMPv3 でのみ使用されます。
snmp_privacy_protocol	NONE	SNMP トラップのプライバシー プロトコルを指定します。可能な値は、NONE、DES、AES128、IDEA、AES192、または AES256 です。SNMPv3 でのみ使用されます。
snmp_authentication_password		snmp_security_name に記載されているセキュリティ名に関連付けられた認証パスワードを指定します。SNMPv3 にのみ使用されます。
snmp_privacy_password		snmp_security_name に記載されているセキュリティ名に関連付けられたプライバシー パスワードを指定します。SNMPv3 にのみ使用されます。
webclient_help_pg	EDB hosted documentation	オンラインの PostgreSQL コア ドキュメントの場所を指定します。



## 第7章 Managing Postgres servers

### 7.1 監視対象データベース サーバーの起動と停止

PEM を使用すると、PEM クライアントを使用して管理対象サーバー・インスタンスを起動および停止できます。PEM がサービスを管理できるようにサーバーを構成するには、サーバー登録ダイアログボックスに入力します。ダイアログボックスを使用して、データベースサーバーと同じホストにインストールされているローカル エージェントである PEM エージェントにデータベースサーバーを登録します。あなたもすることができます：

- [プロパティ] ダイアログ ボックスで **PEM サーバーに保存オプション**を指定します。
- **Advanced** タブの **ServiceID** フィールドにサービススクリプトの名前を指定します。
- EDB Postgres Advanced Server の場合、サービス名は `edb-as-<x>` または `ppas-<x>` です。- PostgreSQL の場合、サービス名は `postgresql-<x>` です。

<x> はサーバーのバージョン番号を示します。

サーバーに接続した後、ツリーでサーバー名を選択し、[ツール]>[キューサーバーの起動] または [\*\*ツール]>[キューサーバーのシャットダウン]\*\* を選択して、サーバーを起動または停止できます。

### 7.2 pg\_hba.conf ファイルの変更

`pg_hba.conf` ファイルのエントリは、ネットワークの認証と承認を制御します。PEM サーバーホスト上の `pg_hba.conf` ファイルは、PEM サーバーと PEM-HTTPD、PEM エージェント、および監視対象サーバー間の接続を許可する必要があります。

PEM サーバーのインストールプロセスでは、PEM が監視するホストの IP アドレスと接続情報の入力を求められます。この情報は、PEM バッキング データベースの `pg_hba.conf` ファイルの先頭に追加されます。

PEM サーバーと他のコンポーネント間の接続を許可するために、`pg_hba.conf` ファイルを手動で変更する必要がある場合もあります。たとえば、PEM-HTTPD インストールが PEM サーバーと同じホスト上にない場合、PEM サーバーホストの `pg_hba.conf` ファイルを変更して、PEM-HTTPD がサーバーに接続できるようにする必要があります。

デフォルトでは、`pg_hba.conf` ファイルは、Postgres インストールの下のデータ ディレクトリにあります。たとえば、EDB Postgres Advanced Server 10 ホストでは、`pg_hba.conf` のデフォルトの場所は次のとおりです。

```
/var/lib/edb/as10/data/pg_hba.conf
```

選択したエディターで `pg_hba.conf` ファイルを変更できます。ファイルを変更したら、サーバーを再起動して変更を有効にします。

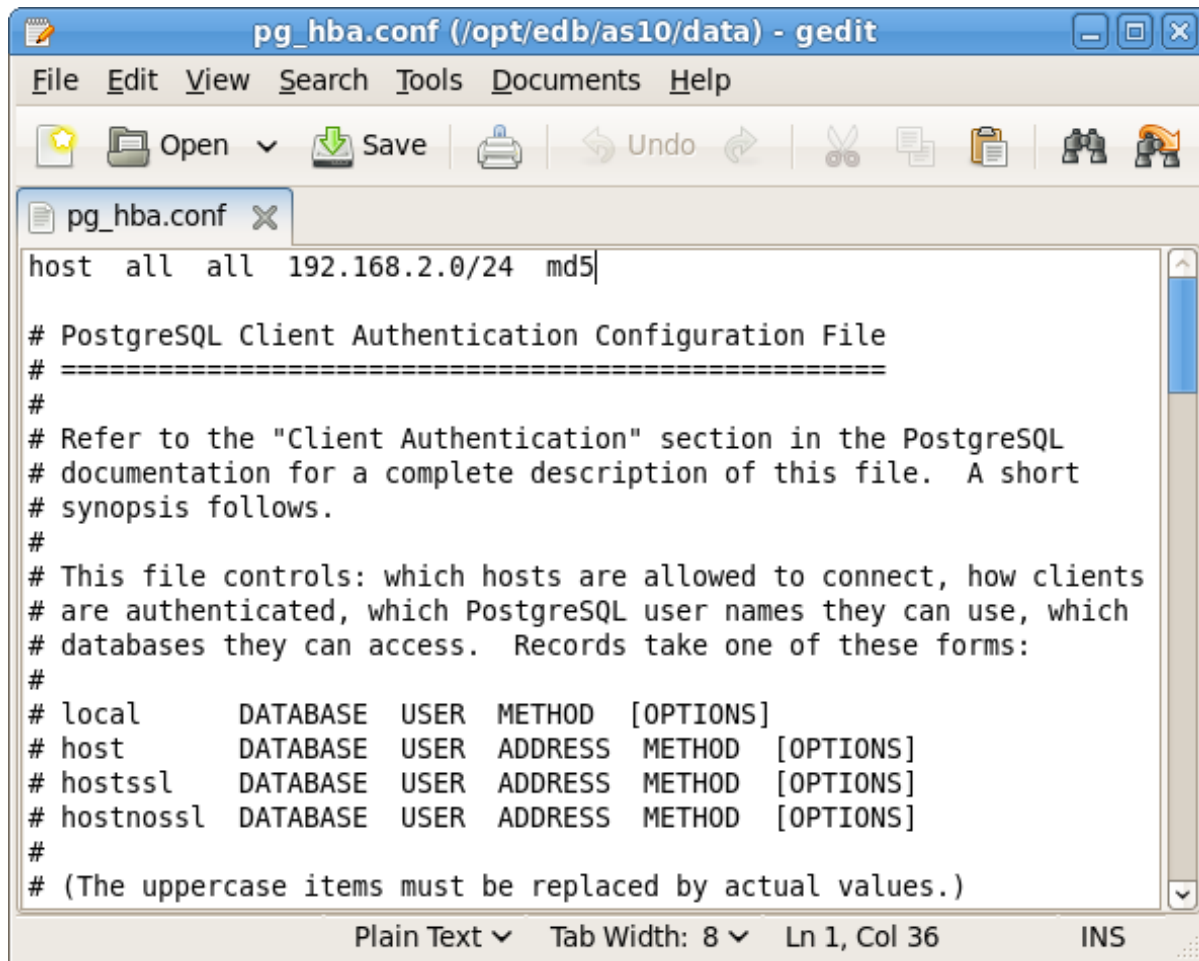


図 1: PEM entries in the pg\_hba.conf file

次の例は、postgres という名前のユーザーから、pg\_hba.conf が存在するホスト上の postgres データベースへの md5 パスワード認証接続を許可する pg\_hba.conf エントリを示しています。接続は 192.168.10.102 の IP アドレスから来ています。

#	TYPE	DATABASE	USER	CIDR-ADDRESS	METHOD
# IPv4 local connections:					
host		postgres	postgres	192.168.10.102/32	md5

ネットワーク ホストのアドレスまたはネットワークアドレス範囲を指定できます。たとえば、アドレスが 192.168.10.23、192.168.10.76、および 192.168.10.184 のサーバーからの接続を許可する場合は、192.168.10.0/24 の CIDR-ADDRESS を入力して、そのネットワーク内のすべてのホストからの接続を許可します。

#	TYPE	DATABASE	USER	CIDR-ADDRESS	METHOD
# IPv4 local connections:					
host		postgres	all	192.168.10.0/24	md5

pg\_hba.conf ファイルエントリのフォーマットの詳細については、[PostgreSQL core documentation](#) を参照してください。

PEM を使用して Postgres サーバーに接続する前に、両方のサーバーの pg\_hba.conf ファイルで接続が許可されていることを確認する必要があります。データベースサーバーへの接続時にエラーが発生した場合は、pg\_hba.conf ファイルを変更して、接続を許可するエントリを追加します。

## 7.3 データベースとオブジェクトの作成と維持

Postgres サーバーの各インスタンスは、1 つ以上のデータベースを管理します。各ユーザーは、データベースに含まれる情報にアクセスする前に、データベースに接続するための認証情報を提供する必要があります。PEM クライアントを使用すると、データベースと、テーブル、インデックス、ストアドプロシージャなど、データベースを構成するオブジェクトを作成および管理できます。

PEM でデータベースを作成するには、任意の管理対象サーバーの **Databases** ノードを右クリックし、**Create>Database** を選択します。データベースを定義したら、新しいデータベースにオブジェクトを作成できます。

たとえば、テーブルを作成するには、**Tables** ノードを右クリックし、**Create>Table** を選択します。[新しいテーブル] ダイアログボックスで、テーブルの属性を指定します。

PEM は、次のような他のデータベースオブジェクトを作成および管理するための同様のダイアログボックスを提供します。

- テーブル
- インデックス

- ストアド プロシージャ
- 機能
- トリガー
- ビュー
- 制約

各オブジェクト タイプがツリーに一覧表示されます。オブジェクトタイプに対応するノードを右クリックして、**作成**メニューにアクセスし、オブジェクトを作成します。選択したオブジェクトの管理タスクを実行するには、名前付きノードのコンテキストメニューから [プロパティ] を選択します。

## 7.4 Linux サービス スクリプトのテンプレート

データベース サーバーのサービス スクリプトにより、PEM サーバーはデータベースサーバーを起動、停止、または再起動できます。これは、構成管理、証明書管理、およびその他の管理タスクを実行するときに必要な場合があります。サービススクリプトはプラットフォーム固有です。

PEM サーバーが常駐する Postgres サーバーには、サービススクリプトが含まれている必要があります。EDB によって生成された Windows の Postgres インストーラーは、サービス スクリプトを作成します。ネイティブパッケージなどの別のソースから Postgres サーバーを使用している場合は、サービススクリプトを提供する必要があります。

---

**注釈:** CentOS または RHEL 7.x の場合 | Rocky Linux または RHEL 8.x の場合、サービス スクリプトは `/usr/lib/systemd/system` にあります。

---

Postgres サービスのカスタマイズについては、[PostgreSQL documentation](#) を参照してください。

## 第8章 Managing a PEM agent

### 8.1 ジョブ通知の管理

PEM コンソールでは、スケジュールされたタスクまたはカスタムエージェントジョブの下にリストされているシステム生成ジョブの成功または失敗時に SMTP トラップを送信するための設定を構成できます。これらの電子メール通知設定を次の3つのレベルで構成して、指定したユーザーグループに電子メール通知を送信できます。これらのレベルは、優先順位に従って表示されます。

- 職務レベル
- エージェントレベル
- PEM サーバー レベル (デフォルト)

#### 8.1.1 ジョブレベルでのジョブ通知の設定

次のいずれかの方法で、カスタム エージェント ジョブに対してのみ、ジョブレベルで電子メール通知設定を構成できます。

- 新しいエージェント ジョブの場合、ジョブの作成中に [エージェントジョブの作成] ウィザードの [通知] タブで電子メール通知設定を構成できます。
- 既存のカスタムジョブの場合、ジョブのプロパティを編集し、通知設定を構成できます。

**通知** タブを使用して、ジョブ レベルで電子メール通知設定を構成します。

- **通知を送信** フィールドを使用して、電子メール通知をいつ送信するかを指定します。
- **メールグループ** フィールドを使用して、メール通知を送信するメールグループを指定します。

#### 8.1.2 エージェントレベルでのジョブ通知の設定

ツリー ビューでエージェントを選択し、右クリックして [プロパティ] を選択します。[プロパティ] ダイアログボックスで、[ジョブ通知] タブを選択します。

[ジョブ通知] タブを使用して、エージェントレベルで電子メール通知設定を構成します。

- **デフォルト設定を上書きしますか?** スイッチを使用して、エージェントレベルのジョブ通知設定でデフォルトのジョブ通知設定を上書きするかどうかを指定します。[はい] を選択して、このダイアログボックスの残りの設定を有効にして、いつ、誰にジョブ通知を送信するかを定義します。

- ジョブ完了時に電子メールを送信しますか？ スイッチを使用して、ジョブが正常に完了したときにジョブ通知を送信するかどうかを指定します。
- ジョブの失敗時に電子メールを送信しますか？ スイッチを使用して、ジョブが失敗したときにジョブ通知を送信するかどうかを指定します。
- 電子メールグループフィールドを使用して、ジョブ通知を送信する電子メールグループを指定します。

### 8.1.3 サーバーレベルでのジョブ通知の設定

サーバー構成ダイアログボックスを使用して、PEM サーバーレベルで電子メール通知構成に関する情報を提供できます。[サーバー構成] ダイアログ ボックスを開くには、[管理]>[サーバー構成] を選択します。

4 つのサーバー構成パラメーターは、PEM サーバーレベルでのジョブ通知設定に関する情報を指定します。

- **job\_failure\_notification** スイッチを使用して、ジョブが失敗するたびに電子メール通知を送信するかどうかを指定します。
- **job\_notification\_email\_group** パラメータを使用して、電子メール通知の送信先の電子メールグループを指定します。
- **job\_retention\_time** パラメーターを使用して、システム内で定期的でないスケジュールされたタスクを保持する日数を指定します。
- **job\_status\_change\_notification** スイッチを使用して、ジョブの状態が変化するたびに電子メール通知を送信するかどうか (つまり、失敗、成功、または中断) を指定します。

## 8.2 PEM スケジュール済みジョブの管理

指定した順序で定義した一連のステップを実行するために、PEM スケジュールジョブを作成できます。これらの手順には、エージェントにバインドされたサーバーで実行できる SQL コードまたはバッチ/シェルスクリプトを含めることができます。これらのジョブは、ビジネス要件に合わせてスケジュールできます。たとえば、特定のデータベースサーバーのバックアップを作成するジョブを作成し、毎月特定の日時に実行するようにスケジュールできます。

PEM スケジュール ジョブを作成または管理するには、PEM ツリーを使用して、ジョブを作成する PEM エージェントを参照します。ツリーには **Jobs** ノードが表示され、その下に現在定義されているジョブが一覧表示されます。ジョブを追加するには、**Jobs** ノードを右クリックし、コンテキスト メニューから **Create Job** を選択します。

[作成 - エージェント ジョブ] ダイアログ ボックスのタブを使用して、PEM スケジュール ジョブを構成する手順とスケジュールを定義します。

[一般] タブを使用して、ジョブに関する一般的な情報を提供します。

- **名前** フィールドにジョブの名前を入力します。
- **Enabled** スイッチを **Yes** に設定して、ジョブを有効にします。ジョブを無効にするには **No** に設定します。
- **コメント** フィールドを使用して、ジョブに関するメモを保存します。

ステップタブを使用して、ジョブが実行するステップを定義および管理します。追加 (+) を選択してステップを追加します。次に、ヘッダーの左側にある作成アイコンを選択して、[ステップの定義] ダイアログ ボックスを開きます。

[ステップの定義] ダイアログ ボックスを使用して、ステップを定義します。

- **名前** フィールドにステップの名前を入力します。手順は、名前の英数字順に実行されます。
- **Enabled** スイッチを使用して、ジョブの実行時にステップを含める (**True**) か、ステップを無効にする (**False**)。
- **Kind** スイッチを使用して、ジョブ ステップが SQL コード (**SQL**) またはバッチ スクリプト (**Batch**) を呼び出すかどうかを示します。
- **SQL** を選択した場合は、**コード** タブを使用して、ステップの SQL コードを指定します。- [**バッチ**] を選択した場合は、[**コード**] タブを使用して、ステップ中に実行するバッチ スクリプトを指定します。
- **Onerror** リストを使用して、ステップの実行中にエラーが発生した場合の pgAgent の動作を指定します。以下から選択:
  - **失敗**—このステップの処理中にエラーが発生した場合は、ジョブを停止してください。- **成功**—ステップを正常に完了したものとマークし、続行します。- **無視**—エラーを無視して続行します。
- **Kind** スイッチの入力として **SQL** を選択した場合:
- **Server** フィールドを使用して、PEM スケジュールジョブを作成するエージェントにバインドされているサーバーを指定します。- **データベース** フィールドを使用して、選択したサーバーに関連付けられているデータベースを指定します。
- **コメント** フィールドを使用して、ステップに関するコメントを入力します。
- [ステップ定義] ダイアログ ボックスの [コード] タブの状況依存フィールドを使用して、ステップ中に実行する SQL コードまたはバッチスクリプトを指定します。
- ステップが SQL コードを呼び出す場合は、**SQL クエリ** フィールドに 1 つ以上の SQL ステートメントを指定します。
- ステップがバッチ スクリプトを呼び出す場合は、**コード** フィールドにスクリプトを指定します。Windows サーバーで実行している場合は、標準のバッチファイル構文を使用します。Linux サーバーでは、最初の行で適切なインタープリター (#!/bin/sh など) を指定すれば、任意のシェルスクリプトを使用できます。定義されたインラインコードに加えて、ファイル システム上の任意のバッチ スクリプト、シェルスクリプト、または SQL ファイルのパスを指定することもできます。
- Linux システムでスクリプトを呼び出すには、agent.cfg ファイルの batch\_script\_user パラメータのエントリを変更し、スクリプトを実行するユーザーを指定する必要があります。このパラメータには、root 以外のユーザーまたは root のいずれかを指定できます。ユーザーを指定しない場合、または指定したユーザーが存在しない場合、スクリプトは実行されません。ファイルを変更した後、エージェントを再起動します。
- Windows システムでスクリプトを呼び出すには、AllowBatchJobSteps のレジストリ エントリを true に設定し、PEM エージェントを再起動します。PEM レジストリ エントリは HKEY\_LOCAL\_MACHINE\Software\Wow6432Node\EnterpriseDB\PEM\agent にあります。

ステップで必要なすべての情報を入力したら、**保存** を選択します。

**追加 (+)** を選択して各ステップを追加するか、**スケジュール** タブを選択してジョブ スケジュールを定義します。

[ **スケジュール** ] タブで、**追加 (+)** を選択して、ジョブのスケジュールを追加します。次に、ヘッダーの左側にある作成アイコンを選択して、[スケジュールの定義] ダイアログ ボックスを開きます。

**スケジュール定義** タブを使用して、ジョブを実行する日時を指定します。

- **名前** フィールドにスケジュールの名前を入力します。
- **Enabled** スイッチを使用して、pgAgent がスケジュールを使用する (**はい**) か、スケジュールを無効にする (**いいえ**) かを示します。
- **Start** フィールドのカレンダーセレクトアを使用して、スケジュールの開始日時を指定します。
- **End** フィールドのカレンダーセレクトアを使用して、スケジュールの終了日時を指定します。
- **コメント** フィールドを使用して、スケジュールに関するコメントを提供します。

[ **繰り返し** ] タブを選択して、cron スタイルの形式でスケジュールを実行する日を定義します。ジョブは、[ **繰り返し** ] タブで選択された日付または時刻要素ごとに実行されます。

各フィールドで値を選択して、フィールドの選択値のリストに追加します。フィールドから値をクリアするには、フィールドの右側にある X を選択します。

- **Days** ボックスのフィールドを使用して、ジョブを実行する日を指定します。 - **WeekDays** フィールドを使用して、ジョブを実行する日を選択します。 - **MonthDays** フィールドを使用して、ジョブを実行する日数を選択します。日付に関係なく、月の最終日にジョブを実行するには、**最終日** を選択します。 - **Months** フィールドを使用して、ジョブを実行する月を選択します。
- **Times** ボックスのフィールドを使用して、ジョブを実行する時間を時間と分で指定します。

[ **例外** ] タブを選択して、スケジュールを実行しない日を指定します。たとえば、休日にジョブを実行したくない場合があります。

**追加 (+)** を選択して、例外テーブルに行を追加します。それから：

- [ **日付** ] 列で、カレンダーセクターを開き、ジョブを実行したくない日付を選択します。\*\*\*\* <Any> \*\*\*\* を指定して、選択した時刻にジョブを実行したくないことを示します。
- [ **時間** ] 列で、時間セクターを開き、ジョブを実行したくない時間を指定します。\*\*\*\* <Any> \*\*\*\* を指定して、選択した日のいつでもジョブを実行したくないことを示します。

**通知** タブを選択して、ジョブ レベルで電子メール通知設定を構成します。

- **通知を送信** フィールドを使用して、電子メール通知をいつ送信するかを指定します。
- **メールグループ** フィールドを使用して、メール通知を送信するメールグループを指定します。

スケジュールの定義が完了したら、**SQL** タブを使用して、ジョブを作成または変更するコードを確認できます。

**保存** を選択して、ジョブ定義を保存します。

ジョブを保存すると、そのジョブは、それが定義されたサーバーの PEM ツリーの **Jobs** ノードの下に一覧表示されます。PEM コンソールの [プロパティ] タブには、選択したジョブの概要が表示され、[統計] タ

ブには、ジョブの各実行の詳細が表示されます。既存のジョブを変更したり、ジョブに関する詳細情報を確認したりするには、ジョブ名を右クリックし、コンテキストメニューから [ プロパティ ] を選択します。

## 8.3 エージェント権限

デフォルトでは、PEM エージェントは、オペレーティング システム ホストの root 権限とデータベースサーバーのスーパーユーザー権限でインストールされます。これらの権限により、PEM エージェントは監視対象のホストとデータベースサーバーでシステムの使用状況に関する無制限のプロープを呼び出し、情報を取得して PEM サーバーに返すことができます。

PEM エージェントの権限が減少すると、PEM の機能が低下します。完全な機能を得るには、PEM エージェントを root として実行します。PEM エージェントがデータベース サーバーのサービスアカウントで実行されている場合、PEM プロープはレポートの生成に使用される統計情報に完全にアクセスできず、機能はそのアカウントの機能に制限されます。PEM エージェントが別の権限の低いアカウントで実行されている場合、機能はさらに制限されます。

PEM エージェントのオペレーティング システム権限を制限すると、一部の PEM プロープが情報を返さず、次の機能が影響を受ける可能性があります。

**注釈:** リストは包括的ではありませんが、制限されている機能の種類の概要を提供します。

Probe or action	Operating system	PEM functionality affected
Data And Logfile Analysis	Linux/ Windows	The Postgres Expert can't access complete information.
Session Information	Linux	The per-process statistics are incomplete.
PG HBA	Linux/ Windows	The Postgres Expert can't access complete information.
Service restart functionality	Linux/ Windows	The Audit Log Manager, Server Log Manager Log Analysis Expert, and PEM might not be able to apply requested modifications.
Package Deployment	Linux/ Windows	PEM can't run downloaded installation modules.
Batch Task	Windows	PEM can't run scheduled batch jobs in Windows.
Collect data from server (root access required)	Linux/ Windows	Columns such as swap usage, CPU usage, IO read, IO write appear as 0 in the session activity dashboard.

PEM エージェントのデータベース権限を制限すると、次の PEM 機能が影響を受ける可能性があります：

Probe	Operating system	PEM functionality affected
Audit Log Collection	Linux/Windows	PEM receives empty data from the PEM database.
Server Log Collection	Linux/Windows	PEM can't collect server log information.
Database Statistics	Linux/Windows	The Database/Server Analysis dashboards contain incomplete information.
Session Waits/System Waits	Linux/Windows	The Session/System Waits dashboards contain incomplete information.
Locks Information	Linux/Windows	The Database/Server Analysis dashboards contain incomplete information.
Streaming Replication	Linux/Windows	The Streaming Replication dashboard doesn't display information.
Slony Replication	Linux/Windows	Slony-related charts on the Database Analysis dashboard don't display information.
Tablespace Size	Linux/Windows	The Server Analysis dashboard doesn't display complete information.
xDB Replication	Linux/Windows	PEM can't send xDB alerts and traps.

プローブが十分な権限を持たずにオペレーティングシステムを照会している場合、プローブは **permission denied** エラーを返すことがあります。プローブが十分な特権なしでデータベースをクエリしている場合、プローブは **permission denied** エラーを返すか、返されたデータを PEM チャートまたはグラフに空の値として表示する可能性があります。

プローブが失敗すると、プローブの名前、プローブが失敗した理由、および問題の解決に役立つヒントを含むエントリがログファイルに書き込まれます。

サーバーで発生したプローブ関連のエラーをプローブ ログダッシュボードで表示したり、PEM ワーカー ログファイルでエラーメッセージを確認したりできます。Linux では、ログファイルのデフォルトの場所は次のとおりです。

```
/var/log/pem/worker.log
```

Windows では、イベント ビューアでログ情報を利用できます。

## 8.4 エージェント構成

多数の設定可能なパラメータとレジストリ エントリが、PEM エージェントの動作を制御します。一部の PEM 機能を有効にするには、PEM エージェントのパラメータ設定を変更する必要がある場合があります。PEM エージェント構成ファイルの値を変更したら、PEM エージェントを再起動して変更を適用します。

PEM\_MAXCONN パラメータを除いて、PEM 機能を有効にするために変更が必要でない限り、最初に EDB サポートの専門家に相談することなく、これらの構成パラメータまたはレジストリエントリを変更しないことを強くお勧めします。

Linux システムでは、PEM 構成オプションは `/usr/edb/pem/agent/etc` にある `agent.cfg` ファイルに保存されます。 `agent.cfg` ファイルには、次のエントリが含まれています。

Parameter name	Description	Default value
<code>pem_host</code>	PEM サーバーの IP アドレスまたはホスト名。	127.0.0.1.
<code>pem_port</code>	エージェントが PEM サーバーと通信するために接続するデータベースサーバーのポート。	Port 5432.
<code>pem_agent</code>	PEM エージェントに割り当てられた一意の識別子。	The first agent is '1', the second agent is '2', and so on.
<code>agent_ssl_key</code>	PEM エージェントのキー ファイルへの完全なパス。	<code>/root/.pem/agent.key</code>
<code>agent_ssl_cert</code>	PEM エージェントの証明書ファイルへの完全なパス。	<code>/root/.pem/agent.crt</code>
<code>agent_flag_dir</code>	HA サポートに使用されます。別のサーバーの監視を引き継ぐための要求をチェックするディレクトリパスを指定します。要求は、指定されたフラグ ディレクトリ内のファイルの形式で行われます。	Not set by default.
<code>log_level</code>	ログレベルは、PEM ログ ファイルに書き込むイベントのタイプ ( <i>debug2</i> 、 <i>debug</i> 、 <i>info</i> 、 <i>warning</i> 、 <i>error</i> のいずれか) を指定します。これらはログの詳細度の降順です。 <i>debug2</i> は可能なすべてをログに記録し、 <i>error</i> はエラーのみをログに記録します。	warning
<code>log_location</code>	PEM ワーカーログ ファイルの場所を指定します。	127.0.0.1.
<code>agent_log_location</code>	PEM エージェント ログ ファイルの場所を指定します。	<code>/var/log/pem/agent.log</code>
<code>long_wait</code>	最初の接続試行が失敗した場合に PEM サーバーへの接続を試行する前に PEM エージェントが待機する最大時間 (秒単位)。	30 seconds
<code>short_wait</code>	実行を待機しているキュー内の次のプローブをチェックする前に、PEM エージェントが待機する最小時間 (秒単位)。	10 seconds
<code>alert_threads</code>	エージェントによって生成されるアラートスレッドの数。詳細については、 <a href="#">アラートスレッドについて</a> を参照してください。	Set to 1 for the agent that resides on the host of the PEM server, 0 for all other agents.

次のページに続く

表 3 – 前のページからの続き

Parameter name	Description	Default value
enable_smtp	複数の PEM エージェント (7.13 以前) に対して true に設定すると、より多くの重複した電子メールが送信される可能性があります。一方、PEM エージェント (7.14 以降) の場合、送信される重複メールが少なくなる可能性があります。	true for PEM server host, false for all others.
enable_snmp	複数の PEM エージェント (7.13 以前) に対して true に設定すると、より多くの重複トラップが送信される可能性があります。一方、PEM エージェント (7.14 以降) の場合、送信する重複トラップが少なくなる可能性があります。	true for PEM server host, false for all others.
enable_nagios	true に設定すると、Nagios アラートが有効になります。	true for PEM server host, false for all others.
enable_webhook	true に設定すると、Webhook アラートが有効になります。	true for PEM server host, false for all others.
max_webhook_retries	失敗時に pemAgent が Webhook の呼び出しを再試行する最大回数を設定します。	Default 3.
connect_timeout	エージェントが接続を待機する最大秒数 (10 進数の整数文字列)。	Not set by default. Set to 0 to indicate for the agent to wait indefinitely.
allow_server_restart	TRUE に設定すると、エージェントは監視対象のデータベース サーバーを再起動できます。このパラメーターの値に応じて、一部の PEM 機能が有効または無効になる場合があります。	False
max_connections	接続スロットルによって使用されるプローブ接続の最大数。	0 (an unlimited number)
connection_lifetime	ConnectionLifetime (または connection_lifetime) を使用して、開いているがアイドル状態の接続を保持する最小秒数を指定します。MaxConnections で指定された値に達し、待機中の要求を満たすために別のデータベースへの新しい接続が必要な場合、このパラメーターは無視されます。	By default, set to 0 (a connection is dropped when the connection is idle after the agent's processing loop).
allow_batch_probes	TRUE に設定すると、ユーザーはカスタム プローブ機能を使用してバッチ プローブを作成できません。	false
heartbeat_connection	TRUE に設定すると、ハートビートの送信に専用接続が使用されます。	false
batch_script_dir	スクリプト ファイル (アラート用) が保存されているパスを指定します。	/tmp
connection_custom_setup	監視対象サーバーとの新しい接続が確立されたときに呼び出す SQL コードを提供するために使用します。	Not set by default.
ca_file	CA 証明書が存在するパスを指定します。	Not set by default.

次のページに続く

表 3 – 前のページからの続き

Parameter name	Description	Default value
batch_script_user	バッチ/シェル スクリプトの実行に使用するユーザーの名前を指定します。	None
webhook_ssl_key	Webhook の SSL クライアント キー ファイルへの完全なパス。	
webhook_ssl_crt	Webhook の SSL クライアント証明書ファイルへの完全なパス。	
webhook_ssl_crl	Webhook サーバー証明書を検証するための CRL ファイルの完全なパス。	
webhook_ssl_ca_crt	Webhook の SSL ca 証明書ファイルへの完全なパス。	
allow_insecure_webhooks	true に設定すると、Webhook が安全でないフラグを使用して呼び出すことができます。	false

64 ビット Windows システムでは、PEM レジストリエントリは次の場所にあります。

HKEY\_LOCAL\_MACHINE\Software\Wow6432Node\EnterpriseDB\PEM\agent

レジストリには、表に示すエントリが含まれています。

Parameter name	Description	Default value
PEM_HOST	PEM サーバーの IP アドレスまたはホスト名。	127.0.0.1.
PEM_PORT	エージェントが PEM サーバーと通信するために接続するデータベースサーバーのポート。	Port 5432.
AgentID	PEM エージェントに割り当てられた一意の識別子。	The first agent is '1', the second agent is '2', and so on.
AgentKeyPath	PEM エージェントのキー ファイルへの完全なパス。	%APP-DATA%\Roaming\pem\agent.key.
AgentCrtPath	PEM エージェントの証明書ファイルへの完全なパス。	%APP-DATA%\Roaming\pem\agent.crt
AgentFlagDir	HA サポートに使用されます。別のサーバーの監視を引き継ぐための要求をチェックするディレクトリ パスを指定します。要求は、指定されたフラグ ディレクトリ内のファイルの形式で行われます。	Not set by default.
LogLevel	PEM ログ ファイルに書き込むイベントのタイプを指定します。これらはログの詳細度の降順です。 <i>debug2</i> は可能なすべてをログに記録し、 <i>error</i> はエラーのみをログに記録します。	warning
LongWait	最初の接続試行が失敗した場合に、PEM サーバーへの接続を試行する前に PEM エージェントが待機する最大時間 (秒単位)。	30 seconds

次のページに続く

表 4 – 前のページからの続き

Parameter name	Description	Default value
shortWait	キュー内の次のプローブ (実行待ち) を確認する前に、PEM エージェントが待機する最小時間 (秒単位)。	10 seconds
AlertThreads	エージェントが生成するアラート スレッドの数。詳細については、 <a href="#">アラートスレッドについて</a> を参照してください。	Set to 1 for the agent that resides on the host of the PEM server, 0 for all other agents.
EnableSMTP	true に設定すると、SMTP 電子メール機能が有効になります。	true for PEM server host, false for all others.
EnableSNMP	true に設定すると、SNMP トラップ機能が有効になります。	true for PEM server host, false for all others.
EnableWebhook	true に設定すると、Webhook アラートが有効になります。	true for PEM server host, false for all others.
MaxWebhookRetries	失敗時に pemAgent が Webhook の呼び出しを再試行する最大回数を設定します。	Default 3.
ConnectTimeout	エージェントが接続を待機する最大秒数 (10 進数の整数文字列)。	Not set by default. If set to 0, the agent waits indefinitely.
AllowServerRestart	TRUE に設定すると、エージェントは監視対象のデータベース サーバーを再起動できます。このパラメーターの値に応じて、一部の PEM 機能が有効または無効になる場合があります。	true
MaxConnections	接続スロットルによって使用されるプローブ接続の最大数。	0 (an unlimited number)
ConnectionLifetime	ConnectionLifetime (または connection_lifetime) を使用して、開いているがアイドル状態の接続を保持する最小秒数を指定します。MaxConnections で指定された値に達し、待機中の要求を満たすために別のデータベースへの新しい接続が必要な場合、このパラメーターは無視されます。	By default, set to 0 (a connection is dropped when the connection is idle after the agent's processing loop).
AllowBatchProbes	TRUE に設定すると、ユーザーはカスタム プローブ機能を使用してバッチ プローブを作成できません。	false
HeartbeatConnection	TRUE に設定すると、ハートビートの送信に専用接続が使用されます。	false
BatchScriptDir	アラート用のスクリプト ファイルを保存するパスを指定します。	/tmp
ConnectionCustom-Setup	監視対象サーバーとの新しい接続が確立されたときに呼び出す SQL コードを提供するために使用します。	Not set by default.
ca_file	CA 証明書が存在するパスを指定します。	Not set by default.
AllowBatchJobSteps	true に設定すると、管理者ユーザー アカウントを使用してバッチ/シェルスクリプトが実行されます。	None

次のページに続く

表 4 – 前のページからの続き

Parameter name	Description	Default value
WebhookSSLKey	Webhook の SSL クライアント キー ファイルへの完全なパス。	
WebhookSSLCrt	Webhook の SSL クライアント証明書ファイルへの完全なパス。	
WebhookSSLCrl	Webhook サーバー証明書を検証するための CRL ファイルの完全なパス。	
WebhookSSLCaCrt	Webhook の SSL ca 証明書ファイルへの完全なパス。	
AllowInsecureWebhooks	true に設定すると、Webhook が安全でないフラグを使用して呼び出すことができます。	false

### 8.4.1 アラートスレッドについて

エージェントによって生成されるアラートスレッドの数は、`alert_threads` または `AlertThreads` パラメータによって決まります。一般に、PEM サーバーに存在するエージェントではこのパラメーターを 1 に設定し、他のすべてのエージェントでは 0 に設定することをお勧めします。ただし、非常に多数のアラートを持つ PEM サーバー・インスタンス (多数の監視対象サーバー、多数の有効なアラート、またはアラート頻度が高いことが原因) で、構成された頻度でアラートが評価されていない場合は、このパラメーターを増やす必要がある場合があります。この状況では、このパラメーターを PEM サーバーに常駐するエージェントで約 8 に設定し、他のすべてのエージェントでは 0 に設定し、それに応じて調整することをお勧めします。

このパラメーターを調整するときは、どのエージェントもどのアラートも処理できることを理解することが重要です。そのため、一般に、複数のエージェントでゼロ以外の数のアラートスレッドを使用する必要はありません。アラートを処理する PEM インスタンスの容量は、すべてのエージェントにわたるアラートスレッドの総数によって決まります。特定のエージェントのスレッド数を増やしても、そのエージェントによって監視されるサーバーに関連するアラートのパフォーマンスは向上しません。

各アラート スレッドは PEM サーバーバックエンドへの接続を開くため、必要以上のスレッドを割り当てると、PEM サーバーのメモリと CPU の使用量が増加します。

## 8.5 エージェントのプロパティ

[PEM エージェントのプロパティ] ダイアログ ボックスには、ダイアログボックスを開いた元の PEM エージェントに関する情報が表示されます。ダイアログボックスを開くには、PEM クライアントツリーでエージェント名を右クリックし、コンテキスト メニューから [プロパティ] を選択します。

PEM エージェントのプロパティダイアログボックスを使用して、PEM エージェントに関する情報を確認または変更します:

- **説明** フィールドには、PEM エージェントの変更可能な説明が表示されます。この説明は、PEM クライアントのツリーに表示されます。
- グループを使用して、PEM クライアントツリーでサーバーとエージェントを整理できます。グループリストを使用して、エージェントが表示されるグループを選択します。

- **Team** フィールドを使用して、エージェントが監視するサーバーにアクセスできるグループロールの名前を指定します。このエージェントによって監視されるサーバーは、接続されたチームメンバーに PEM クライアントツリーで表示されます。これは便利な機能です。 **チーム** フィールドは、真の分離を提供しません。セキュリティ目的で使用しないでください。
- **ハートビート間隔** フィールドには、PEM エージェントから PEM サーバーへのレポート間の経過時間が表示されます。 **分** または **秒** フィールドの横にあるセレクトターを使用して、間隔を変更します。

**ジョブ通知** タブを使用して、エージェントレベルで電子メール通知設定を構成します。

- **デフォルト設定を上書きしますか?** スイッチを使用して、エージェントレベルのジョブ通知設定でデフォルトのジョブ通知設定を上書きするかどうかを指定します。[ **はい** ] を選択して、このダイアログボックスの残りの設定を有効にして、いつ、誰にジョブ通知を送信するかを定義します。 - **ジョブ完了時に電子メールを送信しますか?** スイッチを使用して、ジョブが正常に完了したときにジョブ通知を送信するかどうかを指定します。 - **ジョブの失敗時に電子メールを送信しますか?** スイッチを使用して、ジョブが失敗したときにジョブ通知を送信するかどうかを指定します。 - **電子メールグループ** フィールドを使用して、ジョブ通知を送信する電子メールグループを指定します。

**エージェント構成** タブには、エージェントの現在の構成と機能がすべて表示されます。

- **Parameter** 列には、パラメーターのリストが表示されます。 - **Value** 列には、対応するパラメータの現在の値が表示されます。 - **Category** 列には、対応するパラメータのカテゴリが表示されます。 **構成** または **機能** のいずれかです。

## 第9章 Managing certificates

PEM サーバーバックアップデータベースのデータディレクトリに保存されているファイルには、PEM サーバーが安全な接続を使用するのに役立つ情報が含まれています。

- `ca_certificate.crt`
- `ca_key.key`
- `server.crt`
- `server.key`
- `root.crl`
- `root.crt`

PEM サーバーと共にインストールされる PEM エージェントは、`ca_certificate.crt` ファイルの有効期限を監視します。証明書の有効期限が近づくと、PEM:

- 既存の証明書ファイルのバックアップを作成します。
- 新しい証明書ファイルを作成し、新しい CA 証明書ファイルを PEM サーバー上の `root.crt` ファイルに追加します。
- アクティブなエージェントの証明書ファイルを更新するジョブを作成します。
- PEM サーバーを再起動します。

エージェントをアンインストールすると、そのエージェントに関連付けられた証明書が `root.crl` ファイルに保持されている証明書失効リストに追加され、証明書を使用して PEM サーバーに接続できないようになります。

証明書ファイルを手動で置き換えることができます。

### 9.1 SSL 証明書の置き換え

既存の PEM インストールの SSL 証明書を置き換えることができます。サーバーを同時に新しいバージョンにアップグレードする場合は、SSL 証明書を置き換える前に、すべての PEM インストーラー (最初にサーバーインストーラー、次にエージェント インストーラー) を呼び出します。それから：

1. 最初にサーバーホストで、次に監視対象ノードで、実行中のすべての PEM エージェントを停止します。

Linux ホストで PEM エージェントを停止するには、ターミナルウィンドウを開き、スーパーユーザー権限を想定して、次のコマンドを入力します。

Linux で systemd を使用する場合 (Centos 7 または 8 など)

```
systemctl stop pemagent
```

Windows ホストでは、Services アプレットを使用して PEM エージェントを停止できます。PEM エージェント サービスは、Postgres Enterprise Manager Agent という名前です。[サービス] ダイアログボックスでサービス名を選択し、[サービスを停止する]を選択します。

2. 既存の SSL キーと証明書のバックアップを取ります。SSL キーと証明書は、PEM インストールの下の data ディレクトリに保存されます。たとえば、Linux システムのデフォルトの場所は次のとおりです。

/var/lib/pgsql/x/data、x は PostgreSQL データベースのバージョン

次のファイルをコピーし、各ファイルに拡張子を追加して名前を一意にします。

- ca\_certificate.crt - ca\_key.key - root.crt - root.crl - server.key - server.crt

たとえば、エントリに old という単語が追加された ca\_certificate ファイルのバックアップを作成するには、次のコマンドを使用します。

```
cp ca_certificate.crt ca_certificate_old.crt
```

3. openssl\_rsa\_generate\_key() 関数を使用して ca\_key.key ファイルを生成します。

```
/usr/pgsql-x.x/bin/psql -U postgres -d pem --no-psqlrc -t -A -c "SELECT public.
→openssl_rsa_generate_key(1024)" > /var/lib/pgsql/x/data/ca_key.key
```

ca\_key.key ファイルを作成した後、ca\_certificate.crt ファイルを生成するときに使用する変数 CA\_KEY の内容を cat にします。ca\_key.key ファイルの権限を変更します。

```
CA_KEY=$(cat /var/lib/pgsql/x/data/ca_key.key)
```

```
chmod 600 /var/lib/pgsql/x/data/ca_key.key
```

4. キーを使用して ca\_certificate.crt ファイルを生成します。簡単にするために、一意の名前を持つ一時ファイルに SQL クエリを配置します。

```
echo "SELECT openssl_csr_to_crt(openssl_rsa_key_to_csr(${CA_KEY},
PEM,US, MA, Bedford, Postgres Enterprise Manager,
support@enterprisedb.com), NULL,
/var/lib/pgsql/x/data/ca_key.key)" > /tmp/_random.$$
```

次に、変数を使用してクエリを実行し、コンテンツを ca\_certificate.crt ファイルに配置します。

```
/usr/pgsql-x.x/bin/psql -U postgres -d pem --no-psqlrc -t -A -f /tmp/_random.$$ > /var/
→lib/pgsql/x/data/ca_certificate.crt
```

ca\_certificate.crt ファイルの権限を変更し、SQL コマンドを含む一時ファイルを削除します。

```
chmod 600 /var/lib/pgsql/x/data/ca_certificate.crt

rm -f /tmp/_random.$$
```

5. ca\_certificate.crt ファイルを root.crt ファイルとして再利用します。

```
cp /var/lib/pgsql/x/data/ca_certificate.crt /var/lib/pgsql/x/data/root.crt
```

root.crt ファイルの権限を変更します。

```
chmod 600 /var/lib/pgsql/x/data/root.crt
```

6. openssl\_rsa\_generate\_crl() 関数を使用して、証明書失効リスト (root.crl) を作成します。

```
/usr/pgsql-x/bin/psql -U postgres -d pem --no-psqlrc -t -A -c
"SELECT openssl_rsa_generate_crl(/var/lib/pgsql/x/data/ca_certificate.crt, /var/lib/
→pgsql/x/data/ca_key.key)" > /var/lib/pgsql/x/data/root.crl
```

root.crl ファイルの権限を変更します。

```
chmod 600 /var/lib/pgsql/x/data/root.crl
```

7. openssl\_rsa\_generate\_key() 関数を使用して server.key ファイルを生成します。

```
/usr/pgsql-x/bin/psql -U postgres -d pem --no-psqlrc -t -A -c "SELECT public.openssl_
→rsa_generate_key(1024)" >> /var/lib/pgsql/x/data/server.key
```

server.key ファイルを作成した後、server.crt ファイルを生成するときに使用する変数 SSL\_KEY の内容を cat し、server.key ファイルの権限を変更します。

```
SSL_KEY=$(cat /var/lib/pgsql/x/data/server.key)
```

```
chmod 600 /var/lib/pgsql/x/data/server.key
```

8. SSL\_KEY を使用してサーバー証明書を生成します。証明書を server.crt ファイルに保存します。簡単にするために、最初に SQL クエリを一意的な名前の一時ファイルに配置します。

```
echo "SELECT openssl_csr_to_crt(openssl_rsa_key_to_csr(${SSL_KEY},
PEM,US, MA, Bedford, Postgres Enterprise Manager,
support@enterprisedb.com),
/var/lib/pgsql/x/data/ca_certificate.crt,
/var/lib/pgsql/x/data/ca_key.key)" > /tmp/_random.$$

/usr/pgsql-x/bin/psql -U postgres -d pem --no-psqlrc -t -A -f /tmp/_random.$$ >> /
→var/lib/pgsql/x/data/server.crt
```

9. server.crt ファイルの権限を変更し、一時ファイルを削除します。

```
chmod 600 /var/lib/pgsql/x/data/server.crt

rm -f /tmp/_random.$$
```

10. Postgres サーバーを再起動します。

Linux で `init.d` を使用する場合 (Centos6 ホストなど):

```
/etc/init.d/postgresql-x restart
```

Linux で `systemd` を使用する場合 (Centos7 ホストなど):

```
systemctl restart postgresql-x
```

## 9.2 エージェント SSL 証明書の更新

PEM サーバーと対話するエージェントごとに、次のことを行う必要があります:

- RSA キーと証明書を生成します。
- キーと証明書をエージェントにコピーします。
- エージェントを再起動します。

各エージェントには、`pem` データベースの `pem.agent` テーブルに格納されている一意の識別子があります。キーおよび証明書ファイルを、エージェントの識別子に対応するキーまたは証明書に置き換える必要があります。次のキーファイルペアを生成する前に、手順 2 および 3 で生成された `agent.key` および `agent.crt` ファイルをそれぞれの PEM エージェントホストに移動する必要があります。後続のコマンドは、以前に生成されたファイルを上書きします。

PEM エージェント キー ファイル ペアを生成するには:

1. `psql` を使用して、エージェントの数と対応する識別子を見つけます。

```
/usr/pgsql-x/bin/psql -U postgres -d pem --no-psqlrc -t -A -c "SELECT ID FROM pem.
→agent"
```

- Linux では、`/etc/postgres-reg.ini` ファイルの `PEMAgent` セクションでエージェント識別子とキーと証明書の場所を見つけることもできます。
- Windows では、情報はレジストリに保存されます。
- 64 ビットの Windows インストールでは、以下を確認します。

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\EnterpriseDB\PEM\agent
```

- 32 ビットの Windows インストールでは、以下を確認します。

```
HKEY_LOCAL_MACHINE\SOFTWARE\EnterpriseDB\PEM\agent
```

2. キー ファイルが必要なエージェントを特定したら、エージェントごとに `agent.key` を生成します。キーを生成するには、次のコマンドを実行して、出力をファイルにキャプチャします。

```
/usr/pgsql-x/bin/psql -U postgres -d pem --no-psqlrc -t -A -c "SELECT openssl_rsa_
→generate_key(1024)" > agent.key
```

`agent.key` ファイルの権限を変更します。

```
chmod 600 agent.key
```

3. 各エージェントの証明書を生成します。証明書を生成するには、次のコマンドを実行して、出力を証明書ファイルにキャプチャします。

```
/usr/pgsql-x/bin/psql -U postgres -d pem --no-psqlrc -t -A -c
"SELECT openssl_csr_to_cert(openssl_rsa_key_to_csr($(cat agent.key),
agent<$ID>, US, MA, Bedford, Postgres Enterprise Manager,
support@enterprisedb.com),
/var/lib/pgsql/x/data/ca_certificate.crt,
/var/lib/pgsql/x/data/ca_key.key)" > agent.crt
```

`$ID` は、エージェントのエージェント番号です (psql コマンドラインによって取得されます)。

4. `agent.crt` ファイルの権限を変更します。

```
chmod 600 agent.crt
```

5. PEM エージェントサービスを再起動する前に、各エージェントのキーと証明書ファイルを新しく生成されたファイルに置き換えます。

`init.d` を使用する Linux では、次のコマンドでサービスを再起動します。

```
/etc/init.d/pemagent start
```

`systemd` を使用する Linux では、次のコマンドでサービスを再起動します。

```
systemctl start pemagent
```

Windows ホストでは、Services アプレットを使用して PEM エージェントを開始できます。PEM エージェント サービスは、Postgres Enterprise Manager Agent という名前です。[サービス] ダイアログボックスでサービス名を選択し、[サービスを開始する]を選択します。



## 第10章 Managing configuration settings

Postgre Enterprise Manager は、起動時に複数の構成ファイルを読み取ります。

- **config.py** —これはメインの構成ファイルです。変更しないでください。次のファイルのいずれかがオーバーライドされる可能性がある構成設定のリファレンスとして使用できます。
- **config\_distro.py** —このファイルは **config.py** の後に読み込まれ、パッケージャーが Postgres Enterprise Manager ディストリビューションに必要な設定を変更するためのものです。これには通常、特定のパスとファイルの場所が含まれます。このファイルはオプションです。パッケージャーは、必要に応じて **config.py** と同じディレクトリに作成できます。
- **config\_local.py** —このファイルは **config\_distro.py** の後に読み込まれ、エンドユーザーがローカルの設定や標準に合わせて調整したいデフォルトまたはパッケージ固有の設定を変更することを目的としています。このファイルはオプションです。必要に応じて、**config.py** と同じディレクトリに作成します。

デフォルトの **config.py** ファイルのコピーは、参照用に PEM オンラインヘルプに含まれています。



## 第11章 Troubleshooting agent issues

PEM エージェントで発生する可能性のあるこれらの問題をトラブルシューティングできます。

### 11.1 削除された PEM エージェントの復元

エージェントが `pem.agent` テーブルから削除された場合、それを復元することはできません。エージェントを再登録するには、`pemworker` ユーティリティを使用する必要があります。

エージェントが PEM Web クライアントから削除されたが、`pem.agent` テーブルに `active = f` の値を持つエントリがまだある場合、エージェントを復元できます。

1. `id` および `active` フィールドの値を確認します。

```
pem=# SELECT * FROM pem.agent;
```

2. `pem.agent` テーブルでエージェントのステータスを `true` に更新します。

```
pem=# UPDATE pem.agent SET active=true WHERE id=<x>;
```

`x` は、手順 1 で使用したクエリの出力に表示された識別子です。

3. PEM Web クライアントを更新します。

削除されたエージェントが復元されます。ただし、そのエージェントにバインドされたサーバーがダウンしているように見える場合があります。この問題を解決するには、サーバーの PEM エージェントプロパティを変更して、バインドされたエージェントを再度追加します。変更が正常に完了すると、サーバーは適切に実行されているように見えます。

### 11.2 コマンドラインを使用して、ダウンまたは不明なステータスの PEM エージェントを削除する

そのようなエージェントの数が多い場合、PEM Web インターフェイスを使用してダウンまたは不明ステータスの PEM エージェントを削除するのは困難な場合があります。この状況では、コマンドラインインターフェイスを使用して、停止中または不明なエージェントを削除できます。

次のクエリを使用して、*N* 時間以上停止しているエージェントを削除します。

```
UPDATE pem.agent SET active=false WHERE id IN
(SELECT a.id FROM pem.agent
a JOIN pem.agent_heartbeat b ON (b.agent_id=a.id)
```

(次のページに続く)

(前のページからの続き)

```
WHERE a.id IN
(SELECT agent_id FROM pem.agent_heartbeat WHERE (EXTRACT (HOUR FROM now())-
EXTRACT (HOUR FROM last_heartbeat)) > <N>));
```

次のクエリを使用して、ステータスが不明なエージェントを削除します。

```
UPDATE pem.agent SET active=false WHERE id IN
(SELECT id FROM pem.agent WHERE id NOT IN
(SELECT agent_id FROM pem.agent_heartbeat));
```